

SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Springfield Public Schools
Springfield, New Jersey 07081

AP Computer Science A
Grade 10-12

Curriculum Guide

2024

This curriculum may be modified through varying materials, techniques, and strategies as per an individual student's IEP.

Springfield Board of Education
Approved August 26, 2024



Springfield Public Schools
Springfield, New Jersey

Dr. Rachel Goldberg
Superintendent of Schools

Erica Scudero
Director of Curriculum and Instruction

Curriculum Developed By:

Dr. Norman Francis ~ Principal of Jonathan Dayton High School

Gregory Salmon ~ STEM Supervisor

Mary Ellen Powers ~ Math and Computer Science Teacher

CollegeBoard AP

INCLUDES
- Course framework
- Instructional section
- Sample exam questions

AP® Computer Science A

COURSE AND EXAM DESCRIPTION

Effective Fall 2020

AP CollegeBoard

AP® Computer Science A

COURSE AND EXAM DESCRIPTION

Effective
Fall 2020

AP COURSE AND EXAM DESCRIPTIONS ARE UPDATED PERIODICALLY
Please visit AP Central (apcentral.collegeboard.org) to determine whether a more recent course and exam description is available.

About College Board

College Board is a mission-driven, not-for-profit organization that connects students to college success and opportunity. Founded in 1900, College Board was created to expand access to higher education. Today, the membership association is made up of more than 6,000 of the world's leading educational institutions and is dedicated to promoting excellence and equity in education. Each year, College Board helps more than seven million students prepare for a successful transition to college through programs and services in college readiness and college success—including the SAT® and the Advanced Placement® Program. The organization also serves the education community through research and advocacy on behalf of students, educators, and schools.

For further information, visit collegeboard.org.

AP Equity and Access Policy

College Board strongly encourages educators to make equitable access a guiding principle for their AP programs by giving all willing and academically prepared students the opportunity to participate in AP. We encourage the elimination of barriers that restrict access to AP for students from ethnic, racial, and socioeconomic groups that have been traditionally underrepresented. Schools should make every effort to ensure that their AP classes reflect the diversity of their student population. College Board also believes that all students should have access to academically challenging course work before they enroll in AP classes, which can prepare them for AP success. It is only through a commitment to equitable preparation and access that true equity and excellence can be achieved.

Designers: Sonny Mui and Bill Tully

© 2020 College Board. College Board, Advanced Placement, AP, AP Central, and the acorn logo are registered trademarks of College Board. All other products and services may be trademarks of their

Contents

v **Acknowledgments**
1 **About AP**
4 **AP Resources and Supports**
6 **Instructional Model**
7 **About the AP Computer Science A Course**
7 College Course Equivalent
7 Prerequisites
7 Lab Requirement

COURSE FRAMEWORK

11 **Introduction**
13 **Course Framework Components**
15 Computational Thinking Practices
17 Course Content
22 **Course at a Glance**
27 **Unit Guides**
28 Using the Unit Guides
31 UNIT 1: Primitive Types
43 UNIT 2: Using Objects
63 UNIT 3: Boolean Expressions and `if` Statements
75 UNIT 4: Iteration
87 UNIT 5: Writing Classes
107 UNIT 6: Array
117 UNIT 7: `ArrayList`
131 UNIT 8: 2D Array
139 UNIT 9: Inheritance
151 UNIT 10: Recursion

INSTRUCTIONAL APPROACHES

161 **Selecting and Using Course Materials**
163 **Instructional Strategies**
170 **Developing Computational Thinking Practices**
181 **Teaching the Big Ideas**
182 **Using Strategies for Collaboration**
183 **Using Real-World Data**

EXAM INFORMATION

187 **Exam Overview**
191 **Sample Exam Questions**

SCORING GUIDELINES

207 Question 1: Methods and Control Structures
210 Applying the Scoring Criteria
213 Question 3: Array/ArrayList

APPENDIX

219 Java Quick Reference

Acknowledgments

College Board would like to acknowledge the contributors for their assistance with and commitment to the development of this course. All individuals and their affiliations were current at the time of contribution.

John Cigas, *Park University, Parkville, MO*

Sandy Czajka, *Riverside Brookfield High School, Riverside, IL*

Adrienne Decker, *University at Buffalo, Buffalo, NY*

Robert Glen Martin, *School for the Talented and Gifted, Dallas, TX*

We would also like to thank the following reviewers for their assistance: Owen Astrachan, Adam Cannon, George Fisher, Robert Getka, Ruth Hartsook, Cody Henrichsen, Paul Hilfinger, Allison Johnson, Liz Johnson, Robert Medrano, Sage Miller, Pat St. Clair, Lester Wainwright, Latrice Wicker, and Dianna Xu

College Board Staff

Becky Coutts, *Director, AP Instructional Design and PD Resource Development*

Crystal Furman, *Director, AP Computer Science Content Development*

Daniel McDonough, *Senior Director, AP Content Integration*

Allison Milverton, *Director, AP Curricular Publications*

Erin Spaulding, *Senior Director, AP Instructional Design and PD Resource Development*

Allison Thurber, *Executive Director, AP Curriculum and Assessment*

SPECIAL THANKS *John R. Williamson*

THIS PAGE IS INTENTIONALLY LEFT BLANK.

About AP

College Board's Advanced Placement® Program (AP®) enables willing and academically prepared students to pursue college-level studies—with the opportunity to earn college credit, advanced placement, or both—while still in high school. Through AP courses in 38 subjects, each culminating in a challenging exam, students learn to think critically, construct solid arguments, and see many sides of an issue—skills that prepare them for college and beyond. Taking AP courses demonstrates to college admission officers that students have sought the most challenging curriculum available to them, and research indicates that students who score a 3 or higher on an AP Exam typically experience greater academic success in college and are more likely to earn a college degree than non-AP students. Each AP teacher's syllabus is evaluated and approved by faculty from some of the nation's leading colleges and universities, and AP Exams are developed and scored by college faculty and experienced AP teachers. Most four-year colleges and universities in the United States grant credit, advanced placement, or both on the basis of successful AP Exam scores—more than 3,300 institutions worldwide annually receive AP scores.

AP Course Development

In an ongoing effort to maintain alignment with best practices in college-level learning, AP courses and exams emphasize challenging, research-based curricula aligned with higher education expectations.

Individual teachers are responsible for designing their own curriculum for AP courses, selecting appropriate college-level readings, assignments, and resources. This course and exam description presents the content and skills that are the focus of the corresponding college course and that appear on the AP Exam. It also organizes the content and skills into a series of units that represent a sequence found in widely adopted college textbooks and that many AP teachers have told us they follow in order to focus their instruction. The intention of this publication is to respect teachers' time and expertise by providing a roadmap that they can modify and adapt to their local priorities and preferences. Moreover, by organizing the AP course content and skills into units, the AP Program is able to provide teachers and students with formative assessments—Personal Progress Checks—that teachers can assign throughout the year to measure student progress as they acquire content knowledge and develop skills.

Enrolling Students: Equity and Access

College Board strongly encourages educators to make equitable access a guiding principle for their AP programs by giving all willing and academically prepared students the opportunity to participate in AP. We encourage the elimination of barriers that restrict access to AP for students from ethnic, racial, and socioeconomic groups that have been traditionally underserved. College Board also believes that all students should have access to academically challenging coursework before they enroll in AP classes, which can prepare them for AP success. It is only through a commitment to equitable preparation and access that true equity and excellence can be achieved.

Offering AP Courses: The AP Course Audit

The AP Program unequivocally supports the principle that each school implements its own curriculum that will enable students to develop the content understandings and skills described in the course framework.

While the unit sequence represented in this publication is optional, the AP Program does have a short list of curricular and resource requirements that must be fulfilled before a school can label a course "Advanced Placement" or "AP." Schools wishing to offer AP courses must participate in the AP Course Audit, a process through which AP teachers' course materials are reviewed by college faculty. The AP Course Audit was created to provide teachers and administrators with clear guidelines on curricular and resource requirements for AP courses and to help colleges and universities validate courses marked "AP" on students' transcripts. This process ensures that AP teachers' courses meet or exceed the curricular and resource expectations that college and secondary school faculty have established for college-level courses.

The AP Course Audit form is submitted by the AP teacher and the school principal (or designated administrator) to confirm awareness and understanding of the curricular and resource requirements. A syllabus or course outline, detailing how course requirements are met, is submitted by the AP teacher for review by college faculty.

Please visit collegeboard.org/apcourseaudit for more information to support the preparation and submission of materials for the AP Course Audit.

How the AP Program Is Developed

The scope of content for an AP course and exam is derived from an analysis of hundreds of syllabi and course offerings of colleges and universities. Using this research and data, a committee of college faculty and expert AP teachers work within the scope of the corresponding college course to articulate what students should know and be able to do upon the completion of the AP course. The resulting course framework is the heart of this course and exam description and serves as a blueprint of the content and skills that can appear on an AP Exam.

The AP Test Development Committees are responsible for developing each AP Exam, ensuring the exam questions are aligned to the course framework. The AP Exam development process is a multiyear endeavor; all AP Exams undergo extensive review, revision, piloting, and analysis to ensure that questions are accurate, fair, and valid, and that there is an appropriate spread of difficulty across the questions.

Committee members are selected to represent a variety of perspectives and institutions (public and private, small and large schools and colleges) and a range of gender, racial/ethnic, and regional groups. A list of each subject's current AP Test Development Committee members is available on apcentral.collegeboard.org.

Throughout AP course and exam development, College Board gathers feedback from various stakeholders in both secondary schools and higher education institutions. This feedback is carefully considered to ensure that AP courses and exams are able to provide students with a college-level learning experience and the opportunity to demonstrate their qualifications for advanced placement or college credit.

How AP Exams Are Scored

The exam scoring process, like the course and exam development process, relies on the expertise of both AP teachers and college faculty. While multiple-choice questions are scored by machine, the free-response questions are scored by thousands of college faculty and expert AP teachers. Most are scored at the annual AP Reading, while a small portion is scored online. All AP Readers are thoroughly trained, and their work is monitored throughout the Reading for fairness and consistency. In each subject, a highly respected college faculty member serves as Chief Faculty Consultant and, with the help of AP Readers in leadership positions, maintains the accuracy of the scoring standards. Scores on the free-response questions are weighted and combined with the results of the computer-scored multiple-choice questions, and this raw score is converted into a composite AP score on a 1–5 scale.

AP Exams are **not** norm-referenced or graded on a curve. Instead, they are criterion-referenced, which means that every student who meets the criteria for an AP score of 2, 3, 4, or 5 will receive that score, no matter how many students that is. The criteria for the number of points students must earn on the AP Exam to receive scores of 3, 4, or 5—the scores that research consistently validates for credit and placement purposes—include:

- The number of points successful college students earn when their professors administer AP Exam questions to them.
- The number of points researchers have found to be predictive that an AP student will succeed when placed into a subsequent, higher-level college course.
- Achievement-level descriptions formulated by college faculty who review each AP Exam question.

Using and Interpreting AP Scores

The extensive work done by college faculty and AP teachers in the development of the course and exam and throughout the scoring process ensures that AP Exam scores accurately represent students' achievement in the equivalent college course. Frequent and regular research studies establish the validity of AP scores as follows:

AP Score	Credit Recommendation	College Grade Equivalent
5	Extremely well qualified	A
4	Well qualified	A–, B+, B
3	Qualified	B–, C+, C
2	Possibly qualified	n/a
1	No recommendation	n/a

While colleges and universities are responsible for setting their own credit and placement policies, most private colleges and universities award credit and/or advanced placement for AP scores of 3 or higher. Additionally, most states in the U.S. have adopted statewide credit policies that assure college credit for scores of 3 or higher at public colleges and universities. To confirm a specific college's AP credit/placement policy, a search engine is available at apstudent.org/creditpolicies.

BECOMING AN AP READER

Each June, thousands of AP teachers and college faculty members from around the world gather for seven days in multiple locations to evaluate and score the free-response sections of the AP Exams. Ninety-eight percent of surveyed educators who took part in the AP Reading say it was a positive experience.

There are many reasons to consider becoming an AP Reader, including opportunities to:

- **Bring positive changes to the classroom:** Surveys show that the vast majority of returning AP Readers—both high school and college educators—make improvements to the way they teach or score because of their experience at the AP Reading.
- **Gain in-depth understanding of AP Exam and AP scoring standards:** AP Readers gain exposure to the quality and depth of the responses from the entire pool of AP Exam takers, and thus are better able to assess their students' work in the classroom.
- **Receive compensation:** AP Readers are compensated for their work during the Reading. Expenses, lodging, and meals are covered for Readers who travel.
- **Score from home:** AP Readers have online distributed scoring opportunities for certain subjects. Check collegeboard.org/apreading for details.
- **Earn Continuing Education Units (CEUs):** AP Readers earn professional development hours and CEUs that can be applied to PD requirements by states, districts, and schools.

How to Apply

Visit collegeboard.org/apreading for eligibility requirements and to start the application process.

AP Resources and Supports

By completing a simple activation process at the start of the school year, teachers and students receive access to a robust set of classroom resources.

AP Classroom

AP Classroom is a dedicated online platform designed to support teachers and students throughout their AP experience. The platform provides a variety of powerful resources and tools to provide yearlong support to teachers and enable students to receive meaningful feedback on their progress.

UNIT GUIDES

Appearing in this publication and on AP Classroom, these planning guides outline all required course content and skills, organized into commonly taught units. Each unit guide suggests a sequence and pacing of content, scaffolds skill instruction across units, organizes content into topics, and provides tips on taking the AP Exam.

PERSONAL PROGRESS CHECKS

Formative AP questions for every unit provide feedback to students on the areas where they need to focus. Available online, Personal Progress Checks measure knowledge and skills through multiple-choice questions with rationales to explain correct and incorrect answers, and free-response questions with scoring information. Because the Personal Progress Checks are formative, the results of these assessments cannot be used to evaluate teacher effectiveness or assign letter grades to students, and any such misuses are grounds for losing school authorization to offer AP courses.*

PROGRESS DASHBOARD

This dashboard allows teachers to review class and individual student progress throughout the year. Teachers can view class trends and see where students struggle with content and skills that will be assessed on the AP Exam. Students can view their own progress over time to improve their performance before the AP Exam.

AP QUESTION BANK

This online library of real AP Exam questions provides teachers with secure questions to use in their classrooms. Teachers can find questions indexed by course topics and skills, create customized tests, and assign them online or on paper. These tests enable students to practice and get feedback on each question.

* To report misuses, please call 877-274-6474 (International: +1-212-632-1781).

Digital Activation

In order to teach an AP class and make sure students are registered to take the AP Exam, teachers must first complete the digital activation process. Digital activation gives students and teachers access to the resources and gathers students' exam registration information online, eliminating most of the answer sheet bubbling that has added to testing time and fatigue.

AP teachers and students begin by signing in to My AP and completing a simple activation process at the start of the school year, which provides access to all AP resources, including AP Classroom.

To complete digital activation:

- Teachers and students sign in to, or create, their College Board accounts.
- Teachers confirm that they have added the course they teach to their AP Course Audit account and have had it approved by their school's administrator.
- Teachers or AP Coordinators, depending on who the school has decided is responsible, set up class sections so students can access AP resources and have exams ordered on their behalf.
- Students join class sections with a join code provided by their teacher or AP Coordinator.
- Students will be asked for additional registration information upon joining their first class section, which eliminates the need for extensive answer sheet bubbling on exam day.

While the digital activation process takes a short time for teachers, students, and AP Coordinators to complete, overall it helps save time and provides the following additional benefits:

- **Access to AP resources and supports:** Teachers have access to resources specifically designed to support instruction and provide feedback to students throughout the school year as soon as activation is complete.
- **Streamlined exam ordering:** AP Coordinators can create exam orders from the same online class rosters that enable students to access resources. The coordinator reviews, updates, and submits this information as the school's exam order in the fall.
- **Student registration labels:** For each student included in an exam order, schools will receive a set of personalized AP ID registration labels, which replaces the AP student pack. The AP ID connects a student's exam materials with the registration information they provided during digital activation, eliminating the need for pre-administration sessions and reducing time spent bubbling on exam day.
- **Targeted Instructional Planning Reports:** AP teachers will get Instructional Planning Reports (IPRs) that include data on each of their class sections automatically rather than relying on special codes optionally bubbled in on exam day.

Instructional Model

Integrating AP resources throughout the course can help students develop skills and conceptual understandings. The instructional model outlined below shows possible ways to incorporate AP resources into the classroom.

Plan

Teachers may consider the following approaches as they plan their instruction before teaching each unit.

- Review the overview at the start of each **unit guide** to identify essential questions, conceptual understandings, and skills for each unit.
- Use the **Unit at a Glance** table to identify related topics that build toward a common understanding, and then plan appropriate pacing for students.
- Identify useful strategies in the **Instructional Approaches** section to help teach the concepts and skills.

Teach

When teaching, supporting resources could be used to build students' conceptual understanding and their mastery of skills.

- Use the topic pages in the **unit guides** to identify the required content.
- Integrate the content with a skill, considering any appropriate scaffolding.
- Employ any of the instructional strategies previously identified.
- Use the available resources on the topic pages to bring a variety of assets into the classroom.

Assess

Teachers can measure student understanding of the content and skills covered in the unit and provide actionable feedback to students.

- At the end of each unit, use AP Classroom to assign students the online **Personal Progress Checks**, as homework or as an in-class task.
- Provide question-level feedback to students through answer rationales; provide unit- and skill-level feedback using the progress dashboard.
- Create additional practice opportunities using the **AP Question Bank** and assign them through AP Classroom.

About the AP Computer Science A Course

AP Computer Science A introduces students to computer science through programming. Fundamental topics in this course include the design of solutions to problems, the use of data structures to organize large sets of data, the development and implementation of algorithms to process data and discover new information, the analysis of potential solutions, and the ethical and social implications of computing systems. The course emphasizes object-oriented programming and design using the Java programming language.

College Course Equivalent

AP Computer Science A is equivalent to a first-semester, college-level course in computer science.

Prerequisites

It is recommended that a student in the AP Computer Science A course has successfully completed a first-year high school algebra course with a strong foundation of basic linear functions, composition of functions, and problem-solving strategies that require multiple approaches and collaborative efforts. In addition, students should be able to use a Cartesian (x, y) coordinate system to represent points on a plane. It is important that students and their advisers understand that any significant computer science course builds upon a foundation of mathematical reasoning that should be acquired before attempting such a course.

Computer Language

The AP Computer Science A course requires that solutions of problems be written in the Java programming language. Because the Java programming language is extensive, with far more features than could be covered in a single introductory course, the AP Computer Science A Exam covers a subset of Java.

Lab Requirement

The AP Computer Science A course must include a minimum of 20 hours of hands-on, structured lab experiences to engage students in individual or group problem solving. Thus, each AP Computer Science A course includes a substantial lab component in which students design solutions to problems, express their solutions precisely (e.g., in the Java programming language), test their solutions, identify and correct errors (when mistakes occur), and compare possible solutions. College Board has developed several labs that are aligned to the course framework that fulfill the 20-hour lab requirement. The class period recommendations provided in the unit guides account for the time needed to complete each lab activity as described in the lab guide.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

Course Framework


elloWorld.ja
*/
public class He
{
pu
ain

Introduction

Computer science involves problem-solving, hardware, and algorithms that help people utilize computers and incorporate multiple perspectives to address real-world problems in contemporary life. As the study of computer science continues to evolve, the careful design of the AP Computer Science A course strives to engage a diverse student population, including female and underrepresented students, by allowing them to discover the power of computer science through rewarding yet challenging concepts.

A well-designed, modern AP Computer Science A course that includes opportunities for students to collaborate to solve problems that interest them, as well as ones that use authentic data sources, can help address traditional issues of equity and access. Such a course can broaden participation in computing while providing a strong and engaging introduction to fundamental areas of the discipline.

The AP Computer Science A course reflects what computer science teachers, professors, and researchers have indicated are the main goals of an introductory, college-level computer science programming course:

- **Program Design and Algorithm Development**—Determine required code segments to produce a given output.
- **Code Logic**—Determine the output, value, or result of given program code given initial values.
- **Code Implementation**—Write and implement program code.
- **Code Testing**—Analyze program code for correctness, equivalence, and errors.
- **Documentation**—Describe the behavior and conditions that produce the specified results in a program.
- **Ethical Computing**—Understand the ethical and social implications of computer use.

Students practice the computer science skills of designing, developing, and analyzing their own programs to address real-world problems or pursue a passion.

Compatible Curricula

The AP Computer Science A course is compatible with the recommendations of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers Computing Society (IEEE-CS) in the "Software Development Fundamentals" knowledge area, which is recommended for an introductory course. The AP Computer Science A course also integrates topics from the "Programming Languages" and "Algorithms and Complexity" knowledge areas. Teachers can review the Computer Science Curricula from ACM and IEEE-CS to see their complete curriculum guidelines.

The AP Computer Science A course vertically aligns with subconcepts in the "Algorithms and Programming" core concept in the Computer Science Teachers Association (CSTA) K-12 Computer Science Framework. Subconcepts extended in the AP Computer Science A course include modularity, variables, and control. This vertical alignment allows K-12 CS teachers to make connections from their high school courses to the college-equivalent AP Computer Science course.

AP Computer Science Program

AP Computer Science A is one of two AP Computer Science courses available to students. The AP Computer Science Principles course complements AP Computer Science A by teaching the foundational areas of the discipline. Students can take these two courses in either order or concurrently, as allowed by their school.

Resource Requirements

Students should have access to a computer system that represents relatively recent technology. A school should ensure that each student has access to a computer for at least three hours a week; additional time is desirable. Student and instructor access to computers is important during class time, but additional time is essential for students to individually develop solutions to problems.

The computer system must allow students to create, edit, quickly compile, and execute Java programs comparable in size to those found in the AP Computer Science A Labs. It is highly desirable that these computers provide students with access to the internet. It is essential that each computer science teacher has internet access. A school must ensure that each student has a college-level text for individual use both inside and outside of the classroom. Schools are encouraged to provide copies of the lab materials for individual use.

Course Framework Components

Overview

This course framework provides a description of what students should know and be able to do to qualify for college credit or placement.

The course framework includes two essential components:

1 COMPUTATIONAL THINKING PRACTICES

The computational thinking practices are central to the study and practice of computer science. Students should practice and develop these skills on a regular basis over the span of the course.

2 COURSE CONTENT

The course content is organized into commonly taught units of study that provide a suggested sequence for the course. These units comprise the content and conceptual understandings that colleges and universities typically expect students to master to qualify for college credit and/or placement. This content is grounded in big ideas, which are cross-cutting concepts that build conceptual understanding and spiral throughout the course.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

1

AP COMPUTER SCIENCE A

Computational Thinking Practices

The table that follows presents computational thinking practices that students should develop during the AP Computer Science A course. The practices form the basis of tasks on the AP Exam.

The unit guides later in this publication embed and spiral these practices throughout the course, providing teachers with one way to integrate the practices into the course content with sufficient repetition to prepare students to transfer those skills when taking the AP Exam.

More detailed information about teaching the computational thinking practices can be found in the Instructional Approaches section of this publication.

AP COMPUTER SCIENCE A

Computational Thinking Practices: Skills

Practice 1	Practice 2	Practice 3	Practice 4	Practice 5
Program Design and Algorithm Development 1 Determine required code segments to produce a given output.	*Code Logic* 2 Determine the output, value, or result of given program code given initial values.	*Code Implementation* 3 Write and implement program code.	*Code Testing* 4 Analyze program code for correctness, equivalence, and errors.	*Documentation* 5 Describe the behavior and conditions that produce identified results in a program.
SKILLS				
1.A Determine an appropriate program design to solve a problem or accomplish a task (*not assessed*).	2.A Apply the meaning of specific operators.	3.A Write program code to create objects of a class and call methods.	4.A Use test-cases to find errors or validate results.	5.A Describe the behavior of a given segment of program code.
1.B Determine code that would be used to complete code segments.	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output).	3.B Write program code to define a new type by creating a class.	4.B Identify errors in program code.	5.B Explain why a code segment will not compile or work as intended.
1.C Determine code that would be used to interact with completed program code.	2.C Determine the result or output based on the statement execution order in a code segment containing method calls.	3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.	4.C Determine if two or more code segments yield equivalent results.	5.C Explain how the result of program code changes, given a change to the initial code.
	2.D Determine the number of times a code segment will execute.	3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.		5.D Describe the initial conditions that must be met for a program segment to work as intended or described.
		3.E Write program code to create, traverse, and manipulate elements in 2D array objects.		

2

AP COMPUTER SCIENCE A

Course Content

Based on the Understanding by Design® (Wiggins and McTighe) model, this course framework provides a description of the course requirements necessary for student success, with a focus on big ideas that encompass core principles, theories, and processes of the discipline. The framework also encourages instruction that prepares students for advanced computer science coursework and its integration into a wide array of STEM-related fields.

Big Ideas

The big ideas serve as the foundation of the course and help students to create meaningful connections among concepts. They are often overarching concepts or themes that become threads that run throughout the course. Revisiting the big ideas and applying them in a variety of contexts enables students to develop deeper conceptual understanding. Below are the big ideas of the course and a brief description of each.

BIG IDEA 1: MODULARITY (MOD)
Incorporating elements of abstraction, by breaking problems down into interacting pieces, each with their own purpose, makes writing complex programs easier. Abstracting simplifies concepts and processes by looking at the big picture rather than being overwhelmed by the details. Modularity in object-oriented programming allows us to use abstraction to break complex programs down into individual classes and methods.

BIG IDEA 2: VARIABLES (VAR)
Information used as a basis for reasoning, discussion, or calculation is referred to as *data*. Programs rely on variables to store data, on data structures to organize multiple values when program complexity increases, and on algorithms to sort, access, and manipulate this data. Variables create data abstractions, as they can represent a set of possible values or a group of related values.

continued on next page

BIG IDEA 3: CONTROL (CON)

Doing things in order, making decisions, and doing the same process multiple times are represented in code by using control structures and specifying the order in which instructions are executed. Programmers need to think algorithmically in order to define and interpret processes that are used in a program.

BIG IDEA 4: IMPACT OF COMPUTING (IOC)

Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues. As programmers, we need to understand how our programs will be used and be responsible for the consequences.

UNITS
The course content is organized into commonly taught units. The units have been arranged in a logical sequence frequently found in many college courses and textbooks.

The 10 units in AP Computer Science A, and their weighting on the multiple-choice section of the AP Exam, are listed below.

Pacing recommendations at the unit level and on the Course at a Glance provide suggestions for how to teach the required course content and administer the Personal Progress Checks. The suggested class periods are based on a schedule in which the class meets five days a week for 45 minutes each day. While these recommendations have been made to aid planning, teachers should of course adjust the pacing based on the needs of their students, alternate schedules (e.g., block scheduling), or their school's academic calendar.

TOPICS
Each unit is broken down into teachable segments called topics. The topic pages (starting on page 36) contain all required content for each topic. Although most topics can be taught in one or two class periods, teachers are again encouraged to pace their course to suit the needs of their students and school.

Units	Exam Weighting
Unit 1: Primitive Types	**2.5–5%**
Unit 2: Using Objects	**5–7.5%**
Unit 3: Boolean Expressions and `if` Statements	**15–17.5%**
Unit 4: Iteration	**17.5–22.5%**
Unit 5: Writing Classes	**5–7.5%**
Unit 6: Array	**10–15%**
Unit 7: `ArrayList`	**2.5–7.5%**
Unit 8: 2D Array	**7.5–10%**
Unit 9: Inheritance	**5–10%**
Unit 10: Recursion	**5–7.5%**

Spiraling the Big Ideas

The following table shows how the big ideas spiral across units.

Big Ideas	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5
	Primitive Types	Using Objects	Boolean Expressions and `if` Statements	Iteration	Writing Classes
Modularity MOD	✓	✓			✓
Variables VAR	✓	✓			✓
Control CON	✓	✓	✓	✓	
Impact of Computing IOC					✓

Spiraling the Big Ideas *(cont'd)*

Big Ideas	Unit 6	Unit 7	Unit 8	Unit 9	Unit 10
	Array	`ArrayList`	2D Array	Inheritance	Recursion
Modularity MOD				✓	
Variables VAR	✓	✓	✓		
Control CON	✓	✓	✓		✓
Impact of Computing IOC		✓			

Course at a Glance

Plan

The Course at a Glance provides a useful visual organization of the AP Computer Science A curricular components, including the following:

- Sequence of units, along with approximate weighting and suggested pacing. Please note, pacing is based on 45-minute class periods, meeting five days each week for a full academic year.
- Progression of topics within each unit.
- Spiraling of the big ideas and practices across units.

Teach

COMPUTATIONAL THINKING PRACTICES
Practices spiral across units.

1 Program Design and Algorithm Development
2 Code Logic
3 Code Implementation
4 Code Testing
5 Documentation

+ Indicates 3 or more skills/practices suggested for a given topic. The individual topic page will show all the suggested skills.

BIG IDEAS
Big ideas spiral across units.

