AP ® Computer Science A Syllabus

Curricular Requirements		Page(s)
CR1	The course teaches students to design and implement computer-based solutions to problems.	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
CR2a	The course teaches students to use and implement commonly used algorithms.	12
CR2b	The course teaches students to use commonly used data structures.	12
CR3	The course teaches students to select appropriate algorithms and data structures to solve problems.	3, 4, 5, 6, 7, 8, 9, 10, 11
CR4	The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.	2, 4, 10, 12
CR5	The course teaches students to use elements of the standard Java library from the AP Java subset in Appendix A of the AP Computer Science A Course Description.	4, 12
CR6	The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.	4, 10, 12, 13
CR7	The course teaches students to recognize the ethical and social implications of computer use.	12

Course Overview

Computer Facilities

Our classroom is also our lab—we find this to be very conducive to learning. Our computers are set up in rows facing the white board projector where most instruction takes place. Our lab and the labs around campus are managed and maintained by a full-time tech staff. The labs are very well kept and are functional 100% of the time. This course is on a tight schedule; any downtime during lab is extremely detrimental to student learning, as a minimum of 20 hours of course time is dedicated to hands-on labs.

Text

Barnes, David and Kölling, Michael. *Objects First with Java™: A Practical Introduction Using BlueJ, 5th Edition.* Pearson Education publishing as Prentice Hall, 2012.
http://www.bluej.org/objects-first/

Course Outline

Weeks 1 – 18 [Semester 1]

Weeks 1-3	**Objects and classes** Hands-on introduction to the concepts of objects, classes and methods without going into the details of Java syntax. First look at some source code. • Reading: *Objects First with Java™* Sections 1.1 – 1.15 (Objects and classes, Creating objects, Calling methods, Parameters, Data types, Multiple instances, State, What is in an object?, Java code, Object interaction, Source code, Return values, Objects as parameters) **[CR1]** • Chapter Lab Exercises: 1.1 – 1.36. During these exercises we will analyze the **figures** program, a simple drawing with some geometrical shapes; illustrates creation of objects, method calling, and parameters. We will also analyze the **house** program, an example using shape objects to draw a picture; introduces source code, Java syntax, and compilation. Finally, we will also analyze the **lab-classes project**, which is a simple example with classes of students, which is also used in chapter 2 and 8 again. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR4]** • Review: Vocabulary summary: object, class, method, parameter, signature, type, multiple

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR4 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.

	instances, state, method calling, source code, result. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 3-6	**Understanding class definitions** Open up class definitions and investigate how Java source code is written to create behavior of objects. We discuss how to define fields and implement methods. Here, we also introduce the first types of statements. • Reading: *Objects First with Java™* Sections 2.1 – 1.23 (Examining a class definition, class header, fields, constructors, and methods, accessor and mutator methods, printing from methods, scope highlighting, local variables, parameters, calling methods, expressions) **[CR1]** • Chapter Lab Exercises: 2.1 – 2.93. During these exercises we will work with the **lab-classes** project, which helps illustrate objects, fields, and methods. Next, we will work with the **ticket-machine** project, which is a simulation of a ticket vending machine for train tickets' introduces more about fields, constructors, accessor and mutator methods, parameters, and some simple statements. Finally, we will work on the **book-exercise** project which stores details of a book. It reinforces the constructs used in the ticket-machine example. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: object creation, field, comment, constructor, scope, lifetime, assignment, accessor method, mutator method, println, conditional, Boolean expression, local variable. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 7 - 10	**Object interaction** Discuss interaction of multiple objects. We see how objects can collaborate by invoking each other's methods to perform a common task. We also discuss how one object can create other objects. • Reading: *Objects First with Java™* Sections 3.1 – 3.15 (Abstraction and modularization, class and object diagrams, primitive and object types,