MOD Modularity
VAR Variables
CON Control
IOC Impact of Computing

Assess

Assign the Personal Progress Checks—either as homework or in class—for each unit. Each Personal Progress Check contains formative multiple-choice questions and formative free-response questions that are written in a similar style to what students will experience on the end-of-year exam. Feedback from the Personal Progress Checks shows students the areas on

UNIT 1 Primitive Types

~8–10 Class Periods | 2.5–5% AP Exam Weighting

MOD VAR 2 4	1.1	Why Programming? Why Java?
VAR 1	1.2	Variables and Data Types
CON 1 2	1.3	Expressions and Assignment Statements
CON 2 5	1.4	Compound Assignment Operators
CON 2 5	1.5	Casting and Ranges of Variables

Personal Progress Check 1
Multiple-choice: ~25 questions

UNIT 2 Using Objects

~13–15 Class Periods | 5–7.5% AP Exam Weighting

MOD 5	2.1	Objects: Instances of Classes
MOD VAR 1 3	2.2	Creating and Storing Objects (Instantiation)
MOD 1 3	2.3	Calling a Void Method
MOD 2 3	2.4	Calling a Void Method with Parameters
MOD 1 3	2.5	Calling a Non-void Method
VAR 2	2.6	`String` Objects: Concatenation, Literals, and More
VAR 2 3	2.7	`String` Methods
VAR 2	2.8	Wrapper Classes: `Integer` and `Double`
MOD CON 1 3	2.9	Using the `Math` Class

Personal Progress Check 2
Multiple-choice: ~25 questions
Free-response: 1 question
- Methods and Control Structures: partial

NOTE: Partial versions of the free-response questions are provided to prepare students for



UNIT 3 Boolean Expressions and `if` Statements

~11–13 Class Periods | 15–17.5% AP Exam Weighting

- CON 2 — 3.1 Boolean Expressions
- CON 2 3 — 3.2 `if` Statements and Control Flow
- CON 3 4 — 3.3 `if-else` Statements
- CON 3 4 — 3.4 `else if` Statements
- CON 2 3 — 3.5 Compound Boolean Expressions
- CON 4 — 3.6 Equivalent Boolean Expressions
- CON 2 3 — 3.7 Comparing Objects

Personal Progress Check 3

Multiple-choice: ~20 questions

Free-response: 2 questions

- Methods and Control Structures
- Methods and Control Structures: partial

UNIT 4 Iteration

~14–16 Class Periods | 17.5–22.5% AP Exam Weighting

- CON + — 4.1 `while` Loops
- CON + — 4.2 `for` Loops
- CON 2 3 — 4.3 Developing Algorithms Using Strings
- CON + — 4.4 Nested Iteration
- CON 2 — 4.5 Informal Code Analysis

Personal Progress Check 4

Multiple-choice: ~15 questions

Free-response: 2 questions

- Methods and Control Structures
- Methods and Control Structures: partial

UNIT 5 Writing Classes

~12–14 Class Periods | 5–7.5% AP Exam Weighting

- MOD 1 — 5.1 Anatomy of a Class
- MOD 1 3 — 5.2 Constructors
- MOD 5 — 5.3 Documentation with Comments
- MOD 3 5 — 5.4 Accessor Methods
- MOD 3 4 — 5.5 Mutator Methods
- MOD 1 3 — 5.6 Writing Methods
- MOD 3 5 — 5.7 Static Variables and Methods
- VAR 3 5 — 5.8 Scope and Access
- VAR 2 — 5.9 `this` Keyword
- IOC N/A — 5.10 Ethical and Social Implications of Computing Systems

Personal Progress Check 5

Multiple-choice: ~25 questions

Free-response: 2 questions

- Class
- Class: partial



UNIT 6 Array

~6–8 Class Periods | 10–15% AP Exam Weighting

- VAR 1 3 — 6.1 Array Creation and Access
- VAR + — 6.2 Traversing Arrays
- VAR 3 4 — 6.3 Enhanced `for` Loop for Arrays
- CON + — 6.4 Developing Algorithms Using Arrays

Personal Progress Check 6

Multiple-choice: ~15 questions

Free-response: 2 questions

- Array and `ArrayList` (Array only)
- Array and `ArrayList` (Array only): partial

UNIT 7 ArrayList

~10–12 Class Periods | 2.5–7.5% AP Exam Weighting

- VAR 1 3 — 7.1 Introduction to `ArrayList`
- VAR 2 3 — 7.2 `ArrayList` Methods
- VAR 2 3 — 7.3 Traversing ArrayLists
- CON 3 4 — 7.4 Developing Algorithms Using ArrayLists
- CON 3 5 — 7.5 Searching
- CON 2 — 7.6 Sorting
- IOC N/A — 7.7 Ethical Issues Around Data Collection

Personal Progress Check 7

Multiple-choice: ~15 questions

Free-response: 1 question

- Array and `ArrayList` (`ArrayList` focus)

UNIT 8 2D Array

~10–12 Class Periods | 7.5–10% AP Exam Weighting

- VAR 1 3 — 8.1 2D Arrays
- VAR CON + — 8.2 Traversing 2D Arrays

Personal Progress Check 8

Multiple-choice: ~10 questions

Free-response: 1 question

- 2D Array

UNIT 9 Inheritance

~13–15 Class Periods | 5–10% AP Exam Weighting

- MOD 1 3 — 9.1 Creating Superclasses and Subclasses
- MOD 3 5 — 9.2 Writing Constructors for Subclasses
- MOD 3 5 — 9.3 Overriding Methods
- MOD 1 3 — 9.4 `super` Keyword
- MOD 3 5 — 9.5 Creating References Using Inheritance Hierarchies
- MOD 3 5 — 9.6 Polymorphism
- MOD 1 3 — 9.7 `Object` Superclass

Personal Progress Check 9

Multiple-choice: ~15 questions
Free-response: 2 questions
- Class
- Class: partial

UNIT 10 Recursion

~3–5 Class Periods | 5–7.5% AP Exam Weighting

- CON 1 5 — 10.1 Recursion
- CON 2 — 10.2 Recursive Searching and Sorting

Personal Progress Check 10

Multiple-choice: ~10 questions
Free-response: 1 question
- Methods and Control Structures (recursive and non-recursive solutions allowed)

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

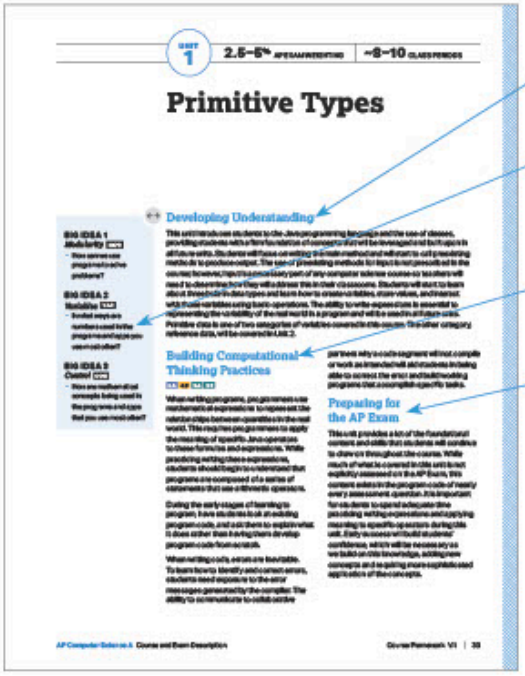
Unit Guides

Designed with input from the community of AP Computer Science A educators, the unit guides offer all teachers helpful guidance in building students' skills and knowledge. The suggested sequence was identified through a thorough analysis of the syllabi of highly effective AP teachers and the organization of typical college textbooks.

This unit structure respects new AP teachers' time by providing one possible sequence they can adopt or modify rather than having to build from scratch. An additional benefit is that these units enable the AP Program to provide interested teachers with formative assessments—the Personal Progress Checks—that they can assign their students at the end of each unit to gauge progress toward success on the AP Exam. However, experienced AP teachers who are pleased with their current course organization and exam results should feel no pressure to adopt these units, which comprise an optional, not mandatory, sequence for this course.



Using the Unit Guides



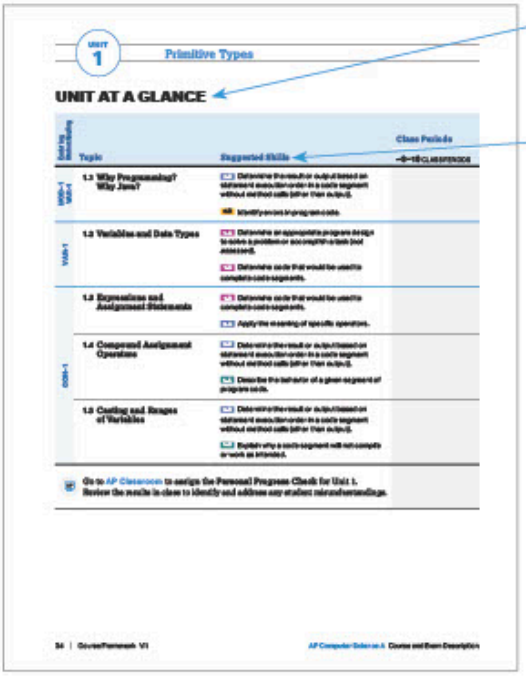
UNIT OPENERS

Developing Understanding provides an overview that contextualizes and situates the key content of the unit within the scope of the course.

The **big ideas** serve as the foundation of the course and develop understanding as they spiral throughout the course. The **essential questions** are thought-provoking questions that motivate students and inspire inquiry.

Building Computational Thinking Practices describes specific skills within the practices that are appropriate to focus on in that unit.

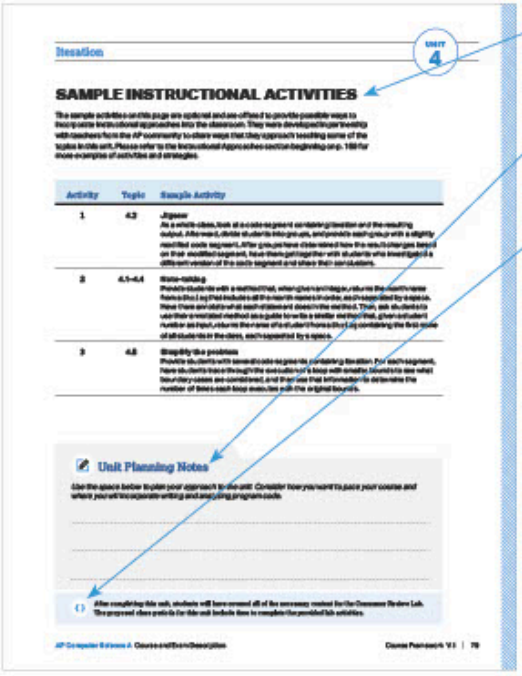
Preparing for the AP Exam provides helpful tips and common student misunderstandings identified from prior exam data.



The **Unit at a Glance** table shows the topics, related enduring understandings, and suggested skills. The Class Periods column has been left blank so that teachers can customize the time they spend on each topic.

The **suggested skills** for each topic show possible ways to link the content in that topic to specific AP Computer Science A skills. The individual skills have been thoughtfully chosen in a way that helps spiral and scaffold the skills throughout the course. The questions on the Personal Progress Checks are based on this pairing. However, AP Exam questions may pair the content with any of the skills.

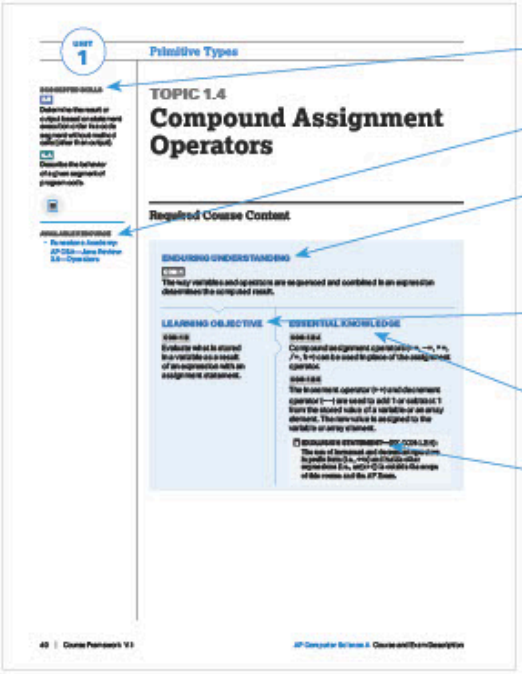
Using the Unit Guides



The **Sample Instructional Activities** page includes optional activities that can help teachers tie together the content and skill of a particular topic.

The **Unit Planning Notes** section provides space for teachers to make notes on their approach to the individual topics and the unit as a whole.

When enough content has been covered to complete an **AP Computer Science A lab**, an icon has been added to this section. Teachers can also choose to integrate lab activities earlier as they cover the content.



TOPIC PAGES

Suggested skills offers one or more possible skills to pair with the topic.

Where possible, **available resources** are listed that might help teachers address a particular topic in their classroom.

Enduring understandings are the long-term takeaways related to the big ideas that leave a lasting impression on students. Students build and earn these understandings over time by exploring and applying course content throughout the year.

Learning objectives define what a student should be able to do with content knowledge to progress toward the enduring understandings.

Essential knowledge statements describe the knowledge required to perform the learning objective.

Exclusion statements define content or specific details about content that does not need to be included in the course. The content in the exclusion statements will not be assessed on the AP Computer Science A Exam. While excluded content will not be assessed on the AP Exam, such content may be provided as background or additional information for the concepts and skills being assessed in lab activities.

REQUIRED COURSE CONTENT LABELING SYSTEM

BIG IDEA	ENDURING UNDERSTANDING	LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON Control	**CON-1** The way variables and operators are sequenced and combined in an expression determines the computed result.	**CON-1.B** Evaluate what is stored in a variable as a result of an expression with an assignment statement.	**CON-1.B.1** The assignment operator (=) allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left.

NOTE: Labels are used to distinguish each element of the required course content and are used throughout this course and exam description. Additionally, they are used in the AP Question Bank and other resources found on AP Classroom. Enduring understandings are labeled sequentially according to the big idea that they are related to. Learning objectives are labeled to correspond with the enduring understanding they relate to. Finally, essential knowledge statements are labeled to correspond with the learning objective to which they relate.

AP COMPUTER SCIENCE A

UNIT 1

Primitive Types

2.5–5%
AP EXAM WEIGHTING

~8–10
CLASS PERIODS

AP

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 1

Multiple-choice: ~25 questions

UNIT 1 | 2.5–5% AP EXAM WEIGHTING | ~8–10 CLASS PERIODS

Primitive Types

BIG IDEA 1
Modularity MOD
- How can we use programs to solve problems?

BIG IDEA 2
Variables VAR
- In what ways are numbers used in the programs and apps you use most often?

BIG IDEA 3
Control CON
- How are mathematical concepts being used in the programs and apps that you use most often?

Developing Understanding

This unit introduces students to the Java programming language and the use of classes, providing students with a firm foundation of concepts that will be leveraged and built upon in all future units. Students will focus on writing the main method and will start to call preexisting methods to produce output. The use of preexisting methods for input is not prescribed in the course; however, input is a necessary part of any computer science course so teachers will need to determine how they will address this in their classrooms. Students will start to learn about three built-in data types and learn how to create variables, store values, and interact with those variables using basic operations. The ability to write expressions is essential to representing the variability of the real world in a program and will be used in all future units. Primitive data is one of two categories of variables covered in this course. The other category, reference data, will be covered in Unit 2.

Building Computational Thinking Practices

2.A 4.B 5.A 5.B

When writing programs, programmers use mathematical expressions to represent the relationships between quantities in the real world. This requires programmers to apply the meaning of specific Java operators to these formulas and expressions. While practicing writing these expressions, students should begin to understand that programs are composed of a series of statements that use arithmetic operators.

During the early stages of learning to program, have students look at existing program code, and ask them to explain what it does rather than having them develop program code from scratch.

When writing code, errors are inevitable. To learn how to identify and correct errors, students need exposure to the error messages generated by the compiler. The ability to communicate to collaborative partners why a code segment will not compile or work as intended will aid students in being able to correct the error and build working programs that accomplish specific tasks.

Preparing for the AP Exam

This unit provides a lot of the foundational content and skills that students will continue to draw on throughout the course. While much of what is covered in this unit is not explicitly assessed on the AP Exam, this content exists in the program code of nearly every assessment question. It is important for students to spend adequate time practicing writing expressions and applying meaning to specific operators during this unit. Early success will build students' confidence, which will be necessary as we build on this knowledge, adding new concepts and requiring more sophisticated application of the concepts.

UNIT 1

Primitive Types

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~8–10 CLASS PERIODS
MOD-1 VAR-1	1.1 Why Programming? Why Java?	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 4.B Identify errors in program code.	
VAR-1	1.2 Variables and Data Types	1.A Determine an appropriate program design to solve a problem or accomplish a task (*not assessed*). 1.B Determine code that would be used to complete code segments.	
CON-1	1.3 Expressions and Assignment Statements	1.B Determine code that would be used to complete code segments. 2.A Apply the meaning of specific operators.	
	1.4 Compound Assignment Operators	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 5.A Describe the behavior of a given segment of program code.	
	1.5 Casting and Ranges of Variables	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 5.B Explain why a code segment will not compile or work as intended.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 1. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	1.1	**Error analysis** Provide students with code that contains syntax errors. Ask students to identify and correct the errors in the provided code. Once they feel they have identified and corrected all syntax errors, have them verify their conclusion by using a compiler and an IDE that does not autocorrect errors.
2	1.3	**Activating prior knowledge** The basic arithmetic operators of +, –, /, and * are similar to what students have experienced in math class or when using a calculator. Give students a list of expressions and ask them to apply what they know from math class to evaluate the meaning of the expressions. Have them verify their results by putting them into a compiler.
3	1.4	**Sharing and responding** Put student into groups of two. Provide each student with a different set of statements; in each pair, one student should have a list of statements that contain compound assignment operators, while the other student should have a list of statements that accomplish the same thing without using compound statements. Be sure the statements are in a different order. Students should take turns describing what a statement does to their partner, and the partner should determine which statement of theirs is equivalent to the one being described.
4	1.5	**Predict and compare** Provide students with several statements that involve casting. Each cast should be on a different value in the statement. Have students predict the resulting value. For any statements that would not compile or work as intended, have students explain the problem and propose a solution. They should verify their results by putting those results into a compiler.

UNIT 1

Primitive Types

SUGGESTED SKILLS

2.B
Determine the result or output based on statement execution order in a code segment without method calls (other than output).

4.B
Identify errors in program code.

AVAILABLE RESOURCE

- Runestone Academy: AP CSA—Java Review: 2.1—What is Java?

TOPIC 1.1
Why Programming? Why Java?

Required Course Content

ENDURING UNDERSTANDING

MOD-1
Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.A
Call `System` class methods to generate output to the console.

ESSENTIAL KNOWLEDGE

MOD-1.A.1
`System.out.print` and `System.out.println` display information on the computer monitor.

MOD-1.A.2
`System.out.println` moves the cursor to a new line after the information has been displayed, while `System.out.print` does not.

ENDURING UNDERSTANDING

VAR-1
To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.A
Create string literals.

ESSENTIAL KNOWLEDGE

VAR-1.A.1
A string literal is enclosed in double quotes.

TOPIC 1.2
Variables and Data Types

SUGGESTED SKILLS

1.A
Determine an appropriate program design to solve a problem or accomplish a task.

1.B
Determine code that would be used to complete code segments.

AVAILABLE RESOURCE
- Runestone Academy: AP CSA—Java Review: 3—Variables

Required Course Content

ENDURING UNDERSTANDING

VAR-1
To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
VAR-1.B Identify the most appropriate data type category for a particular specification.	**VAR-1.B.1** A type is a set of values (a domain) and a set of operations on them. **VAR-1.B.2** Data types can be categorized as either primitive or reference. **VAR-1.B.3** The primitive data types used in this course define the set of operations for numbers and Boolean values.
VAR-1.C Declare variables of the correct types to represent primitive data.	**VAR-1.C.1** The three primitive data types used in this course are `int`, `double`, and `boolean`. **VAR-1.C.2** Each variable has associated memory that is used to hold its value. **VAR-1.C.3** The memory associated with a variable of a primitive type holds an actual primitive value. **VAR-1.C.4** When a variable is declared `final`, its value cannot be changed once it is initialized.

UNIT 1

Primitive Types

SUGGESTED SKILLS

1.B
Determine code that would be used to complete code segments.

2.A
Apply the meaning of specific operators.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 3.5—Operators
- Problets: Arithmetic Expressions in Java

TOPIC 1.3
Expressions and Assignment Statements

Required Course Content

ENDURING UNDERSTANDING

CON-1
The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.A
Evaluate arithmetic expressions in a program code.

ESSENTIAL KNOWLEDGE

CON-1.A.1
A literal is the source code representation of a fixed value.

CON-1.A.2
Arithmetic expressions include expressions of type `int` and `double`.

CON-1.A.3
The arithmetic operators consist of `+`, `-`, `*`, `/`, and `%`.

CON-1.A.4
An arithmetic operation that uses two `int` values will evaluate to an `int` value.

CON-1.A.5
An arithmetic operation that uses a `double` value will evaluate to a `double` value.

CON-1.A.6
Operators can be used to construct compound expressions.

CON-1.A.7
During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped.

CON-1.A.8
An attempt to divide an integer by zero will result in an `ArithmeticException` to occur.

continued on next page

Primitive Types

UNIT 1

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON-1.B Evaluate what is stored in a variable as a result of an expression with an assignment statement.	CON-1.B.1 The assignment operator (=) allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left. CON-1.B.2 During execution, expressions are evaluated to produce a single value. CON-1.B.3 The value of an expression has a type based on the evaluation of the expression.

UNIT 1

Primitive Types

SUGGESTED SKILLS

2.B
Determine the result or output based on statement execution order in a code segment without method calls (other than output).

5.A
Describe the behavior of a given segment of program code.

AVAILABLE RESOURCE

- Runestone Academy: AP CSA—Java Review: 3.5—Operators

TOPIC 1.4
Compound Assignment Operators

Required Course Content

ENDURING UNDERSTANDING

CON-1
The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.B
Evaluate what is stored in a variable as a result of an expression with an assignment statement.

ESSENTIAL KNOWLEDGE

CON-1.B.4
Compound assignment operators (+=, −=, *=, /=, %=) can be used in place of the assignment operator.

CON-1.B.5
The increment operator (++) and decrement operator (−−) are used to add 1 or subtract 1 from the stored value of a variable or an array element. The new value is assigned to the variable or array element.

EXCLUSION STATEMENT—(EK CON-1.B.5): The use of increment and decrement operators in prefix form (i.e., ++x) and inside other expressions (i.e., arr[x++]) is outside the scope of this course and the AP Exam.

TOPIC 1.5
Casting and Ranges of Variables

SUGGESTED SKILLS

2.B
Determine the result or output based on statement execution order in a code segment without method calls (other than output).

5.B
Explain why a code segment will not compile or work as intended.

AVAILABLE RESOURCE
- Runestone Academy: AP CSA—Java Review: 3.6—Casting

Required Course Content

ENDURING UNDERSTANDING

CON-1
The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.C
Evaluate arithmetic expressions that use casting.

ESSENTIAL KNOWLEDGE

CON-1.C.1
The casting operators `(int)` and `(double)` can be used to create a temporary value converted to a different data type.

CON-1.C.2
Casting a `double` value to an `int` causes the digits to the right of the decimal point to be truncated.

CON-1.C.3
Some programming code causes `int` values to be automatically cast (widened) to `double` values.

CON-1.C.4
Values of type `double` can be rounded to the nearest integer by `(int)(x + 0.5)` or `(int)(x – 0.5)` for negative numbers.

CON-1.C.5
Integer values in Java are represented by values of type `int`, which are stored using a finite amount (4 bytes) of memory. Therefore, an `int` value must be in the range from `Integer.MIN_VALUE` to `Integer.MAX_VALUE` inclusive.

CON-1.C.6
If an expression would evaluate to an `int` value outside of the allowed range, an integer overflow occurs. This could result in an incorrect value within the allowed range.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

UNIT 2

Using Objects

5–7.5%
AP EXAM WEIGHTING

~13–15
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 2

Multiple-choice: ~25 questions
Free-response: 1 question

- Methods and Control Structures: partial

UNIT 2 | 5–7.5% AP EXAM WEIGHTING | ~13–15 CLASS PERIODS

Using Objects

BIG IDEA 1
Modularity MOD
- How can we simulate election results using existing program code?

BIG IDEA 2
Variables VAR
- How are appropriate variables chosen to represent a remote control?

BIG IDEA 3
Control CON
- How do the games we play simulate randomness?

Developing Understanding

In the first unit, students used primitive types to represent real-world data and determined how to use them in arithmetic expressions to solve problems. This unit introduces a new type of data: reference data. Reference data allows real-world objects to be represented in varying degrees specific to a programmer's purpose. This unit builds on students' ability to write expressions by introducing them to `Math` class methods to write expressions for generating random numbers and other more complex operations. In addition, strings and the existing methods within the `String` class are an important topic within this unit. Knowing how to declare variables or call methods on objects is necessary throughout the course but will be very important in Units 5 and 9 when teaching students how to write their own classes and about inheritance relationships.

Building Computational Thinking Practices

1.B 1.C 3.A

The study of computer science involves implementing the design or specification for a program. This is the fun and rewarding part of computer science, because it involves putting a plan into practice to create a runnable program. In addition to developing their own programs, students should practice completing partially written program code to fulfill a specification. This builds their confidence and provides them opportunities to be successful during these early stages of learning.

Programmers often rely on existing program code to build new programs. Using existing code saves time, because it has already been tested. By using the `String` class, students will learn how to interact with and utilize any existing Java class to create objects and call methods.

Preparing for the AP Exam

During the free-response portion of the exam, students will be required to call methods of classes that they haven't been exposed to prior to the exam. Students should get plenty of practice identifying the proper parameters to use when calling methods of classes that are provided to them.

Often, students struggle with free-response questions that require them to work with the `String` class. Using the Java Quick Reference (p. 219) regularly during class will help students become more familiar with this resource prior to the exam. Paying close attention to the method descriptions will ensure that students use the correct type and order of parameters when calling `String` methods.

Practice close reading techniques with students prior to the exam, such as underlining keywords, return types, and parameters. Students have approximately 20 minutes to read, process, and answer each of the four free-response questions. These close reading techniques are valuable in helping students process the questions quickly without inadvertently missing key information.

UNIT 2

Using Objects

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~13–15 CLASS PERIODS
MOD-1	2.1 Objects: Instances of Classes	5.A Describe the behavior of a given segment of program code.	
MOD-1 VAR-1	2.2 Creating and Storing Objects (Instantiation)	1.C Determine code that would be used to interact with completed program code. 3.A Write program code to create objects of a class and call methods.	
MOD-1	2.3 Calling a Void Method	1.C Determine code that would be used to interact with completed program code. 3.A Write program code to create objects of a class and call methods.	
MOD-1	2.4 Calling a Void Method with Parameters	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 3.A Write program code to create objects of a class and call methods.	
MOD-1	2.5 Calling a Non-void Method	1.C Determine code that would be used to interact with completed program code. 3.A Write program code to create objects of a class and call methods.	
VAR-1	2.6 `String` Objects: Concatenation, Literals, and More	2.A Apply the meaning of specific operators.	
VAR-1	2.7 `String` Methods	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 3.A Write program code to create objects of a class and call methods.	
VAR-1	2.8 Wrapper Classes: `Integer` and `Double`	2.C Determine the result or output based on the statement execution order in a code segment containing method calls.	
MOD-1 CON-1	2.9 Using the `Math` Class	1.B Determine code that would be used to complete code segments. 3.A Write program code to create objects of a class and call methods.	

Go to AP Classroom to assign the **Personal Progress Check** for Unit 2.
Review the results in class to identify and address any student misunderstandings.

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	2.1	**Using manipulatives** When introducing students to the idea of creating objects, you can use a cookie cutter and modeling clay or dough, with the cutter representing the class and the cut dough representing the objects. For each object cut, write the instantiation. Ask students to describe what the code is doing and how the different parameter values (e.g., thickness, color) change the object that was created.
2	2.2	**Marking the text** Provide students with several statements that define a variable and create an object on a single line. Have students mark up the statements by circling the assignment operator and the `new` keyword. Then, have students underline the variable type and the constructor. Lastly, have them draw a rectangle around the list of actual parameters being passed to the constructor. Using these marked-up statements, ask students to create several new variables and objects.
3	2.9	**Think-pair-share** Provide students with several code segments, each with a missing expression that would contain a call to a method in the `Math` class, and a description of the intended outcome of each code segment. Ask them which statement should be used to complete the code segment. Have them share their responses with a partner to compare answers and come to agreement, and then have groups share with the entire class.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate writing and analyzing program code.

UNIT 2

Using Objects

SUGGESTED SKILL

5.A

Describe the behavior of a given segment of program code.

AVAILABLE RESOURCE

- Runestone Academy: AP CSA—Java Review: 2.2—What is a Class and an Object?

TOPIC 2.1
Objects: Instances of Classes

Required Course Content

ENDURING UNDERSTANDING

MOD-1

Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.B

Explain the relationship between a class and an object.

ESSENTIAL KNOWLEDGE

MOD-1.B.1

An object is a specific instance of a class with defined attributes.

MOD-1.B.2

A class is the formal implementation, or blueprint, of the attributes and behaviors of an object.

TOPIC 2.2

Creating and Storing Objects (Instantiation)

SUGGESTED SKILLS

1.C
Determine code that would be used to interact with completed program code.

3.A
Write program code to create objects of a class and call methods.

Required Course Content

ENDURING UNDERSTANDING

MOD-1
Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.C
Identify, using its signature, the correct constructor being called.

ESSENTIAL KNOWLEDGE

MOD-1.C.1
A signature consists of the constructor name and the parameter list.

MOD-1.C.2
The parameter list, in the header of a constructor, lists the types of the values that are passed and their variable names. These are often referred to as formal parameters.

MOD-1.C.3
A parameter is a value that is passed into a constructor. These are often referred to as actual parameters.

MOD-1.C.4
Constructors are said to be overloaded when there are multiple constructors with the same name but a different signature.

MOD-1.C.5
The actual parameters passed to a constructor must be compatible with the types identified in the formal parameter list.

MOD-1.C.6
Parameters are passed using call by value. Call by value initializes the formal parameters with copies of the actual parameters.

continued on next page

UNIT 2

Using Objects

LEARNING OBJECTIVE

MOD-1.D

For creating objects:

a. Create objects by calling constructors without parameters.

b. Create objects by calling constructors with parameters.

ESSENTIAL KNOWLEDGE

MOD-1.D.1

Every object is created using the keyword `new` followed by a call to one of the class's constructors.

MOD-1.D.2

A class contains constructors that are invoked to create objects. They have the same name as the class.

MOD-1.D.3

Existing classes and class libraries can be utilized as appropriate to create objects.

MOD-1.D.4

Parameters allow values to be passed to the constructor to establish the initial state of the object.

ENDURING UNDERSTANDING

VAR-1

To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.D

Define variables of the correct types to represent reference data.

ESSENTIAL KNOWLEDGE

VAR-1.D.1

The keyword `null` is a special value used to indicate that a reference is not associated with any object.

VAR-1.D.2

The memory associated with a variable of a reference type holds an object reference value or, if there is no object, `null`. This value is the memory address of the referenced object.

TOPIC 2.3
Calling a Void Method

SUGGESTED SKILLS

1.C
Determine code that would be used to interact with completed program code.

3.A
Write program code to create objects of a class and call methods.

AVAILABLE RESOURCE

- Classroom Resources > GridWorld Case Study: Part I

Required Course Content

ENDURING UNDERSTANDING

MOD-1
Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.E
Call non-static void methods without parameters.

ESSENTIAL KNOWLEDGE

MOD-1.E.1
An object's behavior refers to what the object can do (or what can be done to it) and is defined by methods.

MOD-1.E.2
Procedural abstraction allows a programmer to use a method by knowing what the method does even if they do not know how the method was written.

MOD-1.E.3
A method signature for a method without parameters consists of the method name and an empty parameter list.

MOD-1.E.4
A method or constructor call interrupts the sequential execution of statements, causing the program to first execute the statements in the method or constructor before continuing. Once the last statement in the method or constructor has executed or a return statement is executed, flow of control is returned to the point immediately following where the method or constructor was called.

continued on next page

UNIT 2

Using Objects

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
MOD-1.E Call non-static void methods without parameters.	**MOD-1.E.5** Non-static methods are called through objects of the class. **MOD-1.E.6** The dot operator is used along with the object name to call non-static methods. **MOD-1.E.7** Void methods do not have return values and are therefore not called as part of an expression. **MOD-1.E.8** Using a `null` reference to call a method or access an instance variable causes a `NullPointerException` to be thrown.

TOPIC 2.4

Calling a Void Method with Parameters

Required Course Content

ENDURING UNDERSTANDING

MOD-1

Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.F

Call non-static void methods with parameters.

ESSENTIAL KNOWLEDGE

MOD-1.F.1

A method signature for a method with parameters consists of the method name and the ordered list of parameter types.

MOD-1.F.2

Values provided in the parameter list need to correspond to the order and type in the method signature.

MOD-1.F.3

Methods are said to be overloaded when there are multiple methods with the same name but a different signature.

SUGGESTED SKILLS

2.C

Determine the result or output based on the statement execution order in a code segment containing method calls.

3.A

Write program code to create objects of a class and call methods.

AVAILABLE RESOURCE

- Practice-It!: BJP4 Chapter 3: Parameters and Objects—Self-Check 3.2–3.9

UNIT 2

Using Objects

SUGGESTED SKILLS

1.C
Determine code that would be used to interact with completed program code.

3.A
Write program code to create objects of a class and call methods.

AVAILABLE RESOURCES

- The Exam > 2018 AP Computer Science A Exam Free-Response Question #1 (Frog Simulation)

TOPIC 2.5
Calling a Non-void Method

Required Course Content

ENDURING UNDERSTANDING

MOD-1
Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.G
Call non-static non-void methods with or without parameters.

ESSENTIAL KNOWLEDGE

MOD-1.G.1
Non-void methods return a value that is the same type as the return type in the signature. To use the return value when calling a non-void method, it must be stored in a variable or used as part of an expression.

TOPIC 2.6

String Objects: Concatenation, Literals, and More

Required Course Content

ENDURING UNDERSTANDING

VAR-1

To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.E

For `String` class:

a. Create `String` objects.

b. Call `String` methods.

ESSENTIAL KNOWLEDGE

VAR-1.E.1

`String` objects can be created by using string literals or by calling the `String` class constructor.

VAR-1.E.2

`String` objects are immutable, meaning that `String` methods do not change the `String` object.

VAR-1.E.3

`String` objects can be concatenated using the + or += operator, resulting in a new `String` object.

VAR-1.E.4

Primitive values can be concatenated with a `String` object. This causes implicit conversion of the values to `String` objects.

VAR-1.E.5

Escape sequences start with a \ and have a special meaning in Java. Escape sequences used in this course include `\"`, `\\`, and `\n`.

SUGGESTED SKILL

2.A

Apply the meaning of specific operators.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 4—Strings
- Practice-It!: BJP4 Chapter 3: Parameters and Objects—Self-Check 3.18

UNIT 2

Using Objects

SUGGESTED SKILLS

2.C
Determine the result or output based on the statement execution order in a code segment containing method calls.

3.A
Write program code to create objects of a class and call methods.

AVAILABLE RESOURCES

- Java Quick Reference (see Appendix)
- Runestone Academy: AP CSA—Java Review: 4.3—String Methods on the Exam
- CodingBat Java: String-1
- Practice-It!: BJP4 Chapter 3: Parameters and Objects—Self-Check 3.19 and 3.20

TOPIC 2.7
String Methods

Required Course Content

ENDURING UNDERSTANDING

VAR-1
To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.E
For `String` class:
a. Create `String` objects.
b. Call `String` methods.

ESSENTIAL KNOWLEDGE

VAR-1.E.6
Application program interfaces (APIs) and libraries simplify complex programming tasks.

VAR-1.E.7
Documentation for APIs and libraries are essential to understanding the attributes and behaviors of an object of a class.

VAR-1.E.8
Classes in the APIs and libraries are grouped into packages.

VAR-1.E.9
The `String` class is part of the `java.lang` package. Classes in the `java.lang` package are available by default.

VAR-1.E.10
A `String` object has index values from `0` to `length – 1`. Attempting to access indices outside this range will result in an `StringIndexOutOfBoundsException`.

VAR-1.E.11
A `String` object can be concatenated with an object reference, which implicitly calls the referenced object's `toString` method.

continued on next page

Using Objects

UNIT 2

LEARNING OBJECTIVE

VAR-1.E

For `String` class:

a. Create `String` objects.

b. Call `String` methods.

ESSENTIAL KNOWLEDGE

VAR-1.E.12

The following `String` methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

- `String(String str)` — Constructs a new `String` object that represents the same sequence of characters as `str`
- `int length()` — Returns the number of characters in a `String` object
- `String substring(int from, int to)` — Returns the substring beginning at index `from` and ending at index `to - 1`
- `String substring(int from)` — Returns `substring(from, length())`
- `int indexOf(String str)` — Returns the index of the first occurrence of `str`; returns `-1` if not found
- `boolean equals(String other)` — Returns `true` if `this` is equal to `other`; returns `false` otherwise
- `int compareTo(String other)` — Returns a value < 0 if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value > 0 if `this` is greater than `other`

VAR-1.E.13

A string identical to the single element substring at position `index` can be created by calling `substring(index, index + 1)`.

UNIT 2

Using Objects

SUGGESTED SKILL

2.C

Determine the result or output based on the statement execution order in a code segment containing method calls.

AVAILABLE RESOURCE

- Java Quick Reference (see Appendix)

TOPIC 2.8
Wrapper Classes: `Integer` and `Double`

Required Course Content

ENDURING UNDERSTANDING

VAR-1

To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.F

For wrapper classes:

a. Create `Integer` objects.
b. Call `Integer` methods.
c. Create `Double` objects.
d. Call `Double` methods.

ESSENTIAL KNOWLEDGE

VAR-1.F.1

The `Integer` class and `Double` class are part of the `java.lang` package.

VAR-1.F.2

The following `Integer` methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

- `Integer(int value)` — Constructs a new `Integer` object that represents the specified `int` value
- `Integer.MIN_VALUE` — The minimum value represented by an `int` or `Integer`
- `Integer.MAX_VALUE` — The maximum value represented by an `int` or `Integer`
- `int intValue()` — Returns the value of this `Integer` as an `int`

continued on next page

Using Objects

UNIT 2

LEARNING OBJECTIVE

VAR-1.E

For wrapper classes:

a. Create `Integer` objects.
b. Call `Integer` methods.
c. Create `Double` objects.
d. Call `Double` methods.

ESSENTIAL KNOWLEDGE

VAR-1.F.3

The following `Double` methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

- `Double(double value)` — Constructs a new `Double` object that represents the specified `double` value
- `double doubleValue()` — Returns the value of this `Double` as a `double`

VAR-1.F.4

Autoboxing is the automatic conversion that the Java compiler makes between primitive types and their corresponding object wrapper classes. This includes converting an `int` to an `Integer` and a `double` to a `Double`.

VAR-1.F.5

The Java compiler applies autoboxing when a primitive value is:

- Passed as a parameter to a method that expects an object of the corresponding wrapper class.
- Assigned to a variable of the corresponding wrapper class.

VAR-1.F.6

Unboxing is the automatic conversion that the Java compiler makes from the wrapper class to the primitive type. This includes converting an `Integer` to an `int` and a `Double` to a `double`.

VAR-1.F.7

The Java compiler applies unboxing when a wrapper class object is:

- Passed as a parameter to a method that expects a value of the corresponding primitive type.
- Assigned to a variable of the corresponding primitive type.

UNIT 2

Using Objects

SUGGESTED SKILLS

1.B
Determine code that would be used to complete code segments.

3.A
Write program code to create objects of a class and call methods.

AVAILABLE RESOURCES

- Java Quick Reference (see Appendix)
- Practice-It!: BJP4 Chapter 3: Parameters and Objects—Exercises 3.7 and 3.8

TOPIC 2.9
Using the `Math` Class

Required Course Content

ENDURING UNDERSTANDING

MOD-1
Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.H
Call static methods.

ESSENTIAL KNOWLEDGE

MOD-1.H.1
Static methods are called using the dot operator along with the class name unless they are defined in the enclosing class.

ENDURING UNDERSTANDING

CON-1
The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.D
Evaluate expressions that use the `Math` class methods.

ESSENTIAL KNOWLEDGE

CON-1.D.1
The `Math` class is part of the `java.lang` package.

CON-1.D.2
The `Math` class contains only static methods.

continued on next page

Using Objects

UNIT 2

LEARNING OBJECTIVE

CON-1.D

Evaluate expressions that use the `Math` class methods.

ESSENTIAL KNOWLEDGE

CON-1.D.3

The following static `Math` methods—including what they do and when they are used—are part of the Java Quick Reference:

- `int abs(int x)` — Returns the absolute value of an `int` value
- `double abs(double x)` — Returns the absolute value of a `double` value
- `double pow(double base, double exponent)` — Returns the value of the first parameter raised to the power of the second parameter
- `double sqrt(double x)` — Returns the positive square root of a `double` value
- `double random()` — Returns a double value greater than or equal to 0.0 and less than 1.0

CON-1.D.4

The values returned from `Math.random` can be manipulated to produce a random `int` or `double` in a defined range.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

UNIT 3

Boolean Expressions and `if` Statements

15–17.5%
AP EXAM WEIGHTING

~11–13
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 3

Multiple-choice: ~20 questions
Free-response: 2 questions

- Methods and Control Structures
- Methods and Control Structures: partial

UNIT 3 | 15–17.5% AP EXAM WEIGHTING | ~11–13 CLASS PERIODS

Boolean Expressions and `if` Statements

BIG IDEA 1
Control CON

- How can you use different conditional statements to write a pick-your-own-path interactive story?
- Why is selection a necessary part of programming languages?

Developing Understanding

Algorithms are composed of three building blocks: sequencing, selection, and iteration. This unit focuses on selection, which is represented in a program by using conditional statements. Conditional statements give the program the ability to decide and respond appropriately and are a critical aspect of any nontrivial computer program. In addition to learning the syntax and proper use of conditional statements, students will build on the introduction of Boolean variables by writing Boolean expressions with relational and logical operators.

The third building block of all algorithms is iteration, which you will cover in Unit 4. Selection and iteration work together to solve problems.

Building Computational Thinking Practices

2.B 3.C 4.A 4.C

Selection allows programmers to incorporate choice into their programs: to create games that react to interactions of the user, to develop simulations that are more real world by allowing for variability, or to discover new knowledge in a sea of information by filtering out irrelevant data. Students should be able to write program code that uses conditional statements for selection. Have students write their program code on paper and trace it with different sample inputs before writing code on the computer.

Programmers need to make design decisions when creating programs that determine how a program specification will be implemented. Typically, there are many ways in which statements can be written to yield the same result, and this final determination is dictated by the programmer. Exposing students to many different code segments that yield equivalent results allows them to be more confident in their solution and helps expose them to new ways of solving the problem that may be better than their current solution.

Preparing for the AP Exam

The study of computer science involves the analysis of program code. On the multiple-choice section of the exam, students will be asked to determine the result of a given program code segment based on specific input and on the behavior of program code in general. Students should be able to determine the result of program code that uses conditional statements and nested conditional statements to represent nonlinear processes in a program.

Often, students will write program code that mishandles one of the given conditions. The ability to trace program code can be valuable when testing programs to ensure that all conditions are met. Testing for the different expected behaviors of conditional statements is a critical part of program development and is useful when writing program code or analyzing given code segments. Students should develop sample test cases to illustrate each unique behavior to aid in finding errors and validating results.

UNIT 3

Boolean Expressions and `if` Statements

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~11–13 CLASS PERIODS
CON-1	3.1 Boolean Expressions	2.A Apply the meaning of specific operators.	
CON-2	3.2 `if` Statements and Control Flow	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.	
CON-2	3.3 `if-else` Statements	3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements. 4.A Use test-cases to find errors or validate results.	
CON-2	3.4 `else if` Statements	3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements. 4.C Determine if two or more code segments yield equivalent results.	
CON-1, CON-2	3.5 Compound Boolean Expressions	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.	
CON-1	3.6 Equivalent Boolean Expressions	4.C Determine if two or more code segments yield equivalent results.	
CON-1	3.7 Comparing Objects	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 3.A Write program code to create objects of a class and call methods.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 3. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	3.3	**Code tracing** Provide students with several code segments that contain conditional statements. Have students trace various sample inputs, keeping track of the statements that get executed and the order in which they are executed. This can help students find errors and validate results.
2	3.2–3.5	**Pair programming** Have students work with a partner to create a "guess checker" that could be used as part of a larger game. Students compare four given digits to a preexisting four-digit code that is stored in individual variables. Their program should provide output containing the number of correct digits in correct locations, as well as the number of correct digits in incorrect locations. This program can be continually improved as students learn about nested conditional statements and compound Boolean expressions.
3	3.5	**Diagramming** Have students create truth tables by listing all the possible true and false combinations and corresponding Boolean values for a given compound Boolean expression. Once students have created the truth table, provide students with input values. Have students determine the value of each individual Boolean expression and use the truth table to determine the result of the compound Boolean expression.
4	3.6	**Student response system** Provide students with a code segment that utilizes conditional statements and a compound Boolean expression, and ask them to choose an equivalent code segment that uses a nested conditional statement (and vice versa). Have them report their responses using a student response system.
5	3.7	**Predict and compare** Have students predict the output of several different code segments that compare object references—some that use `.equals()` and some that use ==. Once done, have them create a program that contains those code segments and compare the actual and expected results. This is best illustrated using a simple class that you write yourself.

UNIT 3

Boolean Expressions and `if` Statements

SUGGESTED SKILL

2.A

Apply the meaning of specific operators.

AVAILABLE RESOURCES

- Practice-It!: BJP4 Chapter 4: Conditional Execution—Self-Check 4.2

TOPIC 3.1
Boolean Expressions

Required Course Content

ENDURING UNDERSTANDING

CON-1

The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.E

Evaluate Boolean expressions that use relational operators in program code.

ESSENTIAL KNOWLEDGE

CON-1.E.1

Primitive values and reference values can be compared using relational operators (i.e., == and !=).

CON-1.E.2

Arithmetic expression values can be compared using relational operators (i.e., <, >, <=, >=).

CON-1.E.3

An expression involving relational operators evaluates to a Boolean value.

TOPIC 3.2
`if` Statements and Control Flow

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.A

Represent branching logical processes by using conditional statements.

ESSENTIAL KNOWLEDGE

CON-2.A.1

Conditional statements interrupt the sequential execution of statements.

CON-2.A.2

`if` statements affect the flow of control by executing different statements based on the value of a Boolean expression.

CON-2.A.3

A one-way selection (`if` statement) is written when there is a set of statements to execute under a certain condition. In this case, the body is executed only when the Boolean condition is `true`.

SUGGESTED SKILLS

2.B

Determine the result or output based on statement execution order in a code segment without method calls (other than output).

3.C

Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 5.1—Conditionals
- Practice-It! BJP4 Chapter 4: Conditional Execution—Self-Check 4.3; Exercises 4.2 and 4.3
- The Exam > 2017 AP Computer Science A Exam Free-Response Question #1, Part A (Phrase)

UNIT 3

Boolean Expressions and `if` Statements

SUGGESTED SKILLS

3.C
Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

4.A
Use test-cases to find errors or validate results.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 5.1—Conditionals
- Practice-It!: BJP4 Chapter 4: Conditional Execution—Self-Check 4.5–4.6, 4.10–4.12

TOPIC 3.3
`if-else` Statements

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.A
Represent branching logical processes by using conditional statements.

ESSENTIAL KNOWLEDGE

CON-2.A.4
A two-way selection is written when there are two sets of statements— one to be executed when the Boolean condition is `true`, and another set for when the Boolean condition is `false`. In this case, the body of the "if" is executed when the Boolean condition is `true`, and the body of the "else" is executed when the Boolean condition is `false`.

TOPIC 3.4
`else if` Statements

SUGGESTED SKILLS

3.C
Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

4.C
Determine if two or more code segments yield equivalent results.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 5.2—Three or More Options

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.A
Represent branching logical processes by using conditional statements.

ESSENTIAL KNOWLEDGE

CON-2.A.5
A multi-way selection is written when there are a series of conditions with different statements for each condition. Multi-way selection is performed using `if-else-if` statements such that exactly one section of code is executed based on the first condition that evaluates to true.

UNIT 3

Boolean Expressions and `if` Statements

SUGGESTED SKILLS

2.B
Determine the result or output based on statement execution order in a code segment without method calls (other than output).

3.C
Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 5.3—Complex Conditionals
- Practice-It!: BJP4 Chapter 4: Conditional Execution—Exercise 4.12

TOPIC 3.5
Compound Boolean Expressions

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.B
Represent branching logical processes by using nested conditional statements.

ESSENTIAL KNOWLEDGE

CON-2.B.1
Nested `if` statements consist of `if` statements within `if` statements.

ENDURING UNDERSTANDING

CON-1
The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.F
Evaluate compound Boolean expressions in program code.

ESSENTIAL KNOWLEDGE

CON-1.F.1
Logical operators `!` (not), `&&` (and), and `||` (or) are used with Boolean values. This represents the order these operators will be evaluated.

CON-1.F.2
An expression involving logical operators evaluates to a Boolean value.

CON-1.F.3
When the result of a logical expression using `&&` or `||` can be determined by evaluating only the first Boolean operand, the second is not evaluated. This is known as short-circuited evaluation.

TOPIC 3.6
Equivalent Boolean Expressions

SUGGESTED SKILL

4.C

Determine if two or more code segments yield equivalent results.

AVAILABLE RESOURCE

- Runestone Academy: AP CSA—Java Review: 5.5—De Morgan's Laws

Required Course Content

ENDURING UNDERSTANDING

CON-1

The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.G

Compare and contrast equivalent Boolean expressions.

ESSENTIAL KNOWLEDGE

CON-1.G.1

De Morgan's Laws can be applied to Boolean expressions.

CON-1.G.2

Truth tables can be used to prove Boolean identities.

CON-1.G.3

Equivalent Boolean expressions will evaluate to the same value in all cases.

UNIT 3

Boolean Expressions and `if` Statements

SUGGESTED SKILLS

2.C
Determine the result or output based on the statement execution order in a code segment containing method calls.

3.A
Write program code to create objects of a class and call methods.

TOPIC 3.7
Comparing Objects

Required Course Content

ENDURING UNDERSTANDING

CON-1
The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.H
Compare object references using Boolean expressions in program code.

ESSENTIAL KNOWLEDGE

CON-1.H.1
Two object references are considered aliases when they both reference the same object.

CON-1.H.2
Object reference values can be compared, using == and `!=`, to identify aliases.

CON-1.H.3
A reference value can be compared with `null`, using == or `!=`, to determine if the reference actually references an object.

CON-1.H.4
Often classes have their own `equals` method, which can be used to determine whether two objects of the class are equivalent.

AP COMPUTER SCIENCE A

UNIT 4

Iteration

17.5–22.5%
AP EXAM WEIGHTING

~14–16
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 4

Multiple-choice: ~15 questions
Free-response: 2 questions

- Methods and Control Structures
- Methods and Control Structures: partial

UNIT 4 | 17.5–22.5% AP EXAM WEIGHTING | ~14–16 CLASS PERIODS

Iteration

BIG IDEA 1
Control CON

- How does iteration improve programs and reduce the amount of program code necessary to complete a task?
- What situations would warrant the use of one type of loop over another?

Developing Understanding

This unit focuses on iteration using `while` and `for` loops. As you saw in Unit 3, Boolean expressions are useful when a program needs to perform different operations under different conditions. Boolean expressions are also one of the main components in iteration. This unit introduces several standard algorithms that use iteration. Knowledge of standard algorithms makes solving similar problems easier, as algorithms can be modified or combined to suit new situations.

Iteration is used when traversing data structures such as arrays, ArrayLists, and 2D arrays. In addition, it is a necessary component of several standard algorithms, including searching and sorting, which will be covered in later units.

Building Computational Thinking Practices

2.B 2.D 3.C 5.C

Students should be able to determine the result of program code that uses iterative statements to represent nonlinear processes in a program. Students should practice determining the number of times a given loop structure will execute. Spending time analyzing existing program code provides opportunities for students to better understand how iterative structures can be set up and used to solve their own problems, as well as the implications associated with code changes, such as how using a different iterative structure might change the result of a set of program code.

Students should be able to implement program code that uses iterative statements to represent nonlinear processes. Understanding how to write program code that repeats allows students to write programs to solve a wider variety of problems, including those that use data, which will be covered in Units 6, 7, and 8.

Preparing for the AP Exam

While the concept of iteration is tested in isolation on the multiple-choice exam, it is a foundational concept that students will use along with other topics, such as data structures, on free-response questions. Often, students struggle in situations that warrant variation in the Boolean condition of loops, such as when they want to terminate a loop early. Early termination of a loop requires the use of conditional statements within the body of the loop. If the order of the program statements is incorrect, the early termination may be triggered too early or not at all. Provide students with practice ordering statements by giving them strips of paper, each with a line of program code that can be used to assemble the correct and incorrect solutions. Ask them to reassemble the code and trace it to see if it is correct. Using manipulatives in this way makes it easier for students to rearrange the order of the program code to determine if it is in the correct order.

UNIT 4

Iteration

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~14–16 CLASS PERIODS
CON-2	4.1 `while` Loops	1.B Determine code that would be used to complete code segments. 2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.	
	4.2 `for` Loops	3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements. 4.C Determine if two or more code segments yield equivalent results. 5.C Explain how the result of program code changes, given a change to the initial code.	
	4.3 Developing Algorithms Using Strings	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.	
	4.4 Nested Iteration	1.B Determine code that would be used to complete code segments. 3.C Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements. 5.C Explain how the result of program code changes, given a change to the initial code.	
	4.5 Informal Code Analysis	2.D Determine the number of times a code segment will execute.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 4. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	4.2	**Jigsaw** As a whole class, look at a code segment containing iteration and the resulting output. Afterward, divide students into groups, and provide each group with a slightly modified code segment. After groups have determined how the result changes based on their modified segment, have them get together with students who investigated a different version of the code segment and share their conclusions.
2	4.1–4.4	**Note-taking** Provide students with a method that, when given an integer, returns the month name from a `String` that includes all the month names in order, each separated by a space. Have them annotate what each statement does in the method. Then, ask students to use their annotated method as a guide to write a similar method that, given a student number as input, returns the name of a student from a `String` containing the first name of all students in the class, each separated by a space.
3	4.5	**Simplify the problem** Provide students with several code segments containing iteration. For each segment, have students trace through the execution of a loop with smaller bounds to see what boundary cases are considered, and then use that information to determine the number of times each loop executes with the original bounds.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate writing and analyzing program code.

{ } After completing this unit, students will have covered all of the necessary content for the Consumer Review Lab. The proposed class periods for this unit include time to complete the provided lab activities.

UNIT 4

Iteration

SUGGESTED SKILLS

1.B

Determine code that would be used to complete code segments.

2.B

Determine the result or output based on statement execution order in a code segment without method calls (other than output).

3.C

Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 7.2—While Loops
- Practice-It!: BJP4 Chapter 5: Program Logic and Indefinite Loops—Exercises 5.1–5.4
- The Exam > 2017 AP Computer Science A Exam Free-Response Question #3, Part B (Phrase)

TOPIC 4.1
`while` Loops

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.C

Represent iterative processes using a `while` loop.

ESSENTIAL KNOWLEDGE

CON-2.C.1

Iteration statements change the flow of control by repeating a set of statements zero or more times until a condition is met.

CON-2.C.2

In loops, the Boolean expression is evaluated before each iteration of the loop body, including the first. When the expression evaluates to `true`, the loop body is executed. This continues until the expression evaluates to `false`, whereupon the iteration ceases.

CON-2.C.3

A loop is an infinite loop when the Boolean expression always evaluates to `true`.

CON-2.C.4

If the Boolean expression evaluates to `false` initially, the loop body is not executed at all.

CON-2.C.5

Executing a return statement inside an iteration statement will halt the loop and exit the method or constructor.

continued on next page

Iteration

UNIT 4

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON-2.D For algorithms in the context of a particular specification that does not require the use of traversals: a. Identify standard algorithms. b. Modify standard algorithms. c. Develop an algorithm.	CON-2.D.1 There are standard algorithms to: ▪ Identify if an integer is or is not evenly divisible by another integer ▪ Identify the individual digits in an integer ▪ Determine the frequency with which a specific criterion is met CON-2.D.2 There are standard algorithms to: ▪ Determine a minimum or maximum value ▪ Compute a sum, average, or mode

UNIT 4

Iteration

SUGGESTED SKILLS

3.C

Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

4.C

Determine if two or more code segments yield equivalent results.

5.C

Explain how the result of program code changes, given a change to the initial code.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 7.3—For Loops
- Practice-It!: BJP4 Chapter 2: Primitive Data and Definite Loops—Exercises 2.2, 2.3

TOPIC 4.2
`for` Loops

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.E

Represent iterative processes using a `for` loop.

ESSENTIAL KNOWLEDGE

CON-2.E.1

There are three parts in a `for` loop header: the initialization, the Boolean expression, and the increment. The increment statement can also be a decrement statement.

CON-2.E.2

In a `for` loop, the initialization statement is only executed once before the first Boolean expression evaluation. The variable being initialized is referred to as a loop control variable.

CON-2.E.3

In each iteration of a `for` loop, the increment statement is executed after the entire loop body is executed and before the Boolean expression is evaluated again.

CON-2.E.4

A `for` loop can be rewritten into an equivalent `while` loop and vice versa.

CON-2.E.5

"Off by one" errors occur when the iteration statement loops one time too many or one time too few.

TOPIC 4.3
Developing Algorithms Using Strings

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.F

For algorithms in the context of a particular specification that involves `String` objects:

a. Identify standard algorithms.

b. Modify standard algorithms.

c. Develop an algorithm.

ESSENTIAL KNOWLEDGE

CON-2.F.1

There are standard algorithms that utilize `String` traversals to:

- Find if one or more substrings has a particular property
- Determine the number of substrings that meet specific criteria
- Create a new string with the characters reversed

SUGGESTED SKILLS

2.C

Determine the result or output based on the statement execution order in a code segment containing method calls.

3.C

Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

AVAILABLE RESOURCES

- Practice-It!: BJP4 Chapter 3: Parameters and Objects—Exercise 3.19
- Practice-It!: BJP4 Chapter 5: Program Logic and Indefinite Loops—Exercise 5.24
- CodingBat Java: String-2

UNIT 4

Iteration

SUGGESTED SKILLS

1.B

Determine code that would be used to complete code segments.

3.C

Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

5.C

Explain how the result of program code changes, given a change to the initial code.

AVAILABLE LAB

- Classroom Resources > AP Computer Science A: Consumer Review Lab

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 7.4—Nested For Loops
- Practice-It!: BJP4 Chapter 2: Primitive Data and Definite Loops—Exercises 2.4–2.15
- Past AP Free-Response Exam Questions on Methods and Control Structures on AP Question Bank

TOPIC 4.4
Nested Iteration

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.G

Represent nested iterative processes.

ESSENTIAL KNOWLEDGE

CON-2.G.1

Nested iteration statements are iteration statements that appear in the body of another iteration statement.

CON-2.G.2

When a loop is nested inside another loop, the inner loop must complete all its iterations before the outer loop can continue.

Iteration

UNIT 4

TOPIC 4.5
Informal Code Analysis

SUGGESTED SKILL
2.D
Determine the number of times a code segment will execute.

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.H
Compute statement execution counts and informal run-time comparison of iterative statements.

ESSENTIAL KNOWLEDGE

CON-2.H.1
A statement execution count indicates the number of times a statement is executed by the program.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

UNIT 5

Writing Classes

5–7.5%
AP EXAM WEIGHTING

~12–14
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 5

Multiple-choice: ~25 questions
Free-response: 2 questions

- Class
- Class: partial

UNIT 5 | 5–7.5% AP EXAM WEIGHTING | ~12–14 CLASS PERIODS

Writing Classes

BIG IDEA 1
Modularity MOD
- How can using a model of traffic patterns improve travel time?