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

	string concatenation, modulo operator, objects creating objects, multiple constructors and methods, internal and external method calls, this keyboard, break points) **[CR1]** • Chapter Lab Exercises: 3.1 – 3.46. During these exercises we will work with the **clock-display** program which is an implementation of a display for a digital clock; illustrates the concepts of abstraction, modularization, and object interaction. Includes a version with an animated GUI. Next, we will work with the **mail-system** program which is a simulation of a simple email system. This is used to demonstrate object creation and interaction. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: abstraction, modularization, class and object diagram(s), object references, primitive type, object creation, overloading, internal and external method call(s), debugger. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Week 10	**Magpie Lab [4 hours]** Software Development Process; control flow (sequential and conditional); Boolean expressions, laws, and truth tables; using conditional expressions in if, if-else, and nested if statements; and operators (increment, decrement, compound assignment). • Reading: Magpie Introduction and Activities 1 – 4 (APCS A Labs) • Lab: Magpie Activities 1-4 (APCS A Labs). **[CR4] [CR6]**
Weeks 11 - 13	**Grouping Objects** Begin by creating more extensive structures of objects and using collections of objects. Introduce collections and ArrayLists. **[CR5]** We discuss iterations over collection and have a first look at loops, including for each and while. In the second half of the chapter we introduce arrays as a special form of a collection, and the for loop as another form of a loop. • Reading: *Objects First with Java™* Sections 4.1 – 4.17 (The collection abstraction, importing a library class **[CR5]**, diamond notation, key methods of ArrayList, object structures with

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

CR4 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.

CR5 - The course teaches students to use elements of the standard Java library from the AP Java subset in Appendix A of the AP Computer Science A Course Description.

CR6 - The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.

	collections, generic classes, numbering within collections, processing a whole collection, for-each loop, while loop, Iterator type, null keyword, anonymous objects, chaining method calls and using collections, merge sort) • Chapter Lab Exercises: 4.1 – 4.87. During these exercises we will work with the **music-organizer** project which is an implementation of an organizer for music tracks; used to introduce collections and loops. Includes the ability to play MP3 files. A GUI is added in Chapter 11. Next, we use the program **auction** which emulates an auction system and spends more time with collections and loops, this time with iterators. Next, we work with a program called **weblog-analyzer** which is a program that analyzes web access log files; introduces arrays and for loops. Finally, we will use merge sort by using the following web resource: http://goo.gl/njljLk **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: collection, loop, iterator, null, array, merge sort. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 14 - 15	**More-sophisticated behavior** Libraries and interfaces are introduced. We introduce the Java standard library and discuss some important library classes. More importantly, we explain how to read and understand the library documentation. The importance of writing documentation in software development projects is discussed, and we end by practicing how to write suitable documentation for our own classes. Random, Set and Map are examples of classes that we encounter in this chapter. • Reading: *Objects First with Java™* Sections 5.1 – 5.14 (Documentation for library classes, reading class documentation, interfaces versus implementation, using library-class methods, checking string equality, Random class, generator random responses, generating random numbers with limited ranges, using maps for associations, using a HashMap, using sets, dividing strings, writing class documentation, javadoc, public

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

	versus private, information hiding, class variables and constants, static keyword, constants) • Chapter Lab Exercises: 5.1 – 5.73. During these exercises we will work with the **tech-support** program which is an implementation of an *Eliza*-like dialog program to provide “technical support” to customers; introduces use of library classes in general and some specific classes in particular; reading and writing of documentation. Next, we look at the program, **scribble,** which is a shape-drawing program to support learning about classes from their interfaces. Finally, we work with the **bouncing-balls** program which is a graphical animation of bouncing balls; demonstrates interface/implementation separation and simple graphics. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: Java library, library documentation, interface, implementation, immutable, map, set, documentation, access modifier, information hiding, class variable, static variable. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam	CR1 - The course teaches students to design and implement computer-based solutions to problems. CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.
Weeks 16 – 17	**Designing classes** We discuss more formally the issues of dividing a problem domain into classes for implementation. We introduce issues of designing classes well, including concepts such as responsibility-driven design, coupling, cohesion, and refactoring. • Reading: *Objects First with Java™* Sections 6.1 – 6.16 (Making extensions, finding relevant source code, coupling, using encapsulation to reduce coupling, responsibility-driven design, localizing change, implicit coupling, cohesion of methods and classes, Switch statements, refactoring, executing without BlueJ) • Chapter Lab Exercises: 6.1 – 6.56. During these exercises we will work with the **world-of-zuul** program which is a text-based, interactive adventure game. It is highly extendable, makes a great open-ended student project. Used here to	