BIG IDEA 2
Variables VAR
- How can programs be written to protect your bank account balance from being inadvertently changed?

BIG IDEA 3
Impact of Computing IOC
- What responsibility do programmers have for the consequences of programs they create, whether intentional or not?

Developing Understanding

This unit will pull together information from all previous units to create new, user-defined reference data types in the form of classes. The ability to accurately model real-world entities in a computer program is a large part of what makes computer science so powerful. This unit focuses on identifying appropriate behaviors and attributes of real-world entities and organizing these into classes. Students will build on what they learn in this unit to represent relationships between classes through hierarchies, which appear in Unit 9.

The creation of computer programs can have extensive impacts on societies, economies, and cultures. The legal and ethical concerns that come with programs and the responsibilities of programmers are also addressed in this unit.

Building Computational Thinking Practices

1.A 3.B 5.A 5.B 5.D

Computer scientists use computers to study and model the world, and this requires designing program code to fit a given scenario. Spending time to plan out the program up front will shorten overall program development time. Practicing elements of the iterative development process early and often will help create more robust programs and avoid logical errors.

Because revisions to program code are rarely completed by the original author, documenting program code is very important. It allows programmers to more quickly understand how a code segment functions without having to analyze the program code itself. Well-written documentation includes a description of the behavior, as well as the initial conditions that must be met for a code segment to work as intended.

Sometimes students struggle to explain why a code segment will not compile or work as intended. It helps to have students work with a collaborative partner to practice verbally explaining the issues. The ability to explain why a code segment isn't working correctly assists in the development of solutions.

Preparing for the AP Exam

On the free-response section of the exam, students are required to design program code that demonstrates the functionality available for an object of a new class, based on the prompt's specification. This process includes identifying the attributes (data) that define an object of a class and the behaviors (methods) that define what an object of a class can do. Students should have many opportunities in this course to design their own classes based on program specifications and on observations of real-world objects.

Once the new class is designed, students should be able to implement program code to create a new type by creating a class. The behaviors of an object are expressed through writing *methods* in the class, which include expressions, conditional statements, and iterative statements. By being able to create their own classes, programmers are not limited to the existing classes provided within the Java libraries and can therefore represent their own ideas through classes.

UNIT 5

Writing Classes

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~12–14 CLASS PERIODS
MOD-2 MOD-3	5.1 Anatomy of a Class	1.A Determine an appropriate program design to solve a problem or accomplish a task (*not assessed*). 1.B Determine code that would be used to complete code segments.	
MOD-2	5.2 Constructors	1.C Determine code that would be used to interact with completed program code. 3.B Write program code to define a new type by creating a class.	
	5.3 Documentation with Comments	5.D Describe the initial conditions that must be met for a program segment to work as intended or described.	
	5.4 Accessor Methods	3.B Write program code to define a new type by creating a class. 5.B Explain why a code segment will not compile or work as intended.	
	5.5 Mutator Methods	3.B Write program code to define a new type by creating a class. 4.B Identify errors in program code.	
	5.6 Writing Methods	1.B Determine code that would be used to complete code segments. 3.B Write program code to define a new type by creating a class.	
	5.7 Static Variables and Methods	3.B Write program code to define a new type by creating a class. 5.A Describe the behavior of a given segment of program code.	

continued on next page

UNIT AT A GLANCE *(cont'd)*

Enduring Understanding	Topic	Suggested Skills	Class Periods ~12–14 CLASS PERIODS
VAR-1	**5.8 Scope and Access**	3.B Write program code to define a new type by creating a class. 5.B Explain why a code segment will not compile or work as intended.	
	5.9 `this` Keyword	2.C Determine the result or output based on the statement execution order in a code segment containing method calls.	
IOC-1	**5.10 Ethical and Social Implications of Computing Systems**	Curricular requirement, not assessed on the AP Exam	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 5. Review the results in class to identify and address any student misunderstandings.		

UNIT 5 Writing Classes

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	5.1	**Kinesthetic learning** Have students break into groups of 4–5 to play board games. Ask them to play the game for about 10 minutes. While they play the game, they should keep track of the various nouns they encounter and actions that happen as part of the game. The nouns can be represented in the computer as classes, and the actions are the behaviors. At the end of game play, ask students to create UML diagrams for the identified classes.
2	5.3	**Marking the text** Present students with specifications, and have them highlight or underline any preconditions (both implicit and explicit) that exist for the method to function. This includes information about parameters, such as object references not being `null`.
3	5.4–5.7	**Create a plan** When asked to write a method, have students write an outline using pseudocode with paper and pencil. Then, go through it step-by-step with sample input to ensure that the process is correct and to determine if any additional information is needed before beginning to program a solution on the computer.
4	5.7	**Paraphrase** Provide students with several example classes that utilize static variables for unique identification numbers or for counting the number of objects that have been created, but do not provide any description or documentation for the code. Have students spend time creating objects and calling the static methods to investigate how the static variables behave, then have them document the code appropriately to describe how each class utilizes static variables and methods.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate writing and analyzing program code.

TOPIC 5.1
Anatomy of a Class

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE

MOD-2.A

Designate access and visibility constraints to classes, data, constructors, and methods.

ESSENTIAL KNOWLEDGE

MOD-2.A.1

The keywords `public` and `private` affect the access of classes, data, constructors, and methods.

MOD-2.A.2

The keyword `private` restricts access to the declaring class, while the keyword `public` allows access from classes outside the declaring class.

MOD-2.A.3

Classes are designated `public`.

MOD-2.A.4

Access to attributes should be kept internal to the class. Therefore, instance variables are designated as `private`.

MOD-2.A.5

Constructors are designated `public`.

MOD-2.A.6

Access to behaviors can be internal or external to the class. Therefore, methods can be designated as either `public` or `private`.

SUGGESTED SKILL

1.A

Determine an appropriate program design to solve a problem or accomplish a task.

1.B

Determine code that would be used to complete code segments.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 2.7—Parts of a Java Class
- Practice-It!: BJP4 Chapter 8: Classes—Self-Check 8.1–8.7
- Classroom Resources > Object-Oriented Design

UNIT 5

Writing Classes

ENDURING UNDERSTANDING

MOD-3

When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.A

Designate private visibility of instance variables to encapsulate the attributes of an object.

ESSENTIAL KNOWLEDGE

MOD-3.A.1

Data encapsulation is a technique in which the implementation details of a class are kept hidden from the user.

MOD-3.A.2

When designing a class, programmers make decisions about what data to make accessible and modifiable from an external class. Data can be either accessible or modifiable, or it can be both or neither.

MOD-3.A.3

Instance variables are encapsulated by using the `private` access modifier.

MOD-3.A.4

The provided accessor and mutator methods in a class allow client code to use and modify data.

TOPIC 5.2
Constructors

SUGGESTED SKILLS

1.C

Determine code that would be used to interact with completed program code.

3.B

Write program code to define a new type by creating a class.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 2.7—Parts of a Java Class
- Practice-It!: BJP4 Chapter 8: Classes—Exercises 8.14–8.22

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE

MOD-2.B

Define instance variables for the attributes to be initialized through the constructors of a class.

ESSENTIAL KNOWLEDGE

MOD-2.B.1

An object's state refers to its attributes and their values at a given time and is defined by instance variables belonging to the object. This creates a "has-a" relationship between the object and its instance variables.

MOD-2.B.2

Constructors are used to set the initial state of an object, which should include initial values for all instance variables.

MOD-2.B.3

Constructor parameters are local variables to the constructor and provide data to initialize instance variables.

MOD-2.B.4

When a mutable object is a constructor parameter, the instance variable should be initialized with a copy of the referenced object. In this way, the instance variable is not an alias of the original object, and methods are prevented from modifying the state of the original object.

MOD-2.B.5

When no constructor is written, Java provides a no-argument constructor, and the instance variables are set to default values.

UNIT 5

Writing Classes

SUGGESTED SKILL

5.D

Describe the initial conditions that must be met for a program segment to work as intended or described.

AVAILABLE RESOURCE

- External Resource > Zeroturnaround: Reasons, Tips and Tricks for Better Java Documentation

TOPIC 5.3
Documentation with Comments

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE

MOD-2.C

Describe the functionality and use of program code through comments.

ESSENTIAL KNOWLEDGE

MOD-2.C.1

Comments are ignored by the compiler and are not executed when the program is run.

MOD-2.C.2

Three types of comments in Java include `/* */`, which generates a block of comments, `//`, which generates a comment on one line, and `/** */`, which are Javadoc comments and are used to create API documentation.

MOD-2.C.3

A precondition is a condition that must be true just prior to the execution of a section of program code in order for the method to behave as expected. There is no expectation that the method will check to ensure preconditions are satisfied.

MOD-2.C.4

A postcondition is a condition that must always be true after the execution of a section of program code. Postconditions describe the outcome of the execution in terms of what is being returned or the state of an object.

MOD-2.C.5

Programmers write method code to satisfy the postconditions when preconditions are met.

TOPIC 5.4
Accessor Methods

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE

MOD-2.D

Define behaviors of an object through non-void methods without parameters written in a class.

ESSENTIAL KNOWLEDGE

MOD-2.D.1

An accessor method allows other objects to obtain the value of instance variables or static variables.

MOD-2.D.2

A non-void method returns a single value. Its header includes the return type in place of the keyword `void`.

MOD-2.D.3

In non-void methods, a return expression compatible with the return type is evaluated, and a copy of that value is returned. This is referred to as "return by value."

MOD-2.D.4

When the return expression is a reference to an object, a copy of that reference is returned, not a copy of the object.

MOD-2.D.5

The `return` keyword is used to return the flow of control to the point immediately following where the method or constructor was called.

continued on next page

SUGGESTED SKILLS

3.B

Write program code to define a new type by creating a class.

5.B

Explain why a code segment will not compile or work as intended.

AVAILABLE LABS

- Classroom Resources >
  - AP Computer Science A: Data Lab
  - AP Computer Science A: Celebrity Lab

AVAILABLE RESOURCE

- Practice-It!: BJP4 Chapter 8: Classes—Exercises 8.14–8.22

UNIT 5

Writing Classes

LEARNING OBJECTIVE

MOD-2.D

Define behaviors of an object through non-void methods without parameters written in a class.

ESSENTIAL KNOWLEDGE

MOD-2.D.6

The `toString` method is an overridden method that is included in classes to provide a description of a specific object. It generally includes what values are stored in the instance data of the object.

MOD-2.D.7

If `System.out.print` or `System.out.println` is passed an object, that object's `toString` method is called, and the returned string is printed.

Writing Classes

UNIT 5

TOPIC 5.5
Mutator Methods

SUGGESTED SKILLS

3.B

Write program code to define a new type by creating a class.

4.B

Identify errors in program code.

AVAILABLE RESOURCE

- Practice-It!: BJP4 Chapter 8: Classes—Exercises 8.14–8.22

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE

MOD-2.E

Define behaviors of an object through void methods with or without parameters written in a class.

ESSENTIAL KNOWLEDGE

MOD-2.E.1

A void method does not return a value. Its header contains the keyword `void` before the method name.

MOD-2.E.2

A mutator (modifier) method is often a void method that changes the values of instance variables or static variables.

UNIT 5

Writing Classes

SUGGESTED SKILLS

1.B

Determine code that would be used to complete code segments.

3.B

Write program code to define a new type by creating a class.

AVAILABLE RESOURCES

- Practice-It!: BJP4 Chapter 3: Parameters and Objects—Exercises 3.1–3.22
- Practice-It!: BJP4 Chapter 8: Class—Exercises 8.14–8.22

TOPIC 5.6
Writing Methods

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE

MOD-2.F

Define behaviors of an object through non-void methods with parameters written in a class.

ESSENTIAL KNOWLEDGE

MOD-2.F.1

Methods can only access the private data and methods of a parameter that is a reference to an object when the parameter is the same type as the method's enclosing class.

MOD-2.F.2

Non-void methods with parameters receive values through parameters, use those values, and return a computed value of the specified type.

MOD-2.F.3

It is good programming practice to not modify mutable objects that are passed as parameters unless required in the specification.

MOD-2.F.4

When an actual parameter is a primitive value, the formal parameter is initialized with a copy of that value. Changes to the formal parameter have no effect on the corresponding actual parameter.

continued on next page

Writing Classes

UNIT 5

LEARNING OBJECTIVE

MOD-2.F

Define behaviors of an object through non-void methods with parameters written in a class.

ESSENTIAL KNOWLEDGE

MOD-2.F.5

When an actual parameter is a reference to an object, the formal parameter is initialized with a copy of that reference, not a copy of the object. If the reference is to a mutable object, the method or constructor can use this reference to alter the state of the object.

MOD-2.F.6

Passing a reference parameter results in the formal parameter and the actual parameter being aliases. They both refer to the same object.

UNIT 5

Writing Classes

SUGGESTED SKILLS

3.B

Write program code to define a new type by creating a class.

5.A

Describe the behavior of a given segment of program code.

TOPIC 5.7
Static Variables and Methods

Required Course Content

ENDURING UNDERSTANDING

MOD-2

Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
MOD-2.G Define behaviors of a class through static methods.	MOD-2.G.1 Static methods are associated with the class, not objects of the class.
	MOD-2.G.2 Static methods include the keyword `static` in the header before the method name.
	MOD-2.G.3 Static methods cannot access or change the values of instance variables.
	MOD-2.G.4 Static methods can access or change the values of static variables.
	MOD-2.G.5 Static methods do not have a `this` reference and are unable to use the class's instance variables or call non-static methods.
MOD-2.H Define the static variables that belong to the class.	MOD-2.H.1 Static variables belong to the class, with all objects of a class sharing a single static variable.

continued on next page

Writing Classes

UNIT 5

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
MOD-2.H Define the static variables that belong to the class.	**MOD-2.H.2** Static variables can be designated as either `public` or `private` and are designated with the `static` keyword before the variable type. **MOD-2.H.3** Static variables are used with the class name and the dot operator, since they are associated with a class, not objects of a class.

UNIT 5

Writing Classes

SUGGESTED SKILLS

5.B

Explain why a code segment will not compile or work as intended.

3.B

Write program code to define a new type by creating a class.

TOPIC 5.8
Scope and Access

Required Course Content

ENDURING UNDERSTANDING

VAR-1

To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.G

Explain where variables can be used in the program code.

ESSENTIAL KNOWLEDGE

VAR-1.G.1

Local variables can be declared in the body of constructors and methods. These variables may only be used within the constructor or method and cannot be declared to be `public` or `private`.

VAR-1.G.2

When there is a local variable with the same name as an instance variable, the variable name will refer to the local variable instead of the instance variable.

VAR-1.G.3

Formal parameters and variables declared in a method or constructor can only be used within that method or constructor.

VAR-1.G.4

Through method decomposition, a programmer breaks down a large problem into smaller subproblems by creating methods to solve each individual subproblem.

Writing Classes

UNIT 5

TOPIC 5.9
this Keyword

SUGGESTED SKILL
2.C
Determine the result or output based on the statement execution order in a code segment containing method calls.

AVAILABLE RESOURCE
- Past AP Free-Response Exam Questions on Class on AP Question Bank

Required Course Content

ENDURING UNDERSTANDING

VAR-1
To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.H
Evaluate object reference expressions that use the keyword `this`.

ESSENTIAL KNOWLEDGE

VAR-1.H.1
Within a non-static method or a constructor, the keyword `this` is a reference to the current object—the object whose method or constructor is being called.

VAR-1.H.2
The keyword `this` can be used to pass the current object as an actual parameter in a method call.

UNIT 5

Writing Classes

TOPIC 5.10
Ethical and Social Implications of Computing Systems

AVAILABLE RESOURCE

- Classroom Resources > Ethical Use of the Computer

Required Course Content

ENDURING UNDERSTANDING

IOC-1

While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.A

Explain the ethical and social implications of computing systems.

ESSENTIAL KNOWLEDGE

IOC-1.A.1

System reliability is limited. Programmers should make an effort to maximize system reliability.

IOC-1.A.2

Legal issues and intellectual property concerns arise when creating programs.

IOC-1.A.3

The creation of programs has impacts on society, economies, and culture. These impacts can be beneficial and/or harmful.

AP COMPUTER SCIENCE A

UNIT 6

Array

10–15%
AP EXAM WEIGHTING

~6–8
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 6

Multiple-choice: ~15 questions
Free-response: 2 questions

- Array and `ArrayList` (Array only)
- Array and `ArrayList` (Array only): partial

UNIT 6 | 10–15% AP EXAM WEIGHTING | ~6–8 CLASS PERIODS

Array

BIG IDEA 1
Variables VAR
- How can programs leverage volcano data to make predictions about the date of the next eruption?

BIG IDEA 2
Control CON
- How can knowing standard algorithms be useful when solving new problems?

Developing Understanding

This unit focuses on data structures, which are used to represent collections of related data using a single variable rather than multiple variables. Using a data structure along with iterative statements with appropriate bounds will allow for similar treatment to be applied more easily to all values in the collection. Just as there are useful standard algorithms when dealing with primitive data, there are standard algorithms to use with data structures. In this unit, we apply standard algorithms to arrays; however, these same algorithms are used with ArrayLists and 2D arrays as well. Additional standard algorithms, such as standard searching and sorting algorithms, will be covered in the next unit.

Building Computational Thinking Practices

3.D 4.B

Students should be able to implement program code to create, traverse, and manipulate elements in a 1D array. Traversing elements of a 1D array can be accomplished in multiple ways. Programmers need to make decisions about which loop structure is most effective given the problem they are trying to solve. Some loop structures, such as the enhanced `for` loop, only allow programmers to examine the data stored in a 1D array structure, while other loop structures allow the data to be manipulated.

Students should be able to identify and correct errors related to traversing and manipulating 1D array structures. A common run-time error that programmers experience is an out-of-bounds error, which occurs when the program tries to access an element that is beyond the range of elements in a collection. Students should double-check the values of the index being used on at least the initial and final loop iterations to ensure that they aren't out of bounds.

Preparing for the AP Exam

A specific iterative structure is commonly used to traverse an array: starting at the beginning and moving toward the last element in the array. When students are asked to determine the result of program code that contains arrays, they often use this same iterative structure. Knowing this iterative structure can be helpful when students use tracing to determine what an algorithm is doing. Students can follow this traversal for the first few iterations and apply that pattern to the remaining elements in the array.

When preparing for the free-response questions, students should be come familiar with how existing algorithms work, rather than just memorizing the program code. Have students write out an algorithm on paper and test it using manipulatives. This allows students to experience the algorithm on a deeper level than if they simply program it. A strong understanding of how existing algorithms work allows programmers to make modifications to those algorithms to accomplish similar tasks.

UNIT 6

Array

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~6–8 CLASS PERIODS
VAR-2	**6.1 Array Creation and Access**	1.C Determine code that would be used to interact with completed program code. 3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.	
VAR-2	**6.2 Traversing Arrays**	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects. 4.B Identify errors in program code.	
VAR-2	**6.3 Enhanced `for` Loop for Arrays**	3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects. 4.C Determine if two or more code segments yield equivalent results.	
CON-2	**6.4 Developing Algorithms Using Arrays**	1.B Determine code that would be used to complete code segments. 3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects. 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 6. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	6.1	**Diagramming** Provide students with several prompts to create and access elements in an array. After they have determined the code for each prompt, have students draw a memory diagram that shows references and the arrays they point to. Have students update the diagram with each statement to demonstrate how changing the contents through one array reference effects all the array references for this array.
2	6.2	**Error analysis** Provide students with several error-ridden code segments containing array traversals along with the expected output of each segment. Ask them to identify any errors that they see on paper and to suggest fixes to provide the expected output. Have them type up their solutions in an IDE to verify their work.
3	6.3	**Think-pair-share** Ask students to consider two program code segments that are meant to yield the same result: one using a traditional `for` loop and one using a `for each` loop. Have them take a few minutes to think independently about whether the two segments accomplish the same result and, if not, what changes could be made in order for that to happen. Then, ask students to work with their partners to come up with situations where it would make sense to use one type of loop over the other before sharing with the whole class.
4	6.4	**Pair programming** Have students use pair programming to solve an array-based free-response question. Have one student be the driver for Part A while the other navigates, then have them switch for Part B. Once they are done, have partners switch solutions with another group and work through the scoring guidelines to "grade" that solution. Spend time as a class discussing the different approaches students used.

UNIT 6

Array

SUGGESTED SKILLS

1.C
Determine code that would be used to interact with completed program code.

3.D
Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 8.1—Arrays in Java
- Practice-It!: BJP4 Chapter 7: Array—Self-Check 7.1–7.9
- CodingBat Java: Array-1

TOPIC 6.1
Array Creation and Access

Required Course Content

ENDURING UNDERSTANDING

VAR-2
To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.A
Represent collections of related primitive or object reference data using one-dimensional (1D) array objects.

ESSENTIAL KNOWLEDGE

VAR-2.A.1
The use of array objects allows multiple related items to be represented using a single variable.

VAR-2.A.2
The size of an array is established at the time of creation and cannot be changed.

VAR-2.A.3
Arrays can store either primitive data or object reference data.

VAR-2.A.4
When an array is created using the keyword `new`, all of its elements are initialized with a specific value based on the type of elements:

- Elements of type `int` are initialized to `0`
- Elements of type `double` are initialized to `0.0`
- Elements of type `boolean` are initialized to `false`
- Elements of a reference type are initialized to the reference value `null`. No objects are automatically created

VAR-2.A.5
Initializer lists can be used to create and initialize arrays.

continued on next page

Array

UNIT 6

LEARNING OBJECTIVE

VAR-2.A

Represent collections of related primitive or object reference data using one-dimensional (1D) array objects.

ESSENTIAL KNOWLEDGE

VAR-2.A.6

Square brackets ([ ]) are used to access and modify an element in a 1D array using an index.

VAR-2.A.7

The valid index values for an array are 0 through one less than the number of elements in the array, inclusive. Using an index value outside of this range will result in an `ArrayIndexOutOfBoundsException` being thrown.

UNIT 6

Array

SUGGESTED SKILLS

2.B
Determine the result or output based on statement execution order in a code segment without method calls (other than output).

3.D
Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

4.B
Identify errors in program code.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 8.3—Using a For Loop to Loop through an Array
- CodingBat Java: Array-2
- Practice-It!: BJP4 Chapter 7: Arrays—Exercise 7.1–7.18

TOPIC 6.2
Traversing Arrays

Required Course Content

ENDURING UNDERSTANDING

VAR-2
To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.B
Traverse the elements in a 1D array.

ESSENTIAL KNOWLEDGE

VAR-2.B.1
Iteration statements can be used to access all the elements in an array. This is called traversing the array.

VAR-2.B.2
Traversing an array with an indexed `for` loop or `while` loop requires elements to be accessed using their indices.

VAR-2.B.3
Since the indices for an array start at 0 and end at the number of elements – 1, "off by one" errors are easy to make when traversing an array, resulting in an `ArrayIndexOutOfBoundsException` being thrown.

Array

UNIT 6

TOPIC 6.3
Enhanced for Loop for Arrays

Required Course Content

ENDURING UNDERSTANDING

VAR-2

To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.C

Traverse the elements in a 1D array object using an enhanced `for` loop.

ESSENTIAL KNOWLEDGE

VAR-2.C.1

An enhanced `for` loop header includes a variable, referred to as the enhanced `for` loop variable.

VAR-2.C.2

For each iteration of the enhanced `for` loop, the enhanced `for` loop variable is assigned a copy of an element without using its index.

VAR-2.C.3

Assigning a new value to the enhanced `for` loop variable does not change the value stored in the array.

VAR-2.C.4

Program code written using an enhanced `for` loop to traverse and access elements in an array can be rewritten using an indexed `for` loop or a `while` loop.

SUGGESTED SKILLS

3.D

Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

4.C

Determine if two or more code segments yield equivalent results.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 8.2—Looping with the For-Each Loop
- Practice-It!: BJP4 Chapter 7: Arrays—Exercises 7.1–7.18

UNIT 6

Array

SUGGESTED SKILLS

1.B
Determine code that would be used to complete code segments.

3.D
Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

5.D
Describe the initial conditions that must be met for a program segment to work as intended or described.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 8.13—Free-Response Questions
- CodingBat Java: Array 3
- Practice-It!: BJP4 Chapter 7: Array—Exercises 7.1–7.18

TOPIC 6.4
Developing Algorithms Using Arrays

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.I
For algorithms in the context of a particular specification that requires the use of array traversals:

a. Identify standard algorithms.

b. Modify standard algorithms.

c. Develop an algorithm.

ESSENTIAL KNOWLEDGE

CON-2.I.1
There are standard algorithms that utilize array traversals to:

- Determine a minimum or maximum value
- Compute a sum, average, or mode
- Determine if at least one element has a particular property
- Determine if all elements have a particular property
- Access all consecutive pairs of elements
- Determine the presence or absence of duplicate elements
- Determine the number of elements meeting specific criteria

CON-2.I.2
There are standard array algorithms that utilize traversals to:

- Shift or rotate elements left or right
- Reverse the order of the elements

AP COMPUTER SCIENCE A

UNIT 7

ArrayList

2.5–7.5%
AP EXAM WEIGHTING

~10–12
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 7

Multiple-choice: ~15 questions
Free-response: 1 question

- Array and `ArrayList` (`ArrayList` focus)

UNIT 7 | 2.5–7.5% AP EXAM WEIGHTING | ~10–12 CLASS PERIODS

ArrayList

BIG IDEA 1
Variables VAR
- Why is an `ArrayList` more appropriate for storing your music playlist, while an array might be more appropriate for storing your class schedule?

BIG IDEA 2
Control CON
- How can we use statement execution counts to choose appropriate algorithms?

BIG IDEA 3
Impact of Computing IOC
- What personal data is currently being collected, and how?

Developing Understanding

As students learned in Unit 6, data structures are helpful when storing multiple related data values. Arrays have a static size, which causes limitations related to the number of elements stored, and it can be challenging to reorder elements stored in arrays. The `ArrayList` object has a dynamic size, and the class contains methods for insertion and deletion of elements, making reordering and shifting items easier. Deciding which data structure to select becomes increasingly important as the size of the data set grows, such as when using a large real-world data set.

In this unit, students will also learn about privacy concerns related to storing large amounts of personal data and about what can happen if such information is compromised.

Building Computational Thinking Practices

2.C 2.D 3.D 5.C

Students need to consider the impact using `ArrayList` rather than an array has on the structure of their program code. This includes considering the use of `ArrayList` methods and the flexibility of a structure with a dynamic size. For instance, the use of an `ArrayList` will require students to analyze program code that uses method calls.

Providing students with practice writing programs for data sets of undetermined sized—or at least larger than they would be able to analyze easily by hand—presents a more relevant and realistic experience with data. Additionally, this requires students to focus more on the algorithm and ensuring that it will work in all situations rather than on an individual result.

With larger data sets, programmers become concerned with the amount of time it will take for their program code to run. Students should have practice determining the number of times a code segment executes; this can help them gain an idea of how long it will take to run a program on a data set of a given size.

Preparing for the AP Exam

When writing solutions to free-response questions that involve the use of an `ArrayList`, students are often asked to insert or delete elements from the `ArrayList`. In these cases, adjustments will need to be made to the loop counter to account for skipping an element or attempting to access elements that no longer exist.

Students may also be asked to traverse multiple data structures simultaneously. These data structures may be a mixture of array and `ArrayList` objects. As such, it is very easy for students to become confused about how elements are accessed and manipulated within each structure. Additionally, the size of the data structures may not be the same. In these situations, it is best to have different index variables and bounds checks for each structure to avoid accessing elements that are out-of-bounds.

UNIT 7 ArrayList

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~10–12 CLASS PERIODS
VAR-2	7.1 Introduction to `ArrayList`	1.B Determine code that would be used to complete code segments. 3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.	
VAR-2	7.2 `ArrayList` Methods	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.	
VAR-2	7.3 Traversing ArrayLists	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.	
CON-2	7.4 Developing Algorithms Using ArrayLists	3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects. 4.A Use test-cases to find errors or validate results.	
CON-2	7.5 Searching	3.D Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects. 5.C Explain how the result of program code changes, given a change to the initial code.	
CON-2	7.6 Sorting	2.D Determine the number of times a code segment will execute.	
IOC-1	7.7 Ethical Issues Around Data Collection	Curricular requirement, not assessed on the AP Exam	

Go to AP Classroom to assign the **Personal Progress Check** for Unit 7. Review the results in class to identify and address any student misunderstandings.

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	7.2	**Predict and compare** Have students look at the code they wrote to solve the free-response question in Unit 6 (or other code from Unit 6) on paper, and have them rewrite it using an `ArrayList`. Have them highlight the parts that need to be changed and determine how to change them. Then, have students type up the changes in an IDE and confirm that the program still works as expected.
2	7.1–7.5	**Identify a subtask** Have students read through an `ArrayList`-based free-response question in groups, and have them identify all subtasks. These subtasks could be conditional statements, iteration, or even other methods. Once the subtasks have been identified, divide the subtasks among the group members, and have students implement their given subtask. When all students are finished, have them combine the subtasks into a single solution.
3	7.5	**Discussion group** Discuss the algorithm necessary to search for the smallest value in an `ArrayList`. Without explaining what you are doing, change the Boolean expression so that it will find the largest value, and ask students to describe what the resulting algorithm will do. Then, change the algorithm to store and return the location of the largest value, and discuss the change.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate writing and analyzing program code that uses real-world data sets.

UNIT 7

ArrayList

SUGGESTED SKILLS

1.B
Determine code that would be used to complete code segments.

3.D
Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

AVAILABLE RESOURCES

- Java Quick Reference (see Appendix)
- Runestone Academy: AP CSA—Java Review: 9.7—The ArrayList Class
- Practice-It!: BJP4 Chapter 10: ArrayLists—Self-Check 10.2

TOPIC 7.1
Introduction to ArrayList

Required Course Content

ENDURING UNDERSTANDING

VAR-2
To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.D
Represent collections of related object reference data using `ArrayList` objects.

ESSENTIAL KNOWLEDGE

VAR-2.D.1
An `ArrayList` object is mutable and contains object references.

VAR-2.D.2
The `ArrayList` constructor `ArrayList()` constructs an empty list.

VAR-2.D.3
Java allows the generic type `ArrayList<E>`, where the generic type `E` specifies the type of the elements.

VAR-2.D.4
When `ArrayList<E>` is specified, the types of the reference parameters and return type when using the methods are type `E`.

VAR-2.D.5
`ArrayList<E>` is preferred over `ArrayList` because it allows the compiler to find errors that would otherwise be found at run-time.

TOPIC 7.2
ArrayList Methods

Required Course Content

ENDURING UNDERSTANDING

VAR-2

To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.D

Represent collections of related object reference data using `ArrayList` objects.

ESSENTIAL KNOWLEDGE

VAR-2.D.6

The `ArrayList` class is part of the `java.util` package. An import statement can be used to make this class available for use in the program.

VAR-2.D.7

The following `ArrayList` methods—including what they do and when they are used—are part of the Java Quick Reference:

- `int size()` – Returns the number of elements in the list
- `boolean add(E obj)` – Appends `obj` to end of list; returns `true`
- `void add(int index, E obj)` – Inserts `obj` at position `index` (`0 <= index <= size`), moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
- `E get(int index)` – Returns the element at position `index` in the list
- `E set(int index, E obj)` — Replaces the element at position `index` with `obj`; returns the element formerly at position `index`

continued on next page

SUGGESTED SKILLS

2.C

Determine the result or output based on the statement execution order in a code segment containing method calls.

3.D

Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

AVAILABLE RESOURCES

- Java Quick Reference (see Appendix)
- Practice-It! BJP4 Chapter 10: ArrayLists—Exercises 10.2–10.17
- The Exam > 2017 AP Computer Science A Exam Free-Response Question #1, Part A (Digits)

UNIT 7

ArrayList

LEARNING OBJECTIVE

VAR-2.D

Represent collections of related object reference data using `ArrayList` objects.

ESSENTIAL KNOWLEDGE

- `E remove(int index)` — Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`

TOPIC 7.3
Traversing ArrayLists

SUGGESTED SKILLS

2.C
Determine the result or output based on the statement execution order in a code segment containing method calls.

3.D
Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 9.14—Looping through a List
- Practice-It!: BJP4 Chapter 10: ArrayLists—Exercises 10.2–10.17
- The Exam > 2018 AP Computer Science A Exam Free-Response Question #2 (WordPair)

Required Course Content

ENDURING UNDERSTANDING

VAR-2

To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.E

For `ArrayList` objects:

a. Traverse using a `for` or `while` loop

b. Traverse using an enhanced `for` loop

ESSENTIAL KNOWLEDGE

VAR-2.E.1

Iteration statements can be used to access all the elements in an `ArrayList`. This is called traversing the `ArrayList`.

VAR-2.E.2

Deleting elements during a traversal of an `ArrayList` requires using special techniques to avoid skipping elements.

VAR-2.E.3

Since the indices for an `ArrayList` start at 0 and end at the number of elements – 1, accessing an index value outside of this range will result in an `IndexOutOfBoundsException` being thrown.

VAR-2.E.4

Changing the size of an `ArrayList` while traversing it using an enhanced `for` loop can result in a `ConcurrentModificationException` being thrown. Therefore, when using an enhanced `for` loop to traverse an `ArrayList`, you should not add or remove elements.

UNIT 7

ArrayList

SUGGESTED SKILLS

3.D
Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

4.A
Use test-cases to find errors or validate results.

AVAILABLE LAB

- Classroom Resources > AP Computer Science A: Data Lab

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 9.13—Removing an Object at an Index
- Practice-It!: BJP4 Chapter 10: ArrayLists—Exercises 10.2–10.17
- The Exam >
  - 2017 AP Computer Science A Exam Free-Response Question 1, Part B (Digits)
  - Past AP Free-Response Exam Questions on Array/ `ArrayList` on AP Question Bank

TOPIC 7.4

Developing Algorithms Using ArrayLists

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.J
For algorithms in the context of a particular specification that requires the use of `ArrayList` traversals:

a. Identify standard algorithms.

b. Modify standard algorithms.

c. Develop an algorithm.

ESSENTIAL KNOWLEDGE

CON-2.J.1
There are standard `ArrayList` algorithms that utilize traversals to:

- Insert elements
- Delete elements
- Apply the same standard algorithms that are used with 1D arrays

CON-2.J.2
Some algorithms require multiple `String`, array, or `ArrayList` objects to be traversed simultaneously.

TOPIC 7.5
Searching

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.K

Apply sequential/linear search algorithms to search for specific information in array or `ArrayList` objects.

ESSENTIAL KNOWLEDGE

CON-2.K.1

There are standard algorithms for searching.

CON-2.K.2

Sequential/linear search algorithms check each element in order until the desired value is found or all elements in the array or `ArrayList` have been checked.

SUGGESTED SKILLS

3.D

Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.

5.C

Explain how the result of program code changes, given a change to the initial code.

AVAILABLE RESOURCES

- Practice-It!: BJP4 Chapter 10: ArrayLists—Exercises 10.2–10.17
- Runestone Academy: AP CSA—Java Review: 13—Searching and Sorting
- Practice-It!: BJP4 Chapter 13: Searching and Sorting

UNIT 7

ArrayList

SUGGESTED SKILL

2.D

Determine the number of times a code segment will execute.

AVAILABLE LAB

- Classroom Resources > AP Computer Science A: Data Lab

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 13—Searching and Sorting
- Practice-It!: BJP4 Chapter 13: Searching and Sorting—Self-Check 13.29 and 13.30
- Visualgo.net: Sorting
- Sorting.at

TOPIC 7.6
Sorting

Required Course Content

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON-2.L Apply selection sort and insertion sort algorithms to sort the elements of array or `ArrayList` objects.	CON-2.L.1 Selection sort and insertion sort are iterative sorting algorithms that can be used to sort elements in an array or `ArrayList`.
CON-2.M Compute statement execution counts and informal run-time comparison of sorting algorithms.	CON-2.M.1 Informal run-time comparisons of program code segments can be made using statement execution counts.

TOPIC 7.7
Ethical Issues Around Data Collection

Required Course Content

ENDURING UNDERSTANDING

IOC-1

While programs are typically designed to achieve a specific purpose, they may have unintended consequences.

LEARNING OBJECTIVE

IOC-1.B

Explain the risks to privacy from collecting and storing personal data on computer systems.

ESSENTIAL KNOWLEDGE

IOC-1.B.1

When using the computer, personal privacy is at risk. Programmers should attempt to safeguard personal privacy.

IOC-1.B.2

Computer use and the creation of programs have an impact on personal security. These impacts can be beneficial and/or harmful.

AVAILABLE RESOURCES

- Classroom Resources >
  - Ethical Use of the Computer
  - Ethical Issues: Internet Content Providers and Internet Service Providers

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

UNIT 8

2D Array

7.5–10%
AP EXAM WEIGHTING

~10–12
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 8

Multiple-choice: ~10 questions
Free-response: 1 question

- 2D Array

UNIT 8 | 7.5–10% AP EXAM WEIGHTING | ~10–12 CLASS PERIODS

2D Array

BIG IDEA 1
Variables VAR
- Why might you want to use a 2D array to store the spaces on a game board or the pixels in a picture, rather than a 1D array or `ArrayList`?

BIG IDEA 2
Control CON
- Why does the order in which elements are accessed in 2D array traversal matter in some situations?

Developing Understanding

In Unit 6, students learned how 1D arrays store large amounts of related data. These same concepts will be implemented with two-dimensional (2D) arrays in this unit. A 2D array is most suitable to represent a table. Each table element is accessed using the variable name and row and column indices. Unlike 1D arrays, 2D arrays require nested iterative statements to traverse and access all elements. The easiest way to accomplished this is in row-major order, but it is important to cover additional traversal patterns, such as back and forth or column-major.

Building Computational Thinking Practices

1.B 2.B 2.D 3.E

Students should be able to determine the result of program code that traverses and manipulates the elements in a 2D array. Traversals of 2D arrays typically require a set of nested loops. Often the extra dimension of a 2D array is difficult for students to envision. Providing students with practice analyzing and tracing traversals of a 2D array, as well as providing them partial program code to complete, helps students take this more abstract concept and make it concrete and replicable.

Because 2D arrays are traversed using nested loops, the number of times a code segment executes is multiplied. In a nested loop, the inner loop must complete all iterations before the outer loop can continue. It helps to provide students with sample code that will print the values in a 2D array. Teachers can an IDE that shows access to 2D arrays visually and keeps track of the execution count.

Preparing for the AP Exam

The free-response portion of the exam always includes one question that requires students to write program code involving 2D arrays. Because 2D arrays are arrays where each element is an array, it is not uncommon for the question to require students to write solutions involving array or `ArrayList` objects as well.

While there is a specific nested structure to traverse elements in a 2D array in row-major order, this structure can be modified to traverse 2D arrays in other ways, such as column-major, by switching the nested iterative statements. Additional modifications can be made to traverse rows or columns in different ways, such as back and forth or up and down. However, when making these adjustments, students often neglect to adjust the bounds of the iterative statements appropriately. Students should practice traversing 2D arrays in these nonstandard ways, being sure to test the boundary conditions of the iterative statements, to be prepared for this type of free-response question.

UNIT 8

2D Array

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods
			~10–12 CLASS PERIODS
VAR-2	8.1 2D Arrays	1.B Determine code that would be used to complete code segments. 1.C Determine code that would be used to interact with completed program code. 3.E Write program code to create, traverse, and manipulate elements in 2D array objects.	
VAR-2, CON-2	8.2 Traversing 2D Arrays	2.B Determine the result or output based on statement execution order in a code segment without method calls (other than output). 2.D Determine the number of times a code segment will execute. 3.E Write program code to create, traverse, and manipulate elements in 2D array objects. 4.A Use test-cases to find errors or validate results.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 8. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	8.1	**Using manipulatives** Use different-sized egg cartons or ice cube trays with random compartments filled with small toys or candy. Create laminated cards with the code for the construction of, and access to, a 2D array, leaving blanks for the name and size dimensions. Have students fill in the missing code that would be used to represent the physical 2D array objects and access the randomly stored elements.
2	8.2	**Activating prior knowledge** When first introducing 2D arrays and row-major traversal, ask students which part of the nested `for` loop structure loops over a 1D array. Based on what they know about the traversal of 1D array structures, ask them to calculate the number of times the inner loop executes.
3	8.2	**Sharing and responding** As a class, create a set of test cases to be used with answers to a free-response question. Have students write their answers to the free-response question individually on paper. After exchanging solutions with another student, ask students to find errors or validate results of their peers' code by tracing the code with the developed test cases. Allow students an opportunity to provide feedback on the program code as well as the results of each test case.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate writing and analyzing program code that represents 2D data sets.

{ } After completing this unit, students will have covered all of the necessary content for the Steganography Lab. The proposed class periods for this unit include time to complete the provided lab activities.

UNIT 8

2D Array

SUGGESTED SKILLS

1.B

Determine code that would be used to complete code segments.

1.C

Determine code that would be used to interact with completed program code.

3.E

Write program code to create, traverse, and manipulate elements in 2D array objects.

AVAILABLE LABS

- Classroom Resources >
  - AP Computer Science A: Picture Lab
  - AP Computer Science A: Steganography Lab

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 10—Two-dimensional Arrays
- Practice-It!: BJP4 Chapter 7: Arrays—Self-Check 7.31–7.35
- Classroom Resources > GridWorld Resources: A Curriculum Module for AP Computer Science

TOPIC 8.1
2D Arrays

Required Course Content

ENDURING UNDERSTANDING

VAR-2

To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.F

Represent collections of related primitive or object reference data using two-dimensional (2D) array objects.

ESSENTIAL KNOWLEDGE

VAR-2.F.1

2D arrays are stored as arrays of arrays. Therefore, the way 2D arrays are created and indexed is similar to 1D array objects.

EXCLUSION STATEMENT—(EK VAR-2.F.1): 2D array objects that are not rectangular are outside the scope of the course and AP Exam.

VAR-2.F.2

For the purposes of the exam, when accessing the element at `arr[first][second]`, the first index is used for rows, the second index is used for columns.

VAR-2.F.3

The initializer list used to create and initialize a 2D array consists of initializer lists that represent 1D arrays.

VAR-2.F.4

The square brackets `[row][col]` are used to access and modify an element in a 2D array.

VAR-2.F.5

"Row-major order" refers to an ordering of 2D array elements where traversal occurs across each row, while "column-major order" traversal occurs down each column.

TOPIC 8.2
Traversing 2D Arrays

Required Course Content

ENDURING UNDERSTANDING

VAR-2

To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.

LEARNING OBJECTIVE

VAR-2.G

For 2D array objects:

a. Traverse using nested `for` loops.

b. Traverse using nested enhanced `for` loops.

ESSENTIAL KNOWLEDGE

VAR-2.G.1

Nested iteration statements are used to traverse and access all elements in a 2D array. Since 2D arrays are stored as arrays of arrays, the way 2D arrays are traversed using `for` loops and enhanced `for` loops is similar to 1D array objects.

VAR-2.G.2

Nested iteration statements can be written to traverse the 2D array in "row-major order" or "column-major order."

VAR-2.G.3

The outer loop of a nested enhanced `for` loop used to traverse a 2D array traverses the rows. Therefore, the enhanced `for` loop variable must be the type of each row, which is a 1D array. The inner loop traverses a single row. Therefore, the inner enhanced `for` loop variable must be the same type as the elements stored in the 1D array.

SUGGESTED SKILLS

2.B

Determine the result or output based on statement execution order in a code segment without method calls (other than output).

2.D

Determine the number of times a code segment will execute.

3.E

Write program code to create, traverse, and manipulate elements in 2D array objects.

4.A

Use test-cases to find errors or validate results.

AVAILABLE LABS

- Classroom Resources >
  - AP Computer Science A: Picture Lab
  - AP Computer Science A: Steganography Lab

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 10.7—Looping through a 2D Array
- Practice-It!: BJP4 Chapter 7: Arrays—Exercises 7.19–7.19
- The Exam >
  - 2017 AP Computer Science A Exam Free-Response Question #4 (Position)
  - 2018 AP Computer Science A Exam Free-Response Question #4 (ArrayTester)
  - Past AP Exam Questions on 2D Arrays on AP Question Bank

UNIT 8

2D Array

ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.N

For algorithms in the context of a particular specification that requires the use of 2D array traversals:

a. Identify standard algorithms.

b. Modify standard algorithms.

c. Develop an algorithm.

ESSENTIAL KNOWLEDGE

CON-2.N.1

When applying sequential/linear search algorithms to 2D arrays, each row must be accessed then sequential/linear search applied to each row of a 2D array.

CON-2.N.2

All standard 1D array algorithms can be applied to 2D array objects.

AP COMPUTER SCIENCE A

UNIT 9

Inheritance

5–10%
AP EXAM WEIGHTING

~13–15
CLASS PERIODS

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 9

Multiple-choice: ~15 questions
Free-response: 2 questions

- Class
- Class: partial

UNIT 9 | 5–10% AP EXAM WEIGHTING | ~13–15 CLASS PERIODS

Inheritance

BIG IDEA 1
Modularity MOD

- How might the use of inheritance help in writing a program that simulates crops being grown in a virtual world?
- How does inheritance make programs more versatile?

Developing Understanding

Creating objects, calling methods on the objects created, and being able to define a new data type by creating a class are essential understandings before moving into this unit. One of the strongest advantages of Java is the ability to categorize classes into hierarchies through *inheritance*. Certain existing classes can be extended to include new behaviors and attributes without altering existing code. These newly created classes are called *subclasses*. In this unit, students will learn how to recognize common attributes and behaviors that can be used in a *superclass* and will then create a hierarchy by writing subclasses to extend a superclass. Recognizing and utilizing existing hierarchies will help students create more readable and maintainable programs.

Building Computational Thinking Practices

1.A 1.C 3.A 3.B 5.A 5.B 5.D

Students can design hierarchies by listing the attributes and behaviors for each object and pulling common elements into a superclass, leaving unique attributes and behaviors in the subclass. By creating a hierarchy system, students only need to write common program code one time, reducing potential errors and implementation time. This also allows for changes to be made more easily, because they can be made at the superclass level in the hierarchy and automatically apply to subclasses.

During the development of a program, programmers often use comments to describe the behavior of a given segment of program code and to describe the initial conditions that are used. Students who develop the skill of explaining why program code does not work, such as methods being overloaded improperly or superclass objects attempting to call subclass methods, are much better equipped to foresee and avoid these hierarchy issues.

Preparing for the AP Exam

One question on the free-response section of the exam will require students to write a class. This class could be part of an inheritance relationship. When overriding superclass methods in a subclass, method signatures must be the same. This includes the number, type, and order of any parameters of the overridden method. It is important for students to recognize when a method should be overridden, as well as when they can and should use methods from the superclass. Students who duplicate code unnecessarily will not earn full points on this question.

Students will be asked to analyze program code that uses inheritance. In many cases, students struggle with determining whether a method is available to be called by an object of a class. When a method is called on a subclass object, the method that is executed is determined during run-time. If the subclass does not contain the called method, the superclass method will automatically be executed.

UNIT 9

Inheritance

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods ~13–15 CLASS PERIODS
MOD-3	9.1 Creating Superclasses and Subclasses	1.A Determine an appropriate program design to solve a problem or accomplish a task (*not assessed*). 3.B Write program code to define a new type by creating a class.	
	9.2 Writing Constructors for Subclasses	3.B Write program code to define a new type by creating a class. 5.A Describe the behavior of a given segment of program code.	
	9.3 Overriding Methods	3.B Write program code to define a new type by creating a class. 5.D Describe the initial conditions that must be met for a program segment to work as intended or described.	
	9.4 `super` Keyword	1.C Determine code that would be used to interact with completed program code. 3.B Write program code to define a new type by creating a class.	
	9.5 Creating References Using Inheritance Hierarchies	3.A Write program code to create objects of a class and call methods. 5.B Explain why a code segment will not compile or work as intended.	
	9.6 Polymorphism	3.A Write program code to create objects of a class and call methods. 5.B Explain why a code segment will not compile or work as intended.	
	9.7 `Object` Superclass	1.C Determine code that would be used to interact with completed program code. 3.B Write program code to define a new type by creating a class.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 9. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	9.1	**Activating prior knowledge** Have students review what they know about classes, methods, and the scope of variables by having them write a class based on specifications that can easily be extended by subclasses. This class will become the superclass for subclasses they write later in the unit.
2	9.2–9.4	**Create a plan** Given a class design problem that requires the use of multiple classes in an inheritance hierarchy, students identify the common attributes and behaviors among these classes and write these into a superclass. Any additional information that does not belong in the superclass will be categorized to determine the additional classes that might be necessary and what methods will need to be added or overridden in the subclasses.
3	9.4	**Think aloud** Provide students with a code segment that contains method calls using the `super` keyword. Have students describe the code segment out loud to themselves. Give students several individual statements that attempt to interact with the given code segment, and have them talk through each one, describing which statements would work and which ones would not, as well as the reasons why those statements wouldn't work.
4	9.5–9.6	**Student response system** Provide students with several statements where objects are created and the reference type and object type are different but related. Then provide students with calls to methods on these created objects. Use a student response system to have students determine whether each statement is legal, would result in a compile-time error, or would result in a run-time error.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate writing and analyzing program code.

After completing this unit, students will have covered all of the necessary content for the Celebrity Lab. The proposed class periods for this unit include time to complete the provided lab activities.

UNIT 9

Inheritance

SUGGESTED SKILLS

1.A
Determine an appropriate program design to solve a problem or accomplish a task.

3.B
Write program code to define a new type by creating a class.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 11.3—Inheritance
- Classroom Resources >
  - Object-Oriented Design
  - An Introduction to Polymorphism in Java

TOPIC 9.1
Creating Superclasses and Subclasses

Required Course Content

ENDURING UNDERSTANDING

MOD-3
When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.B
Create an inheritance relationship from a subclass to the superclass.

ESSENTIAL KNOWLEDGE

MOD-3.B.1
A class hierarchy can be developed by putting common attributes and behaviors of related classes into a single class called a superclass.

MOD-3.B.2
Classes that extend a superclass, called subclasses, can draw upon the existing attributes and behaviors of the superclass without repeating these in the code.

MOD-3.B.3
Extending a subclass from a superclass creates an "is-a" relationship from the subclass to the superclass.

MOD-3.B.4
The keyword `extends` is used to establish an inheritance relationship between a subclass and a superclass. A class can extend only one superclass.

Inheritance

UNIT 9

TOPIC 9.2
Writing Constructors for Subclasses

Required Course Content

ENDURING UNDERSTANDING

MOD-3

When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.B

Create an inheritance relationship from a subclass to the superclass.

ESSENTIAL KNOWLEDGE

MOD-3.B.5

Constructors are not inherited.

MOD-3.B.6

The superclass constructor can be called from the first line of a subclass constructor by using the keyword `super` and passing appropriate parameters.

MOD-3.B.7

The actual parameters passed in the call to the superclass constructor provide values that the constructor can use to initialize the object's instance variables.

MOD-3.B.8

When a subclass's constructor does not explicitly call a superclass's constructor using `super`, Java inserts a call to the superclass's no-argument constructor.

MOD-3.B.9

Regardless of whether the superclass constructor is called implicitly or explicitly, the process of calling superclass constructors continues until the `Object` constructor is called. At this point, all of the constructors within the hierarchy execute beginning with the `Object` constructor.

SUGGESTED SKILLS

3.B
Write program code to define a new type by creating a class.

5.A
Describe the behavior of a given segment of program code.

AVAILABLE RESOURCES

- Practice-It! BJP4 Chapter 9: Inheritance and Interfaces—Self-Check 9.3
- Classroom Resources >
  - An Introduction to Polymorphism in Java
  - Gradebook Project

UNIT 9

Inheritance

SUGGESTED SKILLS

3.B
Write program code to define a new type by creating a class.

5.D
Describe the initial conditions that must be met for a program segment to work as intended or described.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 11.8—Overriding vs Overloading
- Practice-It!: BJP4 Chapter 9: Inheritance and Interfaces—Exercises 9.4, 9.9
- Classroom Resources >
  - An Introduction to Polymorphism in Java
  - Gradebook Project

TOPIC 9.3
Overriding Methods

Required Course Content

ENDURING UNDERSTANDING

MOD-3
When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.B
Create an inheritance relationship from a subclass to the superclass.

ESSENTIAL KNOWLEDGE

MOD-3.B.10
Method overriding occurs when a public method in a subclass has the same method signature as a public method in the superclass.

MOD-3.B.11
Any method that is called must be defined within its own class or its superclass.

MOD-3.B.12
A subclass is usually designed to have modified (overridden) or additional methods or instance variables.

MOD-3.B.13
A subclass will inherit all public methods from the superclass; these methods remain public in the subclass.

Inheritance

UNIT 9

TOPIC 9.4
super Keyword

Required Course Content

ENDURING UNDERSTANDING

MOD-3

When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.B

Create an inheritance relationship from a subclass to the superclass.

ESSENTIAL KNOWLEDGE

MOD-3.B.14

The keyword `super` can be used to call a superclass's constructors and methods.

MOD-3.B.15

The superclass method can be called in a subclass by using the keyword `super` with the method name and passing appropriate parameters.

SUGGESTED SKILLS

1.C

Determine code that would be used to interact with completed program code.

3.B

Write program code to define a new type by creating a class.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 11.9—Using Super to Call an Overridden Method
- Classroom Resources >
  - Gradebook Project
  - Inheritance and Polymorphism with Sudoku

UNIT 9

Inheritance

SUGGESTED SKILLS

3.A
Write program code to create objects of a class and call methods.

5.B
Explain why a code segment will not compile or work as intended.

AVAILABLE RESOURCES

- Practice-It!: BJP4 Chapter 9: Inheritance and Interfaces—Self-Check 9.8, 9.10
- Classroom Resources > Gradebook Project

TOPIC 9.5
Creating References Using Inheritance Hierarchies

Required Course Content

ENDURING UNDERSTANDING

MOD-3
When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.C
Define reference variables of a superclass to be assigned to an object of a subclass in the same hierarchy.

ESSENTIAL KNOWLEDGE

MOD-3.C.1
When a class S "is-a" class T, T is referred to as a superclass, and S is referred to as a subclass.

MOD-3.C.2
If S is a subclass of T, then assigning an object of type S to a reference of type T facilitates polymorphism.

MOD-3.C.3
If S is a subclass of T, then a reference of type T can be used to refer to an object of type T or S.

MOD-3.C.4
Declaring references of type T, when S is a subclass of T, is useful in the following declarations:

- Formal method parameters
- arrays — `T[]` *var* `ArrayList<T>` *var*

Inheritance

UNIT 9

TOPIC 9.6
Polymorphism

Required Course Content

ENDURING UNDERSTANDING

MOD-3

When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.D

Call methods in an inheritance relationship.

ESSENTIAL KNOWLEDGE

MOD-3.D.1

Utilize the `Object` class through inheritance.

MOD-3.D.2

At compile time, methods in or inherited by the declared type determine the correctness of a non-static method call.

MOD-3.D.3

At run-time, the method in the actual object type is executed for a non-static method call.

SUGGESTED SKILLS

3.A

Write program code to create objects of a class and call methods.

5.B

Explain why a code segment will not compile or work as intended.

AVAILABLE LAB

- Classroom Resources > AP Computer Science A: Celebrity Lab

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 11.15—Polymorphism
- Practice-It!: BJP4 Chapter 9: Inheritance and Interfaces—Self-Check 9.9

UNIT 9

Inheritance

SUGGESTED SKILLS

1.C
Determine code that would be used to interact with completed program code.

3.B
Write program code to define a new type by creating a class.

AVAILABLE LAB

- Classroom Resources > AP Computer Science A: Celebrity Lab

AVAILABLE RESOURCE

- Java Quick Reference (see Appendix)

TOPIC 9.7
Object Superclass

Required Course Content

ENDURING UNDERSTANDING

MOD-3
When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.

LEARNING OBJECTIVE

MOD-3.E
Call `Object` class methods through inheritance.

ESSENTIAL KNOWLEDGE

MOD-3.E.1
The `Object` class is the superclass of all other classes in Java.

MOD-3.E.2
The `Object` class is part of the `java.lang` package

MOD-3.E.3
The following `Object` class methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

- `boolean equals(Object other)`
- `String toString()`

MOD-3.E.4
Subclasses of `Object` often override the `equals` and `toString` methods with class-specific implementations.

AP COMPUTER SCIENCE A

UNIT 10

Recursion

5–7.5%
AP EXAM WEIGHTING

~3–5
CLASS PERIODS

AP

Remember to go to AP Classroom to assign students the online **Personal Progress Check** for this unit.

Whether assigned as homework or completed in class, the **Personal Progress Check** provides each student with immediate feedback related to this unit's topics and skills.

Personal Progress Check 10

Multiple-choice: ~10 questions
Free-response: 1 question

- Methods and Control Structures (recursive and non-recursive solutions allowed)

UNIT 10 | 5–7.5% AP EXAM WEIGHTING | ~3–5 CLASS PERIODS

Recursion

BIG IDEA 1
Control CON

- What real-world processes do you follow that are recursive in nature?
- Why do programmers sometimes prefer using recursive solutions when sorting data in a large data set?

Developing Understanding

Sometimes a problem can be solved by solving smaller or simpler versions of the same problem rather than attempting an iterative solution. This is called *recursion*, and it is a powerful math and computer science idea. In this unit, students will revisit how control is passed when methods are called, which is necessary knowledge when working with recursion. Tracing skills introduced in Unit 2 are helpful for determining the purpose or output of a recursive method. In this unit, students will learn how to write simple recursive methods and determine the purpose or output of a recursive method by tracing.

Building Computational Thinking Practices

1.B 2.C 2.D

To better understand how recursion works, students should spend time writing their own recursive methods. Often, this can be overwhelming for students. One way to scaffold this skill for students is to require them to write a portion of program code, such as the base case, that can be used to complete the recursive method.

Students should be able to determine the result of recursive method calls. Tracing the series of calls is a useful way to glean what the recursive method is doing and how it is accomplishing its purpose. Recursive algorithms, such as sorting and searching algorithms, often produce a result much more quickly than iterative solutions. Students also need to understand how many times statements in a recursive solution execute based on given input values.

Preparing for the AP Exam

Recursion is primarily assessed through the multiple-choice section of the exam. Students are often asked to determine the result of a specific call to a recursive method or to describe the behavior of a recursive method. A call to a recursive method is just like a call to a nonrecursive method. Because a method is an abstraction of a process that has a specific result for each varied input, using a specific input value provides an understanding of how the method functions for that input. By understanding several instances of the method call, students can abstract and generalize the method's overall purpose or process.

While students will not be required to write a recursive solution in the free-response section, recursive solutions are often a more straightforward way of writing the solutions than iterative designs. Writing recursive solutions and analyzing calls to recursive methods help engage students with all aspects of recursive methods and provide them with a deeper understanding of how recursion works.

UNIT 10

Recursion

UNIT AT A GLANCE

Enduring Understanding	Topic	Suggested Skills	Class Periods
			~3–5 CLASS PERIODS
CON-2	10.1 Recursion	1.B Determine code that would be used to complete code segments. 5.A Describe the behavior of a given segment of program code.	
	10.2 Recursive Searching and Sorting	2.C Determine the result or output based on the statement execution order in a code segment containing method calls. 2.D Determine the number of times a code segment will execute.	
	Go to AP Classroom to assign the **Personal Progress Check** for Unit 10. Review the results in class to identify and address any student misunderstandings.		

SAMPLE INSTRUCTIONAL ACTIVITIES

The sample activities on this page are optional and are offered to provide possible ways to incorporate instructional approaches into the classroom. They were developed in partnership with teachers from the AP community to share ways that they approach teaching some of the topics in this unit. Please refer to the Instructional Approaches section beginning on p. 159 for more examples of activities and strategies.

Activity	Topic	Sample Activity
1	10.1	**Sharing and responding** Provide students with the pseudocode to multiple recursive algorithms, and have students write the base case of the recursive methods and share it with their partner. The partner should then provide feedback, including any corrections or additions that may be needed.
2	10.1	**Look for a pattern** Provide students with a recursive method and several different inputs. Have students run the recursive method, record the various outputs, and look for a pattern between the input and related output. Ask students to write one or two sentences as a broad description of what the recursive method is doing.
3	10.2	**Code tracing** When looking at a recursive method to determine how many times it executes, have students create a call tree or a stack trace to show the method being called and the values of any parameters of each call. Students can then count up the number of times a statement executes or a method is called.

Unit Planning Notes

Use the space below to plan your approach to the unit. Consider how you want to pace your course and where you will incorporate analysis of recursive program code.

UNIT 10

Recursion

SUGGESTED SKILLS

1.B
Determine code that would be used to complete code segments.

5.A
Describe the behavior of a given segment of program code.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 12—Recursion
- Practice-It!: BJP4 Chapter 12: Recursion—Self-Check 12.3–12.6, 12.13–12.15
- CodingBat Java: Recursion

TOPIC 10.1
Recursion

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.0
Determine the result of executing recursive methods.

ESSENTIAL KNOWLEDGE

CON-2.0.1
A recursive method is a method that calls itself.

CON-2.0.2
Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call.

CON-2.0.3
Each recursive call has its own set of local variables, including the formal parameters.

CON-2.0.4
Parameter values capture the progress of a recursive process, much like loop control variable values capture the progress of a loop.

CON-2.0.5
Any recursive solution can be replicated through the use of an iterative approach.

EXCLUSION STATEMENT—(EK CON-2.0.5): Writing recursive program code is outside the scope of the course and AP Exam.

CON-2.0.6
Recursion can be used to traverse `String`, array, and `ArrayList` objects.

TOPIC 10.2
Recursive Searching and Sorting

Required Course Content

ENDURING UNDERSTANDING

CON-2
Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON-2.P Apply recursive search algorithms to information in `String`, 1D array, or `ArrayList` objects.	**CON-2.P.1** Data must be in sorted order to use the binary search algorithm.
	CON-2.P.2 The binary search algorithm starts at the middle of a sorted array or `ArrayList` and eliminates half of the array or `ArrayList` in each iteration until the desired value is found or all elements have been eliminated.
	CON-2.P.3 Binary search can be more efficient than sequential/linear search.
	X EXCLUSION STATEMENT—(EK CON-2.P.3): Search algorithms other than sequential/linear and binary search are outside the scope of the course and AP Exam.
	CON-2.P.4 The binary search algorithm can be written either iteratively or recursively.
CON-2.Q Apply recursive algorithms to sort elements of array or `ArrayList` objects.	**CON-2.Q.1** Merge sort is a recursive sorting algorithm that can be used to sort elements in an array or `ArrayList`.

SUGGESTED SKILLS

2.C
Determine the result or output based on the statement execution order in a code segment containing method calls.

2.D
Determine the number of times a code segment will execute.

AVAILABLE RESOURCES

- Runestone Academy: AP CSA—Java Review: 13.3—Binary Search
- Runestone Academy: AP CSA—Java Review: 13.6—Merge Sort
- Practice-It!: BJP4 Chapter 12: Recursion—Exercises 12.1–12.3, 12.6–12.14, 12.18–12.22
- Practice-It!: BJP4 Chapter 13: Searching and Sorting—Self-Check 12.30

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

Instructional Approaches


elloWorld.ja
*/
public class He
{
pu
ain

Selecting and Using Course Materials

Language

AP Computer Science A is taught using the Java programming language. The Java language is extensive, with far more features than could be covered in a single introductory course. Although students are allowed to use any valid Java statements on the AP Exam, it is recommended that students use Java 7 and stay within the scope of the Java Quick Reference (see page 219). However, given the technical constraints of some schools and classrooms, a minimum of Java 6 is acceptable. Features of Java 8, including Lambda expressions and the Stream API, are not covered on the AP Exam and should not be covered in this course.

Textbooks

The AP Computer Science A course requires the use of a college-level textbook that includes discussion of all the required course content. These college-level textbooks often cover more material than what is required in AP Computer Science A. While this is not a problem and can provide opportunities for extended learning for students with a strong background and a desire to go beyond the requirements, teachers should recognize what is beyond the scope of the course and not feel compelled to teach the entire contents of the textbook. Because this course does not use features of Java 8 and beyond, it is not necessary to have the newest textbook, but it is best that it include information and features of Java 7 (though a minimum of Java 6 is acceptable).

An example textbook list of college-level textbooks that meet the AP Course Audit resource requirements is provided on AP Central.

Choosing an Integrated Development Environment (IDE)

An integrated development environment, or IDE, is extremely helpful when teaching students how to program. The table that follows shows several free IDEs that teachers of introductory computer science courses have found useful, but the list is not comprehensive. It is important to consider the needs of the students when choosing an IDE. It helps to choose a single IDE that students will use throughout the year. In addition to considering the needs of their students, there are several questions that teachers should ask themselves before choosing an IDE. How difficult or time consuming will it be to learn a specific IDE? What training is available to them, either through online tutorials, workshop consultants, colleagues, or other means? What features do they feel are essential versus nice to have? What features would get in the way of student learning? An IDE should support the teacher and the way they teach, as well as the student learning that takes place in their classroom.

If an IDE supports autocompletion, we recommend turning off this feature or using an IDE without this feature. Students will have to handwrite their responses to free-response questions, and students who rely heavily on autocompletion during the year may struggle writing code without the IDE.

Online Programming Resources

There are several online resources that allow students to write code to meet a specification and receive real-time evaluation. These resources are a great way for students to get immediate feedback on code that they write, without requiring extensive amounts of time from the teacher. Since the answers for most of these questions are freely available online, they are best used as formative assessments to gauge student understanding.

IDE	Platform	Link
BlueJ	Client based	bluej.org/
Cloud9	Browser based	aws.amazon.com/cloud9/
Dr. Java	Client based	drjava.org/
Eclipse	Client based	eclipse.org/downloads/
Greenfoot	Client based	greenfoot.org/door
JGrasp	Client based	jgrasp.org/
IntelliJ	Client based	jetbrains.com/idea/
NetBeans	Client based	netbeans.org/
Repl.it	Browser based	repl.it/

Alternate Sequencing

One of the biggest considerations when teaching object-oriented programming is when to introduce the creation of a new type by designing a class, which is different from using existing classes to create objects. There are two main options: creating new classes before conditional and iterative statements have been covered, or creating new classes after most other content has been covered. The provided unit guide covers the creation of new classes in Unit 5. If teachers are using a curriculum or textbook that encourages teaching the creation of classes early, it is possible to cover Unit 5 directly after Unit 1 or 2. Once the creation of classes has been covered, all future content should be taught in the context of classes to provide students with more opportunities for practice.

Unit 10, on recursion, could also be taught at any point after Unit 3. Recursion can sometimes be easier for students to understand if they are exposed to it before iteration.

Course Pacing

Another consideration when teaching the course is how much time to spend covering each topic. AP Computer Science A is meant to be an introduction to computer science and has no programming prerequisite while still being a full and comparable introductory college-level course. Each unit has a suggested number of class periods to spend on the topics within that unit. This is based on 45-minute class periods, meeting five days per week to account for 140 instructional days.

To help students achieve success, teachers may decide it is necessary to spend more time on the foundational topics in Units 1–4, which are the building blocks for the rest of the course. Students who have a strong understanding of these foundational units are likely to be more successful, because they can focus more on learning new material without continuing to struggle with foundational material. However, more time spent on these fundamental topics will mean less time available to cover some of the remaining topics. Considering the weighting of the remaining units can help determine which topics may be acceptable to cover in less depth if teachers are running short on time.

Instructional Strategies

The AP Computer Science A course framework details the concepts and skills students must master to be successful on the AP Exam. To address those concepts and skills effectively, it helps to incorporate a variety of instructional approaches and best practices into daily lessons and activities.

The following table presents strategies that can help students develop mastery of the skills by engaging them in learning activities that apply their understanding of course concepts.

Programming and Problem-Solving

Strategy	Definition	Purpose	Example
Code tracing	Students step through program code by hand to determine how a piece of program code operates.	Prepares students to determine the output of program code, as well as detect and correct errors in code, when a computer is not available.	Provide students with code segments containing multiple method calls, and have them trace program execution with various input values, noting which lines get evaluated and the order in which they are executed.
Create a plan	Students analyze the tasks in a problem and create a process for completing the tasks by finding the information needed, interpreting data, choosing how to solve a problem, communicating results, and verifying accuracy.	Assists in breaking tasks into smaller parts and identifying the steps needed to complete the entire task. One example of this is the different phases of an iterative development process and how they work together in the completion of a program.	Given a class-design problem that asks students to model real-world phenomena using a class, students identify the components needed to create such a class. This involves selecting appropriate attributes and behaviors, including the data type necessary for each attribute, the information necessary for a given behavior, and what value might be returned.
Error analysis	Students analyze an existing solution to determine whether (or where) errors have occurred.	Allows students to troubleshoot common errors that may arise when they create similar methods and focus on correcting these errors.	When students begin to write methods, they can analyze the method output and troubleshoot any errors that might lead to incorrect output or run-time errors. By becoming more familiar with the types of errors that may happen, students should be better able to spot them in unfamiliar code.

continued on next page

Programming and Problem-Solving (cont'd)

Strategy	Definition	Purpose	Example
Identify a subtask	Students break a problem into smaller pieces whose outcomes lead to a solution.	Helps organize the pieces of a complex problem and reach a complete solution.	When providing students with a free-response question or other program specification, have them break down the problem into subtasks. These smaller tasks could end up being conditional statements, loops, or even other methods.
Look for a pattern	Students observe trends by looking at expected output or results based on input and specifications.	Helps identify patterns that can be used to design program code, generalize program behavior, or identify errors in existing program code.	Patterns can be detected when trying to describe what a code segment does when several different inputs are provided. These patterns can allow the output of the code to be generalized into a broader description.
Marking the text	Students highlight, underline, and/or annotate text to focus on key information to help understand the text or solve the problem.	Helps students identify important information in the text and make notes about the interpretation of tasks required and concepts to apply to reach a solution.	When completing a free-response question, students should mark parameters to ensure that they know to use them in their solution and, if the method returns a value, what the type is to ensure that they are returning the correct type of value.
Pair programming	Two programmers work together as a pair. One (the driver) writes program code, while the other (the observer, pointer, or navigator) reviews each line of program code as it is typed in.	Reinforces the need for students to explain their process in a way that another student can understand. It also provides built-in support as students are learning new material.	When students practice writing code to fulfill a specification—especially early in the year when covering the fundamentals of selection and iteration—it is helpful for them to work in teams. By following the pair programming paradigm, students gain experience both writing and reviewing code and can work together to solve problems when they arise, without waiting for help from the teacher.

continued on next page

Programming and Problem-Solving (cont'd)

Strategy	Definition	Purpose	Example
Predict and compare	Students make conjectures about what results will develop in an activity and confirm or modify the conjectures based on outcomes.	Stimulates thinking by making, checking, and correcting predictions based on evidence from the outcome.	Given a set of methods and input/output pairs, students attempt to match the methods with the appropriate input/output pairs. Students then run the methods using the given inputs to confirm their original match selection.
Simplify the problem	Students use simpler numbers or statements to solve a problem.	Provides insight into the problem or the strategies needed to solve the problem.	When introducing recursion, larger problems are discussed in their simplest form for students to understand the "base case" before moving on to the more general case.
Think aloud	Students talk through a difficult problem by describing what the text means.	Explaining a difficulty aloud engages students with the problem in a new way that puts them in the role of thinking of potential solutions. This trains students to consider solutions for themselves prior to seeking assistance from a teacher or peer.	Give students a rubber duck or other inanimate object. By describing the problem aloud to the object, students are better able to work out the solution for themselves.

Cooperative Learning

Strategy	Definition	Purpose	Example
Ask the expert	Students are assigned as "experts" on problems they have mastered; groups rotate through the expert stations to learn about problems they have not yet mastered.	Provides opportunities for students to share their knowledge and learn from one another.	When learning different methods in the `String` class, the teacher assigns students as "experts" on the different methods within the Java Quick Reference (see Appendix). There is a station for each method, and students rotate through stations in groups, working with the station expert to complete a series of problems using the corresponding method.
Discussion group	Students engage in an interactive, small-group discussion.	Allows students to gain new understanding of or insight into a problem or solution by listening to multiple perspectives.	Have students read an article or articles on a social or ethical implication of computer science in the news and then discuss, in small groups, why these are issues and how they might have been avoided.
Jigsaw	Each student in a group reads a different text or passage, taking on the role of "expert" on what was read. Students share the information from that reading with students from other groups and then return to their original groups to share their new knowledge.	Students summarize and present information to others in a way that facilitates an understanding of a topic without having each student read the text in its entirety; by teaching others, they become experts.	After completing a free-response question, have different groups of students look at different responses to the question, such as student samples found on AP Central. Once students have discussed their solution, have them get into groups with students who examined different responses, and have them share the ways in which the response was correct.
Kinesthetic learning	Students' body movements are used to create knowledge or understanding of a new concept. A kinesthetic-tactile learning style requires students to manipulate or touch materials to learn.	Allows students to gain new understanding of or insight into a problem or solution by acting out the steps taken when executing a code segment.	Before introducing sorting algorithms, have a group of 10–15 students put themselves in sorted order based on height, and then ask participants and observers what they noticed about the process. Draw analogies from the process that students used to a similar sorting algorithm.

continued on next page

Cooperative Learning (cont'd)

Strategy	Definition	Purpose	Example
Sharing and responding	Students communicate with another person or a small group of peers who respond to a proposed problem or solution.	Gives students the opportunity to discuss their work with peers, make suggestions for improvements to the work of others, and/or receive appropriate and relevant feedback on their own work.	Before starting to implement a solution to a specification, have students determine the classes and methods that they plan on using. Before moving on to implementation, have students share their designs with others, who will provide feedback and suggestions.
Student response system	Students use a classroom response system or other means of electronic or nonelectronic communication to send answers or information in response to a question or request.	Provides immediate feedback on student understanding of material to facilitate instruction.	When covering object creation and method calls early in the year, a student response system can be used to ensure that all students understand the flow of execution, allowing for the early correction of misunderstandings and potential remediation on a foundational topic.
Think-pair-share	Students think through a problem alone, pair with a partner to share ideas, and then share results with the class.	Enables the development of initial ideas that are then tested with a partner in preparation for revising ideas and sharing them with a larger group.	Provide students with a short warm-up problem to solve; this could be a portion of code or a river-crossing brain teaser. Have students work on the problem on their own for a set period of time and then work with their partner. Have students share their solutions with the class.
Unplugged activities	Students use engaging games and puzzles that use manipulatives and kinesthetic learning activities.	Provides students with a different way of engaging with the material, away from the computer, to further understanding.	When learning about the structure of classes, take a printout of a simple class and cut it into different sections. As students enter the classroom, hand them an envelope full of the paper strips, and ask them to reassemble the class in order: class header, instance variables, constructors, methods.

continued on next page

Cooperative Learning (cont'd)

Strategy	Definition	Purpose	Example
Using manipulatives	Students use objects to examine relationships between the information given.	Provides a tactile or visual representation of data that supports comprehension of information in a problem.	When illustrating how values are passed to and returned from methods, have students act as methods and pass a foam ball to different students, and have them pass it back to you after performing their function. Values are taped to the ball to represent parameters and return values. To show the need for a variable to store the value returned by a method call, do not catch the ball when a student returns it to you to show how the value is "dropped."

Making Connections

Strategy	Definition	Purpose	Example
Activating prior knowledge	The teacher provides students an opportunity to recall what they already know about a concept and make connections to current studies.	Prepares students to establish content connections.	When introducing 2D arrays, have students spend time writing down everything they already know about 1D arrays, and then show how that relates to 2D arrays. Although there are similarities between the two, there are some differences that can cause confusion if not addressed early on.
Diagramming	Students use a visual representation to organize information.	Builds comprehension and facilitates discussion by representing information in visual form.	Use, or have students create, diagrams or flowcharts to visualize algorithms before implementing those algorithms in program code.
Note-taking	Students create a record of information while listening to a speaker.	Helps students accurately understand a concept and keep a record of program examples with correct syntax.	Students take notes on the proper syntax that would be used to traverse a 2D array in row-major order.

continued on next page

Making Connections (cont'd)

Strategy	Definition	Purpose	Example
Paraphrase	Students restate, in their own words, essential information expressed in a text.	Assists with comprehension, recall of information, and problem-solving.	Provide students with several methods that accomplish different tasks (such as finding the minimum or maximum, detecting duplicates, etc.), as well as a list of general descriptions like "finds the minimum" or "detects duplicates." Ask students to match the description with the appropriate method.
Quickwrite	Students write for a short, specific amount of time about a designated topic related to a given prompt.	Generates multiple ideas in a quick fashion.	To help synthesize concepts after learning about traversals, provide students with code to traverse an array or `ArrayList` object. Ask them to come up with as many equivalent code segments as they can.
Vocabulary organizer	Students use a graphic organizer with a designated format to maintain an ongoing record of vocabulary words with definitions, pictures, notation, and connections.	Provides a reinforcement of learned words and a personal, ever-present tool for building word knowledge and awareness.	At the end of each unit, review the vocabulary that students should know and have included in their vocabulary organizer. Provide the definition, and have students come up with their own examples or pictures that will help them remember the terms later in the course. There are several apps and websites that help students create flashcards and quizzes to help organize the vocabulary that they learn throughout the year.

Developing Computational Thinking Practices

Throughout the AP Computer Science A course, students will develop computational thinking practices that are fundamental to the discipline of computer science. Because computational thinking practices represent the complex skills demonstrated by adept computer scientists, students will benefit from multiple opportunities to develop these skills in a scaffolded manner.

All the multiple-choice questions on the AP Exam will be associated with one of the skills in the tables that follow, so providing students with practice applying these skills will help prepare them for the exam and set them up for success. In addition, each scoring criteria in the scoring guidelines for the free-response questions will be associated with skills in the practice of code implementation. The sample exam questions at the end of this course and exam description show how the questions relate to specific computational thinking practices.

Practice 1: ***Program Design and Algorithm Development***

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
1.A: Determine an appropriate program design to solve a problem or accomplish a task.	Read through a problem specification and determine appropriate classes and/or methods that need to be written.	Students will be presented with a free-response question where they will need to design a class to satisfy the given specification. While students are not assessed on the quality of their specific design, having practiced designing program code during class will allow students to more easily interpret the specification and design their class.	**Marking the text** When students are reading through a specification, have them highlight nouns and circle verbs to help determine what classes might be needed, as well as potential attributes and behaviors of those classes. **Additional Strategies** ▪ Sharing and responding

continued on next page

Practice 1: *Program Design and Algorithm Development (cont'd)*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
1.B: Determine code that would be used to complete code segments.	Given a section of incomplete program code, students will be asked to determine which code segment (Boolean expression, loop, `if` statement, etc.) should be used to complete the program code so that it functions as described.	Some questions will require students to determine appropriate Boolean expressions to meet the specification. Students often find it difficult to reason through decisions that require the use of the operator representing *not*. Teachers should provide students with practice using `!` (not), such as in `x = !x`. The topic of negation is often difficult for students the first time they see it. Some questions will require students to determine the appropriate loop header to produce the desired result or meet a specification. Students often confuse the role of the loop condition, which refers to when to *iterate*, not when to *stop*.	**Think-pair-share** Provide students with a code segment with a missing expression, and ask them what statement should be used to complete the code segment. After they have determined what they think, have them share with a partner to see if they have the same answer and can come to agreement. Then, have pairs share with the entire class. **Additional Strategies** ▪ Student response system
1.C: Determine code that would be used to interact with completed program code.	Given a section of completed program code (class or hierarchy of classes), students will be asked to determine the correct code segment to create objects and call methods to meet a specification.	Some questions focus on the syntax of a method or constructor call, including handling return values, using the correct object, use of keywords (e.g., `new`), and proper order of parameters. If a question uses inheritance hierarchies, students will need to demonstrate understanding of whether a method is able to be called and whether a reference is able to refer to a given type of object.	**Diagramming** Have students create UML diagrams of existing classes and methods and refer to those diagrams when deciding what method calls are legal and appropriate. **Additional Strategies** ▪ Think aloud

Practice 2: *Code Logic*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
2.A: Apply the meaning of specific operators.	Given a Java expression, apply mathematical and logical order of operations to compute a result.	Understand which mathematical and logical operators are required for this course by referring to the curriculum framework.	**Vocabulary organizer** When introducing new operators, provide students with the name and definition, and then have them come up with their own examples or pictures to help them remember the operators and how they function. **Additional Strategies** ▪ Activating prior knowledge
2.B: Determine the result or output based on statement execution order in a code segment without method calls (other than output).	Given program code containing conditional statements, iterative statements, or array traversals, determine the value stored in a variable (which could be an array) or what will be printed to the screen.	When evaluating the result of a program code segment, execution order matters. For example, when using string concatenation, `7 + 5 + "hello"` would yield `"12hello"`, while `"hello" + 7 + 5` would yield `"hello75"`.	**Code tracing** When teaching iteration, tracing tables are helpful for keeping track of loop variables, other program variables, and output. Have students make a column for each variable in their code and one for output. Each time through the loop is represented by a new row in the table, with the current value of each variable being stored in cells on the table. **Additional Strategies** ▪ Diagramming ▪ Student response system

continued on next page

Practice 2: *Code Logic (cont'd)*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
2.C: Determine the result or output based on the statement execution order in a code segment containing method calls.	Given program code that contains calls to instance, static, overridden, or recursive methods, such as `Math`, `ArrayList`, and `String` method calls, determine the value stored in a variable (which could be an array or `ArrayList`) or what will be printed to the screen.	When calling methods, if the actual parameters are primitive values, a copy of the value is sent to be stored in the formal parameter. Consequently, changes made to the value of the formal parameter in the method are not reflected in the actual parameter. Conversely, if the actual parameters are reference values, then the formal parameter is an alias of the actual parameter (both the actual and formal parameter will refer to the same object), and any changes made to the referenced object within the method persist after the method is called.	**Predict and compare** When introducing students to a new type of method call, have them predict the output of several different statements that use that type of method call. Once done, have them create a program that contains the same statements and compare actual and expected results. **Additional Strategies** ▪ Kinesthetic learning ▪ Simplify the problem
2.D: Determine the number of times a code segment will execute.	Given program code that contains iteration or recursion, determine the number of times the body of the loop, a given statement, or the recursive call will be executed.	When determining the number of times a code segment executes, students should be aware of the boundaries of the iteration to avoid being off by one in their count.	**Kinesthetic learning** When looking at a recursive method to determine how many times it executes, have students call each other as recursive methods. Students should stand up when "called." The count would be equivalent to the number of students in the room who are standing. **Additional Strategies** ▪ Simplify the problem

Practice 3: *Code Implementation*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
3.A: Write program code to create objects of a class and call methods.	Write program code to: ▪ create objects using constructors with and without parameters. ▪ call methods and overloaded methods.	When writing program code that contains multiple interacting classes, students will need to be sure that they are using the right methods for each object type.	**Modeling** When teaching how to create objects, model the correct way to declare a reference type variable, and then initialize it with a call to a constructor. This helps students begin to recognize the parts of the statement that change, such as the class name and parameter list, and what remains the same, such as the `new` keyword and the statement structure. **Additional Activities** ▪ Marking the text ▪ Sharing and responding
3.B: Write program code to define a new type by creating a class.	Write program code to: ▪ declare a class through a class header without inheritance. ▪ create classes with primitive and reference instance variables. ▪ define methods with and without return values and/or parameters. ▪ create classes in an inheritance hierarchy.	When creating subclasses, parent objects must exist before a child object (just like in real life), which is why the parent constructor must be called first, implicitly or explicitly.	**Sharing and responding** When students are developing their own class, have students share their implementation of a class with a partner to see if there is any expected or desired behavior missing. **Additional Strategies** ▪ Create a plan

continued on next page

Practice 3: *Code Implementation (cont'd)*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
3.C: Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.	Write program code: ▪ using expressions, conditional statements, or iterative statements containing primitive and reference values. ▪ using combinations of interacting expressions, conditional statements, and iterative statements. ▪ to implement modifications of standard algorithms that use conditionals or iterative statements.	Method specifications often require conditional and iterative statements to be nested together. If the conditional statements are located inside the body of the iterative statements, the conditions will be checked as many times as the statement executes.	**Note-taking** When first asking students to write their own methods, have them write down descriptions of the steps needed to define a method so that they can refer to a record of the process at a later point in time. **Additional Strategies** ▪ Pair programming
3.D: Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.	Write program code that: ▪ creates, traverses, parallel traverses, and utilizes or manipulates the elements in 1D array or `ArrayList` objects. ▪ traverses 1D array or `ArrayList` objects in an order other than from left to right. ▪ implements standard algorithms using 1D array or `ArrayList` objects.	When writing programs to traverse and manipulate arrays, it is important for students to remember that arrays have no remove method. Item removal can be mimicked by setting locations to a default value or shifting elements within the array to keep empty or available locations at the end. On the other hand, when items are added or removed in an `ArrayList`, items are automatically shifted to make room or remove empty spaces. One problem that many students have is forgetting to create the array or `ArrayList` when it is has been declared as an instance variable in a class declaration. It is not enough to declare the instance variable; the array or `ArrayList` needs to be created in the constructor.	**Unplugged activities** Before introducing searching algorithms, have students attempt to search for a specific item among a large group of unsorted items, making note of their process and also the time that it takes. Then have them search for an item in a large group of sorted items and note the difference in process and time taken. **Additional Strategies** ▪ Identify a subtask

continued on next page

Practice 3: *Code Implementation (cont'd)*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
3.E: Write program code to create, traverse, and manipulate elements in 2D array objects.	Write program code that: ▪ creates, traverses, parallel traverses, and utilizes or manipulates the elements in 2D array objects. ▪ traverses 2D array objects in an order other than row-major. ▪ implements standard algorithms using 2D array objects.	When writing program code that uses 2D arrays, students often forget how to access elements. It is important to stress that 2D array access uses `[row][col]` and not (row, col). Students also need to understand that 2D arrays are arrays of arrays. Using nested enhanced `for` loops or initializer lists are two ways to emphasize this for students.	**Jigsaw** Ask students to form groups of four and come up with a way of traversing a 2D array other than row-major order. Once they have their pattern, ask them to develop and implement the algorithm to match their traversal. Ask students to create four new groups comprised of one member from each of the former groups to present their pattern and program code for the traversal. **Additional Strategies** ▪ Activating prior knowledge ▪ Using manipulatives

Practice 4: *Code Testing*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
4.A: Use test-cases to find errors or validate results.	Given program code, ▪ develop a set of test cases that can be used to validate results or highlight errors. ▪ determine what can be verified by a given set of test cases.	When trying to find errors or validate results in program code that contains Boolean expressions, truth tables can be especially helpful. One error that students often encounter—and can be difficult to catch, because the code will still compile—is using a single = in a Boolean expression. Although the code will compile, it will not work as expected.	**Code tracing** Students can check whether they have been given correct output for a given method by tracing their code with input values that are around the extreme ranges of the input values. **Additional Strategies** ▪ Sharing and responding

continued on next page

Practice 4: ***Code Testing (cont'd)***

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
4.B: Identify errors in program code.	Given program code, identify compile-time and run-time errors.	A common error is trying to access or use variables when they do not exist. Local variables are only accessible in the block in which they are declared. Students need to remember that they cannot use local variables in other methods or outside of the block in which those variables are declared (such as a loop). Another common error is confusion between array and `ArrayList` access. This includes confusion of `[ ]` with `get( )`, as well as confusion between `.length` and `.size( )`	**Predict and compare** When students begin to write methods, provide them with the specifications for several methods and a collection of method headers. Have them select which method header belongs with which specification, and then have them implement each method. **Additional Strategies** ▪ Error analysis
4.C: Determine if two or more code segments yield equivalent results.	Given two or more program code segments, typically `if` statements, loops, or array/`ArrayList` traversals, compare and contrast to determine if they yield the same results.	When students are comparing and contrasting code segments, it can be helpful to provide them with code that uses an array and ask them to rewrite it using an `ArrayList`. This same type of activity can be done using `while` and `for` loops, or nested and non-nested statements.	**Think-pair-share** Ask students to consider two program code segments that are meant to yield the same result. Have them take a few minutes to think independently about the benefits of one segment over the other. Then, ask students to compare and add to this list with their partner and finally to share with the whole group. **Additional Strategies** ▪ Predict and compare

Practice 5: *Documentation*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
5.A: Describe the behavior of a given segment of program code.	Given program code, describe the general behavior that might be included in program documentation.	When looking at program code that contains loops, students need to know that "If loops" do not exist. Code that executes one time should be contained within an `if` statement. Code that may need to repeat should be contained within the body of a loop.	**Paraphrase** When covering nested conditionals and compound Boolean expressions, provide different code segments to students. Have them paraphrase what is happening in the code, as opposed to just explaining each line or expression individually. Then, have them pair up with students whose code accomplishes the same task in a different manner. **Additional Strategies** ▪ Look for a pattern ▪ Sharing and responding
5.B: Explain why a code segment will not compile or work as intended.	Given program code without example test cases, determine why it might not compile or, if it would compile, why it won't work as intended. This goes beyond finding the error and involves explaining the consequences of having the error in the program.	When trying to determine why a code segment will not work as intended, there are several things that should be examined. If return statements are included inside both the `if` and `else` clauses within a loop, an early return is ensured. It is also important for students to understand that constructors do not have a return type, and it should be emphasized that constructors are not methods.	**Think aloud** When determining why a particular method fails to provide the correct output, students should ask themselves a series of questions out loud to identify potential locations of errors, such as parameter mismatch, incorrect Boolean expressions or loop bounds, or incorrect order of expressions. **Additional Strategies** ▪ Quickwrite

continued on next page

Practice 5: *Documentation (cont'd)*

Skills	Key Tasks	Instructional Notes	Sample Activities and Strategies
5.C: Explain how the result of program code changes, given a change to the initial code.	Given program code and a proposed change, explain how it will affect the result of execution.	Since there are many ways to write a program to accomplish the same task, it is helpful for students to be exposed to many different ways to write a solution. This allows them to see that while their own solution may be different from another, it may still be correct.	**Discussion group** When looking at how the outcome of program code can change given a change to initial code, provide students with examples of static variables (unique identification numbers, the number of objects created, etc.) to help distinguish between static and instance variables. **Additional Strategies** ▪ Jigsaw
5.D: Describe the initial conditions that must be met for a program segment to work as intended or described.	Given program code, describe the initial conditions stated explicitly in the preconditions of the program documentation or implicitly required based on the methods being called in the code segment.	The methods and classes presented in free-response questions contain both pre- and postconditions that provide valuable information about what needs to be true prior to executing methods and what the outcome will be when using these methods. While the compiler ignores these comments, students should not, as they provide valuable insight into how the program code is written.	**Marking the text** Provide students with a complete class implementation with associated Java documentation, and have them circle the pre- and postconditions for each method. **Additional Strategies** ▪ Ask the expert

Using an Iterative Development Process

The instructional strategies tables include strategies that facilitate students' use of a process to plan and create computer programs. Planning processes are iterative and cyclical in nature and require students to reflect on what they have created. If necessary, students should return to prior stages to modify their plans and change their development. This iterative process of development and revision allows students to have more confidence in their solution and in their ability to develop solutions. Allowing students to revise and improve on previous work is also another way to provide feedback and can be used as a method of formative assessment throughout the course.

Several of the computational thinking practices for this course represent phases that can be seen in a development process to plan and create computer programs. Throughout any design process, students should collaborate when developing a program.

- **Program Design and Algorithm Development:** In this phase, students create a representation or model of their solution. Strategies such as **diagramming** and **using manipulatives** can give students confidence in the design of their solution.

- **Code Implementation:** In this phase, students implement their solution by writing a program. The strategy of **pair programming** allows students to have constant support during implementation. To successfully implement a program, students need to understand the operators and program execution order needed to produce the desired results.
- **Code Testing:** In this phase, students evaluate and test the solution they implemented. Teachers can use strategies such as **predict and compare** and **error analysis** to determine whether the solution is appropriate. This phase may require students to revisit earlier phases of development to make improvements on their solution.
- **Documentation:** In this phase, students describe what the program does, often adding comments to program code. Students could keep a journal or log book to note design decisions and rationales. This allows students to come back to a program later and recall how it was constructed. This is helpful when modifications need to be made to an existing program, either by the student who wrote the initial version or by another student. Strategies such as **note-taking** or **paraphrasing** can help students document their process at different stages.

The phases listed above need not be implemented in linear order. Students may choose to return to earlier phases as their design ideas change and develop.

Teaching the Big Ideas

As demonstrated in the unit guides and topic pages, the big ideas spiral throughout the course. When teaching the big ideas and building conceptual understanding, it is important to give students scaffolded practice with various components of the big ideas. Below are some of the implicit elements of big ideas that should be called out and developed throughout the year.

- **Modularity:** Modularity requires students to simplify concepts and processes by looking at the big picture rather than the details and developing abstractions. Whether this is in the representation of objects or concepts, in the use of preexisting processes, or in the creation and organization of code into different methods or classes. There are several instructional strategies that can help students make these connections, including **using manipulatives** and **diagramming.**
- **Variables:** Utilizing variables to help simplify concepts and processes is an important component of this big idea and involves the creation of data abstractions. In order to help students see the role that variables play in generalizable solutions, try the instructional strategies of **creating a plan** and **identifying a subtask.** These strategies can also be used to show how variables manage large amounts of data or complex relationships. In addition, when attempting to define or interpret processes and the role that variables play in those processes, strategies of **kinesthetic learning** and **look for a pattern** are helpful.
- **Control:** Being able to write and evaluate mathematical expressions is a necessary component in determining the computed result of an expression. **Code tracing, error analysis,** and **simplify the problem** are strategies that can help students understand the relationship between variables and quantities and provide them opportunities to practice writing and evaluating mathematical expressions. Other strategies like **pair programming** and **predict and compare** provide students with practice developing algorithmic thinking, such as defining and interpreting processes that are used in a program, like selection and iteration.
- **Impact of Computing:** Strategies such as **discussion group, jigsaw,** and **think aloud** provide students opportunities to explain the cause and effect of a program. These opportunities can help distinguish a program's intended purpose from unintended consequences.

Using Strategies for Collaboration

Effective collaboration has been shown to have significant positive impacts on students; however, this does not simply mean having students work in groups. Using appropriate strategies and creating a collaborative environment in the classroom—one that places a strong emphasis on valuing and discussing contributions of ideas from all group members—helps student engagement and confidence and encourages participation from a wider population of students. Students should be given many opportunities to practice collaboration throughout the course, especially as they consider different collaborative methods that will help improve their programs using an iterative development process.

Some effective ways to incorporate collaboration in the classroom and during the development of programs include the following:

- Have an established protocol for students to equally participate and share their ideas, such as the **think-pair-share** strategy. This strategy builds students' confidence and can create a successful collaborative learning community within a classroom.
- Provide students with opportunities to work together to solve problems. River-crossing problems and critical thinking problem-solving questions can be solved collaboratively.
- Brainstorm ideas and solutions in a team environment.
- Work together to design subtasks of a larger project, develop those subtasks, and integrate them in the completion of the project.
- Implement a solution together using **pair programming.** One student (the driver) writes program code, while the other student (the observer, pointer, or navigator) reviews each line of program code as it is typed in. The partners change roles after designated time intervals. For multiday projects, teachers can have students rotate every 20–30 minutes. Smaller programming projects can be used as warm-up problems where students rotate roles after each problem.
- Have students provide feedback to each other on a program at various points in the development process to improve the overall quality.
- Have students provide technical support to each other when working to solve a problem on their own.

Using Real-World Data

Data and information are very important in the field of computer science. Many of our computer systems are heavily reliant on the collection and use of data to determine important information or new knowledge. Data is what makes simulations more robust; for example, it is what ensures that you get appropriate suggestions based on your viewing preferences when streaming content, and it is what helps you navigate through many of the apps on today's technological devices. As students learn about data and how it can be used, their understanding can help them see the connection between computer science and other careers and fields of study. One of the best ways for students to learn about and interact with data, while also reinforcing the concepts in the course, is to use real-world data to solve a problem of interest. This can include the following steps:

- In discussion groups, or using online tools for collaboration, have students read about and examine possible sources of data in various fields, such as medicine, business, criminal justice, marketing, civil engineering, and municipal planning.
- Have students generate and pose questions about a set of data and use **sharing and responding** to refine those questions.
- Use **pair programming** to have students develop a program to process information from a large, real-world data set.

Using Input

Although this course does not prescribe or test an approach for getting data into a program through specific learning objectives focused on input, input is a necessary component for any computer programming course. There are several ways to accomplish input, such as the `Scanner` class or `JOptionPane`, and teachers should use an approach that fits their style, textbook, or other materials in use. Input is essential for the creation of interesting and engaging programs and should therefore play an important role in the programming practice that takes place throughout the year.

Several of the labs for AP Computer Science A include the opportunity to bring in real-world data, and the more that data can be used as part of everyday practice, the more comfortable students will become. By showing students how to solve problems and answer questions using real-world data sets, we are reinforcing the power of computer science and its applications to other disciplines.

There are a variety of ways to bring in data, and it is not necessary for students to write this code themselves. Providing students with starter code that can be used to read or process data can allow for richer programming exercises, and examples of this type of started code can be found on the online teacher community.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

Exam Information


elloWorld.ja
*/
public class He
{
pu
ain

Exam Overview

The AP Computer Science A Exam assesses student understanding of the computational thinking practices and learning objectives outlined in the course framework. The exam is 3 hours long and includes 40 multiple-choice questions and 4 free-response questions. As part of the exam, students will be given the Java Quick Reference (see Appendix), which lists accessible methods from the Java library that may be included in the exam. The details of the exam, including exam weighting and timing, can be found below:

Section	Question Type	Number of Questions	Exam Weighting	Timing
I	Multiple-choice questions	40	50%	90 minutes
II	Free-response questions	4		90 minutes
	Question 1: Methods and Control Structures (9 points)		12.5%	
	Question 2: Class (9 points)		12.5%	
	Question 3: Array/`ArrayList` (9 points)		12.5%	
	Question 4: 2D Array (9 points)		12.5%	

The exam assesses content from the three big ideas for the course:
Big Idea 1: Modularity
Big Idea 2: Variables
Big Idea 3: Control

The AP Exam also assesses each of the 10 units of the course with the following weighting on the multiple-choice section:

Units	Exam Weighting
Unit 1: Primitive Types	2.5–5%
Unit 2: Using Objects	5–7.5%
Unit 3: Boolean Expressions and `if` Statements	15–17.5%
Unit 4: Iteration	17.5–22.5%
Unit 5: Writing Classes	5–7.5%
Unit 6: Array	10–15%
Unit 7: `ArrayList`	2.5–7.5%
Unit 8: 2D Array	7.5–10%
Unit 9: Inheritance	5–10%
Unit 10: Recursion	5–7.5%

How Student Learning Is Assessed on the AP Exam

The AP Computer Science A computational thinking practices are assessed on the AP Exam as detailed below.

Section I: Multiple-Choice

The AP Computer Science A Exam multiple-choice section includes mostly individual questions, with one or two sets of multiple questions (typically two to three questions per set).

Computational Thinking Practices 1, 2, 4, and 5 are all assessed in in the multiple-choice section with the following exam weighting (Computational Thinking Practice 3 is not assessed in the multiple-choice section):

Computational Thinking Practice	Exam Weighting
Practice 1: Program Design and Algorithm Development	**30–35%**
Practice 2: Code Logic	**40–45%**
Practice 4: Code Testing	**12–18%**
Practice 5: Documentation	**12–18%**

Section II: Free-Response

The second section of the AP Computer Science Exam includes four free-response questions, all of which assess Computational Thinking Practice 3: Code Implementation. All five skills within this practice are assessed across the four free-response questions, with the following skill focus for each question:

Free-response question 1: Methods and Control Structures focuses on assessing students' ability to:

- write program code to create objects of a class and call methods (Skill 3.A)
- write program code to satisfy method specifications using expressions, conditional statements, and iterative statements (Skill 3.C)

Free-response question 2: Class focuses on assessing students' ability to:

- write program code to define a new type by creating a class (Skill 3.B)
- write program code to satisfy method specifications using expressions, conditional statements, and iterative statements (Skill 3.C)

Free-response question 3: Array/`ArrayList` focuses on assessing students' ability to:

- write program code to satisfy method specifications using expressions, conditional statements, and iterative statements (Skill 3.C)
- write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects (Skill 3.D)

Free-response question 4: 2D Array focuses on assessing students' ability to:

- write program code to satisfy method specifications using expressions, conditional statements, and iterative statements (Skill 3.C)
- write program code to create, traverse, and manipulate elements in 2D array objects (Skill 3.E)

Task Verbs Used in Free-Response Questions

The following task verbs are commonly used in the free-response questions:

Assume: Suppose to be the case without any proof or need to further address the condition.

Complete (program code): Express in print form the proper syntax to represent a described algorithm or program given part of the code.

Implement/Write: Express in print form the proper syntax to represent a described algorithm or program.

Sample Exam Questions

The sample exam questions that follow illustrate the relationship between the course framework and the AP Computer Science A Exam and serve as examples of the types of questions that appear on the exam. After the sample questions is a table that shows to which skill, learning objective(s), and unit each question relates. The table also provides the answers to the multiple-choice questions.

Section I: Multiple-Choice

1. Consider the following code segment.

```
int a = 5;
int b = 2;
double c = 3.0;
System.out.println(5 + a / b * c - 1);
```

What is printed when the code segment is executed?

(A) `0.666666666666667`

(B) `9.0`

(C) `10.0`

(D) `11.5`

(E) `14.0`

2. Consider the `processWords` method. Assume that each of its two parameters is a `String` of length two or more.

```
public void processWords(String word1, String word2)
{
   String str1 = word1.substring(0, 2);
   String str2 = word2.substring(word2.length() - 1);
   String result = str2 + str1;
   System.out.println(result.indexOf(str2));
}
```

Which of the following best describes the value printed when `processWords` is called?

(A) The value `0` is always printed.

(B) The value `1` is always printed.

(C) The value `result.length() - 1` is printed.

(D) A substring containing the last character of `word2` is printed.

(E) A substring containing the last two characters of `word2` is printed.

3. Which of the following statements assigns a random integer between 25 and 60, inclusive, to `rn`?

(A)
```
int rn = (int) (Math.random() * 25) + 36;
```
(B)
```
int rn = (int) (Math.random() * 25) + 60;
```
(C)
```
int rn = (int) (Math.random() * 26) + 60;
```
(D)
```
int rn = (int) (Math.random() * 36) + 25;
```
(E)
```
int rn = (int) (Math.random() * 60) + 25;
```

4. Vehicles are classified based on their total interior volume. The `classify` method is intended to return a vehicle classification `String` value based on total interior volume, in cubic feet, as shown in the table below.

Vehicle size class	Total interior volume
Minicompact	Less than 85 cubic feet
Subcompact	85 to 99 cubic feet
Compact	100 to 109 cubic feet
Mid-Size	110 to 119 cubic feet
Large	120 cubic feet or more

The `classify` method, which does not work as intended, is shown below.

```
public static String classify(int volume)
{
    String carClass = "";
    if (volume >= 120)
    {
       carClass = "Large";
    }
    else if (volume < 120)
    {
       carClass = "Mid-Size";
    }
    else if (volume < 110)
    {
       carClass = "Compact";
    }
    else if (volume < 100)
    {
       carClass = "Subcompact";
    }
```

```
    else
    {
       carClass = "Minicompact";
    }
    return carClass;
}
```

The `classify` method works as intended for some but not all values of the parameter `volume`. For which of the following values of `volume` would the correct value be returned when the `classify` method is executed?

(A) 80

(B) 90

(C) 105

(D) 109

(E) 115

5. Which of the following best describes the value of the Boolean expression shown below?

```
a && !(b || a)
```

(A) The value is always `true`.

(B) The value is always `false`.

(C) The value is `true` when `a` has the value `false`, and is `false` otherwise.

(D) The value is `true` when `b` has the value `false`, and is `false` otherwise.

(E) The value is `true` when either `a` or `b` has the value `true`, and is `false` otherwise.

6. Consider the following code segment.

```
int val = 48;
int div = 6;
while ((val % 2 == 0) && div > 0)
{
    if (val % div == 0)
    {
       System.out.print(val + " ");
    }
    val /= 2;
    div--;
}
```

What is printed when the code segment is executed?

(A) `48 12 6`

(B) `48 12 6 3`

(C) `48 12 6 3 1`

(D) `48 24 12 6`

(E) `48 24 12 6 3`

7. Consider the following class definition.

```
public class Example
{
    private int x;
    // Constructor not shown.
}
```

Which of the following is a correct header for a method of the `Example` class that would return the value of the `private` instance variable `x` so that it can be used in a class other than `Example`?

(A) `private int getX()`

(B) `private void getX()`

(C) `public int getX()`

(D) `public void getX()`

(E) `public void getX(int x)`

8. In the following code segment, assume that the string `str` has been properly declared and initialized. The code segment is intended to print the number of strings in the array `animals` that have `str` as a substring.

```
String[] animals = {"horse", "cow", "goat", "dog",
                    "cat", "mouse"};
int count = 0;
for (int i = 0; i <= animals.length; i++)
{
    if (animals[i].indexOf(str) >= 0)
    {
       count++;
    }
}
System.out.println(count);
```

Which of the following changes should be made so the code segment works as intended?

(A) The Boolean expression in the for loop header should be changed to `i < animals.length`.

(B) The Boolean expression in the for loop header should be changed to `i < animals.length -1`.

(C) The Boolean expression in the for loop header should be changed to `i < animals[i].length`.

(D) The condition in the `if` statement should be changed to `animals[i].equals(str)`.

(E) The condition in the `if` statement should be changed to `animals[i].substring(str)`.

9. Consider an integer array, `nums`, which has been declared and initialized with one or more integer values. Which of the following code segments updates `nums` so that each element contains the square of its original value?

I.

```
int k = 0;
while (k < nums.length)
{
   nums[k] = nums[k] * nums[k];
}
```

II.

```
for (int k = 0; k < nums.length; k++)
{
   nums[k] = nums[k] * nums[k];
}
```

III.

```
for (int n : nums)
{
   n = n * n;
}
```

(A) II only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

10. Consider the following code segment.

```
ArrayList<String> numbers = new ArrayList<String>();
numbers.add("one");
numbers.add("two");
numbers.add(0, "three");
numbers.set(2, "four");
numbers.add("five");
numbers.remove(1);
```

Which of the following represents the contents of `numbers` after the code segment has been executed?

(A) `["one", "four", "five"]`

(B) `["three", "two", "five"]`

(C) `["three", "four", "two"]`

(D) `["three", "four", "five"]`

(E) `["one", "two", "three", "four", "five"]`

11. Consider the following method, which is intended to return a list containing the elements of the parameter `myList` with all even elements removed.

```
public static ArrayList<Integer> removeEvens
   (ArrayList<Integer> myList)
{
   for (int i = 0; i < myList.size(); i++)
   {
         if (myList.get(i) % 2 == 0)
         {
             myList.remove(i);
         }
   }
   return myList;
}
```

Which of the following best explains why the code segment does not work as intended?

(A) The code segment causes an `IndexOutOfBoundsException` for all lists because of an incorrect Boolean expression in the for loop.

(B) The code segment causes an `IndexOutOfBoundsException` for lists with at least one even element because the indexes of all subsequent elements change by one when a list element is removed.

(C) The code segment returns a list with fewer elements than intended because it fails to consider the last element of `myList`.

(D) The code segment removes the wrong elements of `myList` because the condition in the `if` statement to test whether an element is even is incorrect.

(E) The code segment skips some elements of `myList` because the indexes of all subsequent elements change by one when a list element is removed.

12. Consider the following code segment.

```
int[][] points = {{11, 12, 13, 14, 15},
                  {21, 22, 23, 24, 25},
                  {31, 32, 33, 34, 35},
                  {41, 42, 43, 44, 45}};
for (int row = 0; row < points.length; row++)
{
   for (int col = points[0].length - 1;
        col >= row; col--)
```

```
    {
        System.out.print(points[row][col] + " ");
    }
    System.out.println();
}
```

What is printed when this code segment is executed?

(A) 15 14
```
25 24 23
35 34 33 32
45 44 43 42 41
```

(B) 15 14 13 12
```
25 24 23
35 34
45
```

(C) 11 12 13 14 15
```
21 22 23 24
31 32 33
41 42
```

(D) 15 14 13 12 11
```
25 24 23 22
35 34 33
45 44
```

(E) 15 14 13 12 11
```
25 24 23 22 21
35 34 33 32 31
45 44 43 42 41
```

13. Consider the following code segment.

```
int[][] arr = {{1,  2,  3,  4},
               {5,  6,  7,  8},
               {9, 10, 11, 12}};
int sum = 0;
for (int[] x : arr)
{
    for (int y = 0; y < x.length - 1; y++)
    {
        sum += x[y];
    }
}
```

What is the value of `sum` as a result of executing the code segment?

(A) 36

(B) 54

(C) 63

(D) 68

(E) 78

14. Consider the following class definitions.

```
public class Thing1
{
   public void calc(int n)
   {
     n *= 3;
     System.out.print(n);
   }
}

public class Thing2 extends Thing1
{
   public void calc(int n)
   {
     n += 2;
     super.calc(n);
     System.out.print(n);
   }
}
```

The following code segment appears in a class other than `Thing1` or `Thing2`.

```
Thing1 t = new Thing2();
t.calc(2);
```

What is printed as a result of executing the code segment?

(A) 4

(B) 6

(C) 68

(D) 124

(E) 1212

15. Consider the following two methods, which are intended to return the same values when they are called with the same positive integer parameter `n`.

```
public static int mystery1(int n)
{
   if (n > 1)
   {
     return 5 + mystery1(n - 1);
   }
   else
   {
     return 1;
   }
}
```

```
public static int mystery2(int n)
{
   int total = 0;
   int x = 1;
   while (x < n)
   {
     total += 5;
     x++;
   }
   return total;
}
```

Which, if any, of the following changes to `mystery2` is required so that the two methods work as intended?

(A) The variable `total` should be initialized to `1`.

(B) The variable `x` should be initialized to `0`.

(C) The condition in the `while` loop header should be `x < n - 1`.

(D) The condition in the `while` loop header should be `x <= n`.

(E) No change is required; the methods, as currently written, return the same values when they are called with the same positive integer parameter `n`.

Section II: Free-Response

The following are examples of the kinds of free-response questions found on the exam. Note that on the actual AP Exam, there will be four free-response questions.

Methods and Control Structures (Free-Response Question 1 on the AP Exam)

This question involves the use of *check digits*, which can be used to help detect if an error has occurred when a number is entered or transmitted electronically. An algorithm for computing a check digit, based on the digits of a number, is provided in part (a).

The `CheckDigit` class is shown below. You will write two methods of the `CheckDigit` class.

```
public class CheckDigit
{
   /** Returns the check digit for num, as described in part (a).
    *  Precondition: The number of digits in num is between one and
    *  six, inclusive.
    *            num >= 0
    */
    public static int getCheck(int num)
    {
        /* to be implemented in part (a) */
    }

   /** Returns true if numWithCheckDigit is valid, or false
    *  otherwise, as described in part (b).
    *  Precondition: The number of digits in numWithCheckDigit
    *  is between two and seven, inclusive.
```

```
     *              numWithCheckDigit >= 0
     */
    public static boolean isValid(int numWithCheckDigit)
    {
      /* to be implemented in part (b) */
    }

    /** Returns the number of digits in num. */
    public static int getNumberOfDigits(int num)
    {
      /* implementation not shown */
    }

    /** Returns the nthdigit of num.
     *  Precondition: n >= 1 and n <= the number of digits in num
     */
    public static int getDigit(int num, int n)
    {
      /* implementation not shown */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

(a) Complete the `getCheck` method, which computes the check digit for a number according to the following rules.

- Multiply the first digit by 7, the second digit (if one exists) by 6, the third digit (if one exists) by 5, and so on. The length of the method's `int` parameter is at most six; therefore, the last digit of a six-digit number will be multiplied by 2.
- Add the products calculated in the previous step.
- Extract the check digit, which is the rightmost digit of the sum calculated in the previous step.

The following are examples of the check-digit calculation.

Example 1, where num has the value 283415

- The sum to calculate is $(2 \times 7) + (8 \times 6) + (3 \times 5) + (4 \times 4) + (1 \times 3) + (5 \times 2)$ $= 14 + 48 + 15 + 16 + 3 + 10 = 106$.
- The check digit is the rightmost digit of 106, or 6, and `getCheck` returns the integer value 6.

Example 2, where num has the value 2183

- The sum to calculate is $(2 \times 7) + (1 \times 6) + (8 \times 5) + (3 \times 4) = 14 + 6 + 40 + 12 = 72$.
- The check digit is the rightmost digit of 72, or 2, and `getCheck` returns the integer value 2.

Two helper methods, `getNumberOfDigits` and `getDigit`, have been provided.

- `getNumberOfDigits` returns the number of digits in its `int` parameter.
- `getDigit` returns the nth digit of its `int` parameter.

The following are examples of the use of `getNumberOfDigits` and `getDigit`.

Method Call	Return Value	Explanation
`getNumberOfDigits(283415)`	6	The number 283415 has 6 digits.
`getDigit(283415, 1)`	2	The first digit of 283415 is 2.
`getDigit(283415, 5)`	1	The fifth digit of 283415 is 1.

Complete the `getCheck` method below. You must use `getNumberOfDigits` and `getDigit` appropriately to receive full credit.

```
/** Returns the check digit for num, as described in part (a).
 *  Precondition: The number of digits in num is between one and six,
 *  inclusive.
 *                num >= 0
 */
public static int getCheck(int num)
```

(b) Write the `isValid` method. The method returns `true` if its parameter `numWithCheckDigit`, which represents a number containing a check digit, is valid, and `false` otherwise. The check digit is always the rightmost digit of `numWithCheckDigit`.

The following table shows some examples of the use of `isValid`.

Method Call	Return Value	Explanation
`getCheck(159)`	2	The check digit for 159 is 2.
`isValid(1592)`	`true`	The number 1592 is a valid combination of a number (159) and its check digit (2).
`isValid(1593)`	`false`	The number 1593 is not a valid combination of a number (159) and its check digit (3) because 2 is the check digit for 159.

Complete method `isValid` below. Assume that `getCheck` works as specified, regardless of what you wrote in part (a). You must use `getCheck` appropriately to receive full credit.

```
/** Returns true if numWithCheckDigit is valid, or false
 *  otherwise, as described in part (b).
 *  Precondition: The number of digits in numWithCheckDigit is
 *  between two and seven, inclusive.
```

```
 *                numWithCheckDigit >= 0
 */
public static boolean isValid(int numWithCheckDigit)
```

Array/ArrayList (Free-Response Question 3 on the AP Exam)

The `Gizmo` class represents gadgets that people purchase. Some `Gizmo` objects are electronic and others are not. A partial definition of the `Gizmo` class is shown below.

```
public class Gizmo
{
   /** Returns the name of the manufacturer of this Gizmo. */
   public String getMaker()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is electronic, and false
    *  otherwise.
    */
   public boolean isElectronic()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is equivalent to the Gizmo
    *  object represented by the
    *  parameter, and false otherwise.
    */
   public boolean equals(Object other)
   {
      /* implementation not shown */
   }

   // There may be instance variables, constructors, and methods not shown.
}
```

The `OnlinePurchaseManager` class manages a sequence of `Gizmo` objects that an individual has purchased from an online vendor. You will write two methods of the `OnlinePurchaseManager` class. A partial definition of the `OnlinePurchaseManager` class is shown below.

```
public class OnlinePurchaseManager
{
   /** An ArrayList of purchased Gizmo objects,
    *  instantiated in the constructor.
    */
   private ArrayList<Gizmo> purchases;

   /** Returns the number of purchased Gizmo objects that are electronic
    *  whose manufacturer is maker, as described in part (a).
    */
```

```
    public int countElectronicsByMaker(String maker)
    {
        /* to be implemented in part (a) */
    }

    /** Returns true if any pair of adjacent purchased Gizmo objects are
     *  equivalent, and false otherwise, as described in part (b).
     */
    public boolean hasAdjacentEqualPair()
    {
        /* to be implemented in part (b) */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

(a) Write the `countElectronicsByMaker` method. The method examines the `ArrayList` instance variable `purchases` to determine how many `Gizmo` objects purchased are electronic and are manufactured by `maker`.

Assume that the `OnlinePurchaseManager` object `opm` has been declared and initialized so that the `ArrayList purchases` contains `Gizmo` objects as represented in the following table.

Index in purchases	0	1	2	3	4	5
Value returned by method call `isElectronic()`	`true`	`false`	`true`	`false`	`true`	`false`
Value returned by method call `getMaker()`	`"ABC"`	`"ABC"`	`"XYZ"`	`"lmnop"`	`"ABC"`	`"ABC"`

The following table shows the value returned by some calls to `countElectronicsByMaker`.

Method Call	Return Value
`opm.countElectronicsByMaker("ABC")`	2
`opm.countElectronicsByMaker("lmnop")`	0
`opm.countElectronicsByMaker("XYZ")`	1
`opm.countElectronicsByMaker("QRP")`	0

Complete method `countElectronicsByMaker` below.

```
/** Returns the number of purchased Gizmo objects that are electronic and
 *  whose manufacturer is maker, as described in part (a).
 */
public int countElectronicsByMaker(String maker)
```

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user's purchase history to detect such occurrences and request confirmation.

Write the `hasAdjacentEqualPair` method. The method detects whether two adjacent `Gizmo` objects in `purchases` are equivalent, using the `equals` method of the `Gizmo` class. If an adjacent equivalent pair is found, the `hasAdjacentEqualPair` method returns `true`. If no such pair is found, or if `purchases` has fewer than two elements, the method returns `false`.

Complete method `hasAdjacentEqualPair` below.

```
/** Returns true if any pair of adjacent purchased Gizmo objects are
 *  equivalent, and false otherwise, as described in part (b).
 */
public boolean hasAdjacentEqualPair()
```

Answer Key and Question Alignment to Course Framework

Multiple-Choice Question	Answer	Skill	Learning Objective	Unit
1	C	2.A	CON-1.A	1
2	A	5.A	VAR-1.E.b	2
3	D	1.C	CON-1.D	2
4	E	4.A	CON-2.A	3
5	B	5.A	CON-1.G	3
6	A	2.B	CON-2.C	4
7	C	1.C	MOD-2.D	5
8	A	4.B	VAR-2.B	6
9	A	1.B	VAR-2.C	6
10	D	2.C	VAR-2.D	7
11	E	5.B	VAR-2.E	7
12	D	2.B	VAR-2.G.a	8
13	B	2.B	VAR-2.G.b	8
14	D	2.C	MOD-3.B	9
15	A	4.C	CON-2.O	10

Free-Response Question	Question Type	Skill	Learning Objective	Unit
1	Methods and Control Structures	3.A, 3.C	MOD-1.H, CON-1.A, CON-1.E, CON-2.A CON-2.E	1,2,3,4,5
3	Array / `ArrayList`	3.A, 3.C, 3.D	MOD-1.G, CON-1.B, CON-1.H, CON-2.C, CON-2.J.b, VAR-1.E.b, VAR-2.D, VAR-2.E	1,2,3,4,5,7

The scoring information for the questions within this course and exam description, along with further exam resources, can be found on the AP Computer Science A Exam Page on AP Central.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

Scoring Guidelines

Question 1: Methods and Control Structures

This question involves the use of *check digits*, which can be used to help detect if an error has occurred when a number is entered or transmitted electronically. An algorithm for computing a check digit, based on the digits of a number, is provided in part (a).

The `CheckDigit` class is shown below. You will write two methods of the `CheckDigit` class.

```
public class CheckDigit
{
    /** Returns the check digit for num, as described in part (a).
     *  Precondition: The number of digits in num is between one and
     *  six, inclusive.
     *            num >= 0
     */
     public static int getCheck(int num)
     {
         /* to be implemented in part (a) */
     }

    /** Returns true if numWithCheckDigit is valid, or false
     *  otherwise, as described in part (b).
     *  Precondition: The number of digits in numWithCheckDigit
     *  is between two and seven, inclusive.
     *              numWithCheckDigit >= 0
     */
    public static boolean isValid(int numWithCheckDigit)
    {
      /* to be implemented in part (b) */
    }

    /** Returns the number of digits in num. */
   public static int getNumberOfDigits(int num)
    {
      /* implementation not shown */
    }

    /** Returns the nthdigit of num.
     *  Precondition: n >= 1 and n <= the number of digits in num
     */
    public static int getDigit(int num, int n)
    {
      /* implementation not shown */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

(a) Complete the `getCheck` method, which computes the check digit for a number according to the following rules.

- Multiply the first digit by 7, the second digit (if one exists) by 6, the third digit (if one exists) by 5, and so on. The length of the method's `int` parameter is at most six; therefore, the last digit of a six-digit number will be multiplied by 2.
- Add the products calculated in the previous step.
- Extract the check digit, which is the rightmost digit of the sum calculated in the previous step.

The following are examples of the check-digit calculation.

Example 1, where num has the value 283415

- The sum to calculate is $(2 \times 7) + (8 \times 6) + (3 \times 5) + (4 \times 4) + (1 \times 3) + (5 \times 2) = 14 + 48 + 15 + 16 + 3 + 10 = 106$.
- The check digit is the rightmost digit of 106, or 6, and `getCheck` returns the integer value 6.

Example 2, where num has the value 2183

- The sum to calculate is $(2 \times 7) + (1 \times 6) + (8 \times 5) + (3 \times 4) = 14 + 6 + 40 + 12 = 72$.
- The check digit is the rightmost digit of 72, or 2, and `getCheck` returns the integer value 2.

Two helper methods, `getNumberOfDigits` and `getDigit`, have been provided.

- `getNumberOfDigits` returns the number of digits in its `int` parameter.
- `getDigit` returns the nth digit of its `int` parameter.

The following are examples of the use of `getNumberOfDigits` and `getDigit`.

Method Call	Return Value	Explanation
`getNumberOfDigits(283415)`	6	The number 283415 has 6 digits.
`getDigit(283415, 1)`	2	The first digit of 283415 is 2.
`getDigit(283415, 5)`	1	The fifth digit of 283415 is 1.

Complete the `getCheck` method below. You must use `getNumberOfDigits` and `getDigit` appropriately to receive full credit.

```
/** Returns the check digit for num, as described in part (a).
 *  Precondition: The number of digits in num is between one and six,
 *  inclusive.
 *                num >= 0
 */
public static int getCheck(int num)
```

(b) Write the `isValid` method. The method returns `true` if its parameter `numWithCheckDigit`, which represents a number containing a check digit, is valid, and `false` otherwise. The check digit is always the rightmost digit of `numWithCheckDigit`.

The following table shows some examples of the use of `isValid`.

Method Call	Return Value	Explanation
`getCheck(159)`	2	The check digit for 159 is 2.
`isValid(1592)`	`true`	The number 1592 is a valid combination of a number (159) and its check digit (2).
`isValid(1593)`	`false`	The number 1593 is not a valid combination of a number (159) and its check digit (3) because 2 is the check digit for 159.

Complete method `isValid` below. Assume that `getCheck` works as specified, regardless of what you wrote in part (a). You must use `getCheck` appropriately to receive full credit.

```
/** Returns true if numWithCheckDigit is valid, or false
 *  otherwise, as described in part (b).
 *  Precondition: The number of digits in numWithCheckDigit is
 *  between two and seven, inclusive.
 *                numWithCheckDigit >= 0
 */
public static boolean isValid(int numWithCheckDigit)
```

Applying the Scoring Criteria

Apply the question scoring criteria first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

v) Array/collection access confusion (`[ ]` `get`)

w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

x) Local variables used but none declared

y) Destruction of persistent data (e.g., changing value referenced by parameter)

z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ < > <> ≠)
- `[ ]` vs. `( )` vs. `<>`
- = instead of == and vice versa
- `length`/`size` confusion for array, String, List, or ArrayList; with or without ( )
- Extraneous `[ ]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing ; where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `( )` on parameter-less method or constructor invocations
- Missing `( )` around `if` or `while` conditions

Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be* **unambiguously *inferred from context, for example, "ArayList" instead of "ArrayList". As a counterexample, note that if the code declares "`int G = 99, g = 0;`", then uses "`while (G < 10)`" instead of "`while (g < 10)`", the context does* **not** *allow for the reader to assume the use of the lower case variable.*

Scoring Guidelines for Question 1: Methods and Control Structures — 9 points

Learning Objectives: MOD-1.H CON-1.A CON-1.E CON-2.A CON-2.E

Canonical solution

(a) — 4 points

```
public static int getCheck(int num)
{
    int sum = 0;
    for (int i = 1; i <= getNumberOfDigits(num); i++)
    {
      sum += (8 - i) * getDigit(num, i);
    }
    return sum % 10;
}
```

(b) — 5 points

```
public static boolean isValid(int numWithCheckDigit)
{
    int check = numWithCheckDigit % 10;
    int num = numWithCheckDigit / 10;
    int newCheck = getCheck(num);
    if (check == newCheck)
    {
      return true;
    }
    else
    {
      return false;
    }
}
```

(a) `getCheck`

Scoring Criteria	Decision Rules	
1 Calls `getNumberOfDigits` and `getDigit`		1 point 3.A MOD-1.H
2 Calculates one partial sum	**Responses earn the point if they...** • Calculate the sum inside or outside the context of a loop.	1 point 3.C CON-1.A
3 Calculates all partial sums (*in the context of a loop, no bounds errors*)	**Responses earn the point if they...** • Use either a `for` or `while` loop.	1 point 3.C CON-2.E
4 Returns the last digit of the calculated sum as the check digit	**Responses still earn the point even if they...** • Calculate the sum incorrectly.	1 point 3.C CON-1.A
	Total for part (a)	**4 points**

(b) `isValid`

Scoring Criteria	Decision Rules	
5 Obtains check digit of `numWithCheckDigit`		1 point 3.C CON-1.A
6 Obtains number remaining in `numWithCheckDigit` after check digit removed		1 point 3.C CON-1.A
7 Calls `getCheck` on number without check digit		1 point 3.A MOD-1.H
8 Compares check digit of `numWithCheckDigit` and return value from `getCheck`	**Responses still earn the point even if they...** • Calculated check digit incorrectly; or • Calculated number without check digit incorrectly.	1 point 3.C CON-1.E
9 Returns `true` or `false` depending on the result of the previous comparison	**Responses still earn the point even if they...** • Do not use an if statement	1 point 3.C CON-2.A
	Total for part (b)	**5 points**
Question specific penalties		
None		
	Total for question 1	**9 points**

Question 3: Array/ArrayList

The `Gizmo` class represents `gadgets` that people purchase. Some `Gizmo` objects are electronic and others are not. A partial definition of the `Gizmo` class is shown below.

```
public class Gizmo
{
   /** Returns the name of the manufacturer of this Gizmo. */
   public String getMaker()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is electronic, and false
    *  otherwise.
    */
   public boolean isElectronic()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is equivalent to the Gizmo
    *  object represented by the
    *  parameter, and false otherwise.
    */
   public boolean equals(Object other)
   {
      /* implementation not shown */
   }

   // There may be instance variables, constructors, and methods not shown.
}
```

The `OnlinePurchaseManager` class manages a sequence of `Gizmo` objects that an individual has purchased from an online vendor. You will write two methods of the `OnlinePurchaseManager` class. A partial definition of the `OnlinePurchaseManager` class is shown below.

```
public class OnlinePurchaseManager
{
   /** An ArrayList of purchased Gizmo objects,
    *  instantiated in the constructor.
    */
   private ArrayList<Gizmo> purchases;

   /** Returns the number of purchased Gizmo objects that are electronic
    *  whose manufacturer is maker, as described in part (a).
    */
   public int countElectronicsByMaker(String maker)
   {
      /* to be implemented in part (a) */
   }
```

```
    /** Returns true if any pair of adjacent purchased Gizmo objects are
     *  equivalent, and false otherwise, as described in part (b).
     */
    public boolean hasAdjacentEqualPair()
    {
        /* to be implemented in part (b) */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

(a) Write the `countElectronicsByMaker` method. The method examines the `ArrayList` instance variable `purchases` to determine how many `Gizmo` objects purchased are electronic and are manufactured by `maker`.

Assume that the `OnlinePurchaseManager` object `opm` has been declared and initialized so that the `ArrayList purchases` contains `Gizmo` objects as represented in the following table.

Index in purchases	0	1	2	3	4	5
Value returned by method call `isElectronic()`	`true`	`false`	`true`	`false`	`true`	`false`
Value returned by method call `getMaker()`	`"ABC"`	`"ABC"`	`"XYZ"`	`"lmnop"`	`"ABC"`	`"ABC"`

The following table shows the value returned by some calls to `countElectronicsByMaker`.

Method Call	Return Value
`opm.countElectronicsByMaker("ABC")`	2
`opm.countElectronicsByMaker("lmnop")`	0
`opm.countElectronicsByMaker("XYZ")`	1
`opm.countElectronicsByMaker("QRP")`	0

Complete method `countElectronicsByMaker` below.

```
/** Returns the number of purchased Gizmo objects that are electronic and
 *  whose manufacturer is maker, as described in part (a).
 */
public int countElectronicsByMaker(String maker)
```

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user's purchase history to detect such occurrences and request confirmation.

Write the `hasAdjacentEqualPair` method. The method detects whether two adjacent `Gizmo` objects in `purchases` are equivalent, using the `equals` method of the `Gizmo` class. If an adjacent equivalent pair is found, the `hasAdjacentEqualPair` method returns `true`. If no such pair is found, or if `purchases` has fewer than two elements, the method returns `false`.

Complete method `hasAdjacentEqualPair` below.

```
/** Returns true if any pair of adjacent purchased Gizmo objects are
 *  equivalent, and false otherwise, as described in part (b).
 */
public boolean hasAdjacentEqualPair()
```

Scoring Guidelines for Question 3: Array/ArrayList — 9 points

Learning Objectives: MOD-1.G | CON-1.B | CON-1.H | CON-2.C | CON-2.J.B | VAR-1.E.B | VAR-2.D | VAR 2.E

Canonical solution

(a) — 4 points

```
public int countElectronicsByMarker(String maker)
{
    int result = 0;
    for (Gizmo g : purchases)
    {
      if (g.getMaker().equals(maker) && g.isElectronic())
      {
         result++;
      }
    }
    return result;
}
```

(b) — 5 points

```
public boolean hasAdjacentEqualPair()
{
    Gizmo g1 = purchases.get(0);
    for (int pos = 1; pos < purchases.size(); pos++)
    {
      Gizmo g2 = purchases.get(pos);
      if (g1 . equals(g2))
      {
         return true;
      }
      g1 = g2;
    }
    return false;
}
```

(a) `countElectronicsByMaker`

	Scoring Criteria	Decision Rules	
1	Accesses all elements of `purchases` (*no bounds errors*)	**Responses earn the point if they...** • Use a for, enhanced for, or while loop.	**1 point** 3.D VAR 2.E
2	Calls `isElectronic` to test whether or not a `Gizmo` object is electronic and `getMaker` to get the name of the manufacturer of a `Gizmo` object	**Responses earn the point if they...** • Call methods on any Gizmo object, regardless of whether this is in the context of a loop.	**1 point** 3.A MOD-1.G
3	Tests whether or not the maker of a `Gizmo` object matches `maker`, using an appropriate `String` comparison at least once, in the context of a loop		**1 point** 3.C VAR-1.E.b
4	Computes the number of elements that are electronic and made by `maker`	**Responses still earn the point even if they...** • Do not return the number of elements that are electronic and made by `maker`	**1 point** 3.C CON-1.B
		Total for part (a)	**4 points**

(b) `hasAdjacentEqualPair`

	Scoring Criteria	Decision Rules	
5	Determines whether the `ArrayList` contains at least two elements		**1 point** 3.D VAR-2.D
6	Accesses all necessary elements of `purchases` (*no bounds error*)		**1 point** 3.D CON-2.J.b
7	Accesses two adjacent elements of the `ArrayList` in the context of a comparison	**Responses still earn the point even if they...** • Do not access adjacent elements in the context of a loop.	**1 point** 3.D VAR 2.E
8	Compares two distinct elements of the `ArrayList` for equivalence		**1 point** 3.C CON-1.H
9	Returns `true` if at least one equivalent pair is found, and `false` otherwise or if the `ArrayList` has fewer than two elements	**Responses earn the point if they...** • Use a `for` or `while` loop to access the pairs of elements.	**1 point** 3.C CON-2.C
		Total for part (b)	**5 points**
	Question specific penalties		
	None		
		Total for question 3	**9 points**

Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position `index` (`0 <= index <= size`), moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	



IEP AND 504 ACCOMMODATIONS AND MODIFICATIONS FOR EARLY CHILDHOOD AND ELEMENTARY SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
Review of directions
Use of mnemonics
Have student restate information
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Use of a study carrel
Assistance in maintaining uncluttered space
Weekly home-school communication tools (notebook, daily log, phone calls or email messages)
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Use of manipulatives
No penalty for spelling errors or sloppy handwriting
Follow a routine/schedule
Alternate quiet and active time
Teach time management skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Adjusted assignment timelines
Visual daily schedule

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Untimed tests
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book tests
Read test and directions to student
Provide study outline prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator and/or number line for math tests
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS MODIFICATIONS:

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Have parent sign behavior chart
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

IEP AND 504 ACCOMODATIONS AND MODIFICATIONS FOR MIDDLE AND HIGH SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
A locker with adapted lock
Review of directions
Review sessions
Use of mnemonics
Have student restate information
Provision of notes or outlines
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Weekly or bi-weekly home-school communication
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Lab and math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of tape recorder for lectures
Use of Interactive notebook
Use of a scribe
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Ability to type all assignments
Follow a routine/schedule
Teach time management and organizational skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Daily check-in with case manager or special education teacher
Adjusted assignment timelines
Visual daily schedule

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book or open-note tests
Read test and directions to student
Provide study guides prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Pace long-term projects
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Allow outlining, instead of writing for an essay or major project
Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator for math tests
Film or video supplements in place of reading text
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands

Pass/no pass option
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS FOR STUDENTS:

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Accommodations and Modifications for ESL Students

Give tests orally
Give instructions/directions in writing and orally
Assign a peer tutor, same language or English speaking
Allow use of bilingual/translation dictionary
Provide summaries of text or study guides of particular chapters
Allow errors in speaking
Allow errors in writing
Accept writing in first language
Highlight key vocabulary
Reduce amount of work required
Assess comprehension through demonstration or other alternative means (gestures, drawings)
Rephrase questions, directions, and explanations
Reduce multiple choices on assessments
Provide study guides and/ or outlines
Shorten reading assignments
Allow extended time for test/project completion
Use of audio books
Use of graphic organizers
Use reduced text, so that print is not dense
Use visuals/pictures throughout lesson/unit/assignemt
Adapt homework to reflect language proficiency/home support
Adapt homework to reflect home support
Provide hands-on activities and explanations
Allow extended time to answer questions, and permit drawing, as an explanation
Use of translation dictionaries to locate words in the native language
Use anecdotal records and portfolio of work as forms of assessment
Use story retellings to assess comprehension

Springfield Public Schools

Cultivating compassionate and extraordinary learners!

What Does the Gifted and Talented Experience Look Like?

Springfield Public Schools

Mission Statement

The Gifted and Talented Program of the Springfield Public Schools aims to provide a meaningful and rigorous education that instills the value of lifelong learning and empowers all students to reach their full potential and become self-fulfilled creative leaders in accordance with the comprehensive standards of our state. We accomplish this by offering challenging, real-world experiences in a student-centered environment.

We recognize that students come from diverse socioeconomic, cultural, and ethnic backgrounds, and the mission of the gifted and talented program is to provide opportunities for students to explore their intellectual, creative, and artistic gifts and talents within the classroom and with beyond-the-classroom experiences. It is essential to provide diverse, appropriate, and ongoing learning experiences and environments that incorporate students' academic, psychological, and social needs. Our responsibility is to provide students with educational alternatives that teach, challenge, and expand their knowledge, while simultaneously stressing the development of independent and self-directed learners who continuously generate questions, analyze, synthesize, and evaluate information and ideas.
New Jersey Administrative Code 6A:8-3.1 defines students who are gifted and talented as those students who possess or demonstrate high levels of ability in one or more content areas when compared to their chronological peers in the local district and who require modification of their educational program if they are to achieve in accordance with their capabilities.

Goals:

To support our mission, we have established the following goals to guide our work:

- ➢ To infuse into the general education program a broad range of activities for high-level learning that will challenge all students to perform at advanced levels, and provide extended opportunities, resources, and support in particular areas where student interest or talent is demonstrated.

- ➢ To provide a continuum of gifted education services that will strengthen the total school program, offer enriching and challenging learning opportunities for all students, and provide specialized learning experiences

- ➢ To promote achievement of students' best potentials and increased ability.

Through a productive and collaborative relationship among the building principals, classroom teachers, and the Gifted and Talented teacher, our schools can create programming that will enhance the learning experience for every student and serve the unique and specialized needs of students currently demonstrating gifted behaviors.

Elements of the Schoolwide Enrichment Model, W.I.N (What I Need), differentiated instruction, advanced classes, or individual programs into a student's regular instruction program are integral to this overarching framework for gifted education in the Springfield School District.

Reference Board Policy and Regulation 2464 – Gifted and Talented Pupils (M)

Early Childhood Center
Edward V. Walton

Schoolwide Virtual Enrichment

All first and second grade students have access to the Gifted and Talented Virtual Classroom. This virtual classroom is a space for students to complete various challenges and enrichment opportunities at their own pace. Students can receive feedback and additional adaptable resources to meet their individual learning needs. The virtual classroom design expands accessibility to enrichment resources and allows students to choose learning activities according to their unique interests and abilities. The purpose of the activities is to encourage problem-solving, develop critical thinking

skills, and expand curiosity and imagination. Activities are aligned to the New Jersey Student Learning Standards.

Gifted and Talented Teacher as a Resource

The Gifted and Talented teacher serves as a resource to all teachers within Edward V. Walton School. Teachers seeking resources for their high-ability learners will receive additional support from the Gifted and Talented teacher in various ways. This support may include additional units of study, challenging resources, and student projects.

Elementary Schools
James Caldwell
Thelma L. Sandmeier

The goal is to engage small groups of students in intensive learning experiences through standards-based, rigorous, and relevant units of study that supplement, not supplant, the grade-level curriculum and support students' unique learning needs.

Enrichment Experience Grade 3

Beginning in third grade, the G&T teacher will cycle through all classrooms to work on problem-solving and enrichment tasks with students. Driven by assessment data, academic performance, student interest, and teacher observation, small groups of students will meet with the G&T teacher during W.I.N. (What I Need) periods throughout the week that the teacher is assigned to their classroom teacher. Groupings will be flexible and monitored to best meet each student's needs throughout the school year. Activities will include brain teasers, puzzles, strategic thinking, problem-solving, and STEM challenges.

Targeted Gifted and Talented Programing Grades 4-5

Students may become identified as Gifted and Talented in fourth grade. Students that are identified may receive support from the G&T Teacher for up to five periods (200 minutes) a week during W.I.N. (What I Need) periods. Identified students will meet to explore grade-level content while incorporating depth and complexity in a student-centered, problem-based environment. The Gifted and Talented Program aims to provide students with various advanced and challenging learning experiences across content areas to promote independent thinking skills, refine critical thinking and communication skills, cultivate joy in learning, and expand student interests through problem-solving. The instruction that students receive incorporates 21st Century Skills and advanced content that is challenging, hands-on, and relevant to the real world.

Strategies and Practices that Support Gifted and Talented Students

- Adjusted Pace of Lessons
- Inquiry-Based Instruction
- Higher-Order Thinking Activities
- Interest-Based Content
- Real World Problem Solving/Problem Based Learning
- Independent Study/Passion Projects
- Units of Study with Like-Minded Peers

Schoolwide Virtual Enrichment

All JCS and TLS students have access to grade level-specific Gifted and Talented Virtual Classrooms. These virtual classrooms are a space for students to complete various challenges and enrichment opportunities at their own pace. Students are able to receive feedback and additional opportunities based on their needs. The virtual classrooms have expanded the accessibility of enrichment resources directly to the students and allows for students to choose opportunities according to their interests and abilities. The purpose of the activities is to encourage problem solving, develop critical thinking skills, and expand curiosity and imagination in conjunction with the New Jersey Student Learning Standards.

Gifted and Talented Teacher as a Resource

In addition to classroom support, the Gifted and Talented teacher also serves as a resource to all teachers in the building. Teachers that are seeking resources for their high ability learners will receive additional support from the Gifted and Talented teacher in various ways. This may include additional units of study, challenging resources, student projects, and meeting with the G&T teacher as needed.

Additional Enrichment Opportunities

Across all grade levels 3-5, as recommended by the classroom teacher, students exhibiting a need for additional enrichment may have the opportunity to meet with the G&T teacher as the schedule allows. Groupings will be flexible and monitored to best meet the students' needs throughout the school year. Activities will include, brain teasers, puzzles, strategic thinking activities, problem-solving challenges, and STEM challenges. Students engage in an extended process of asking questions, using resources, and developing answers that lead to thinking about what and how they are learning. Students are held accountable for how they work and use their time by demonstrating their learning through products or presentations.

Students self-select to participate in individual and/or group projects based on interest and commitment. The classroom teacher and/or the enrichment teacher will help guide and monitor student exploration of chosen topics, facilitating their learning with resources and teaching of needed skills.

Secondary Schools
Florence M. Gaudineer Middle School
Jonathan Dayton High School

Grades 6-12: Access to specific content programs with higher levels of academic challenges (i.e. Pre-Algebra, Accelerated Communication Arts, and Advanced Placement Classes)

Dual Credit Courses provide students the opportunity to obtain high school credit, and college credit, simultaneously through partnerships with several institutions.

Project Lead the Way (PLTW) students become builders of technology by engaging in real world activities. Students are empowered as design thinkers collaborating on real world projects.

Gifted and Talented Identification Criteria

The Springfield Schools Gifted and Talented Program recognizes that students require differentiated learning experiences beyond the core curriculum being taught because of their advanced abilities in one or more content areas. These students may also have other cognitive qualities such as curiosity, creativity, motivation, leadership, and passion that require modifications to their academic experience.

Multiple Measures of Eligibility

Kindergarten through grade twelve identification process for gifted and talented students includes multiple measures. Multiple measures may include but are not limited to: achievement test scores; grades; student performance or products; intelligence testing; parent, student, and/or teacher recommendation; and other appropriate measures. The identification methodology is developmentally appropriate, non-discriminatory, and related to the programs and services offered by the Springfield School District. The identification procedures are reviewed annually.

According to N.J.A.C. 6A:8-3.1, all public school districts must have a board-approved gifted and talented identification process and provide services for identified students enrolled in the grades of that school district. The regulations require that identification and services be made available to students in grades K-12.

When students are recommended for the Gifted and Talented Program there are multiple measures that are considered to determine if services are necessary and appropriate. The assessments used are as follows:

- *Cognitive Abilities Test (CogAT)*
- *Classroom Performance/Student Portfolios*
- *DRA2 Level*
- *iReady Diagnostic Scores*
- *Gifted Indicators Checklist*
- *Start Strong/NJSLA Assessment Data*

The N.J.A.C. requires that students be compared to their chronological peers in the local school district. The New Jersey Department of Education has not established state-level criteria for identifying students who are gifted, such as mandated tests or assessments, grade point averages, or IQ scores. Local school districts must use multiple measures to identify students.

Procedures
https://www.straussesmay.com/seportal/Public/DistrictPolicy.aspx?policyid=2464&id=b8a33a9f505e4998a6db3fb8e1de778b

Gifted and Talented Program Identification Matrix

Minimum Score Required for Consideration: 14 Points

1 Point	2 Points	3 Points
Cogat Assessment Overall Stanine Score of a 7	Cogat Assessment Overall Stanine Score of an 8	Cogat Assessment Overall Stanine Score of a 9
iReady Math Assessment Just Below Current Grade Level	iReady Math Assessment On Grade Level	iReady Math Assessment ½ Year + Above Current Level
IReady ELA On Grade Level	IReady ELA ½ Year Above Current Level	IReady ELA One Full Grade Level + Above Current Level
NJSLA Math Approaching Expectations	NJSLA Math Met Expectations	NJSLA Math Exceed Expectations
NJSLA ELA Approaching Expectations	NJSLA ELA Met Expectations	NJSLA ELA Exceed Expectations
HOPE Teacher Rating Scale Raw Score: 11 - 29	HOPE Teacher Rating Scale Raw Score: 30 - 49	HOPE Teacher Rating Scale Raw Score: 50-66

Team meetings present the opportunity to review data and assess the educational needs of all students. Teacher provides examples of student work exhibiting above and beyond grade level expectations, creativity, leadership, critical and abstract thinking. Programs for gifted and talented students will be periodically evaluated for their continuing efficacy and adjusted accordingly. The parent of any student identified as gifted or talented shall be consulted regarding any program designed to address the student's particular needs.

Resource Links
https://riversideinsights.com/cognitive_abilities_test
https://www.curriculumassociates.com/programs/i-ready-learning
https://www.nj.gov/education/assessment/resources/
https://www.routledge.com/HOPE-Teacher-Rating-Scale-Involving-Teachers-in-Equitable-Identification/Gentry-Pereira-Peters-McIntosh-Fugate/p/book/9781618214522#:~:text=The%20HOPE%20Teacher%20Rating%20Scale,making%20it%20easy%20to%20use.

Gifted and talented student: The *"Strengthening Gifted and Talented Education Act"* defines a gifted and talented student as a "student who possesses or demonstrates a high level of ability in one or more content areas when compared to their chronological peers in the school district and who require modifications of their educational program if they are to achieve in accordance with their capabilities. "

Twice-exceptional (2E) students: A twice-exceptional (2E) student is defined as "a student who is both gifted and a student with a disability. These students may also be referred to as having dual exceptionalities or as being gifted with learning disabilities (GT/LD). This also applies to students who are gifted with ADHD or gifted with autism" (New Jersey Association for Gifted Children, *www.nagc.org).*

Instructional adaptation: An adjustment or modification to instruction enabling a student who is gifted and talented to participate in, benefit from, and demonstrate knowledge and application of the New Jersey Student Learning Standards in one or more content areas at the instructional level of the student, not just the student's grade level.

Identifying gifted and talented students: The focus of identification is to provide programming and services in which students' specific learning needs and potential are appropriately addressed and met. In every New Jersey school district, there are students who require modifications to the general education curriculum if they are to achieve in accordance with their abilities. By identifying a student's general intellectual ability, creativity, or specific academic area aptitudes, we are acknowledging that they need programming and services outside of the general education/grade level curriculum to advance their learning.

NJDOE Fast Facts

https://www.nj.gov/education/standards/gifted/

https://www.nj.gov/education/standards/gifted/legislation.shtml

Supplemental Resources/Programming for Gifted Students

Edward V. Walton

- PETS Program – Primary Education Thinking Skills
- Group Solutions/Group Solutions 2 – Cooperative Logic Activities
- Lollipop Logic
- Beginning Thinking Skills
- AIMS Education Programming

James Caldwell and Thelma L. Sandmeier

Math/ELA

- Project M3: Mentoring Mathematical Minds
- Junior Great Books
- AIMS Education Programming
- Borenson Hands on Equations
- BizWorld
- Perplexors
- Advanced Math Explorations grade 3-5
- Challenging Units for Gifted Learners (ELA and Math)
- Creative Curriculum Extenders
- Differentiated Lessons & Projects
- Challenge Math Grades 4+
- Differentiating Instruction with Menus (All Content Areas)
- Jacob's Ladder
- Genius Hour/Passion Project
- Spatial Reasoning
- Logic Puzzles, Brain Games, Analogies

STEM

- Engineering is Elementary
- Engineer Throughout the Year
- PBS Design Squad
- DefinedSTEM
- Project Lead the Way (PLTW)

Enrichment Events

Experience for all students to be enriched through events related to the curricular or extracurricular activities. Examples include after school clubs, PTA-sponsored events, leadership opportunities, visual and performing arts, and community service opportunities.