	discuss good class design, coupling, and cohesion. Used again in Chapter 9. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: coupling, cohesion, code duplication, encapsulation, responsibility-driven design, localizing change, method cohesion, class cohesion, refactoring, switch statement. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Week 18	**Well-behaved objects** This section deals with a whole group of issues connected to producing correct, understandable, and maintainable classes. It covers issues ranging from writing clear, understandable code - including style and commenting - to testing and debugging. Test strategies are introduced and a number of debugging methods are discussed in detail. • Reading: *Objects First with Java™* Sections 7.1 – 7.12 (Testing and debugging, using inspectors, test automation, regression testing, automated testing using JUnit, recording a test, fixtures, debugging, commenting and style, manual walk-throughs, choosing a debugging strategy) • Chapter Lab Exercises: 7.1 – 7.37. During these exercises we will work with the **online-shop** program which emulates the early stages of part of an online shopping website, dealing with user comments; used to discuss testing and debugging strategies. Next, we work with the **calculator** program which is an implementation of a desk calculator. This example reinforces concepts introduced earlier and is used to discuss testing and debugging. Finally, we work with the **bricks** program, which is a simple debugging exercise; models filling pallets with bricks for simple computations. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: testing, debugging, positive testing, assertion, fixture, walkthrough. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

Weeks 19 – 36 [Semester 2]

Weeks 19 – 21	**Improving structure with inheritance** Introduce inheritance and polymorphism with many of the related detailed issues. We discuss a part of the implementation of a social networking site to illustrate the concepts. Issues of code inheritance, subtyping, polymorphic method calls and overriding are discussed in detail. • Reading: *Objects First with Java™* Sections 8.1 – 8.11 (Using inheritance, inheritance hierarchies, access rights, initialization, subtyping, subclasses and subtypes, polymorphic variables, casting, Object class, autoboxing and wrapper classes, collection hierarchy) • Chapter Lab Exercises: 8.1 – 8.19. During these exercises we will work with the **network** program which is part of a social network application. This project is discussed and then extended in great detail to introduce the foundations of inheritance and polymorphism. Next, we come back to the **lab-classes** program to discuss inheritance. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: inheritance, superclass, subclass, inheritance hierarchy, superclass constructor, reuse, subtype, variables and subtypes, substitution, Object, autoboxing. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 22 - 24	**More about inheritance** Introduce inheritance and polymorphism with many of the related detailed issues. We discuss a part of the implementation of a social networking site to illustrate the concepts. Issues of code inheritance, subtyping, polymorphic method calls and overriding are discussed in detail. • Reading: *Objects First with Java™* Sections 9.1 – 9.12 (Static type and dynamic type, overriding, dynamic method lookup, super call in methods, method polymorphism, toString, equals and hashCode, protected access, instanceof operator)

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

	• Chapter Lab Exercises: 9.1 – 9.16. During these exercises we will work with the **network** program which is part of a social network application. This project is discussed and then extended in great detail to introduce the foundations of inheritance and polymorphism. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: static type, dynamic type, overriding, method polymorphism, toString, protected. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 25 - 26	**Further abstraction techniques** Implement a predator/prey simulation. This serves to discuss additional abstraction mechanisms based on inheritance, namely interfaces and abstract classes. • Reading: *Objects First with Java™* Sections 10.1 – 10.12 (Abstract classes and methods, multiple inheritance, flexibility through abstraction, interfaces as types, library support through abstract classes and interfaces, event-driven simulations) • Chapter Lab Exercises: 10.1 – 10.72. . During these exercises we will work with the **foxes-and-rabbits** program which is a classic predator-prey simulation; reinforces inheritance concepts and adds abstract classes and interfaces. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: abstract method, abstract class, abstract subclass, superclass method calls, multiple inheritance, and interface. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 27 – 28	**Building graphical user interfaces** This chapter introduces two new examples: an image viewer and a sound player. Both examples serve to discuss how to build graphical user interfaces (GUIs). • Reading: *Objects First with Java™* Sections 11.1 – 11.10 (Components, layout, and event handling, AWT and Swing, creating a frame, adding simple components, adding menus, event handling, receipt of events, inner classes, anonymous inner

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

	classes, adding images, nested containers, image filters, dialogs, improving program structure, buttons, borders, further extensions) • Chapter Lab Exercises: 11.1 – 11.79. During these exercises we will work with the **image-viewer** program which is a simple image view and manipulation application. We concentrate mainly on building the GUI. Next, we will work on the **music-player** program which adds a GUI to the music-organizer project from Chapter 4. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: components, layout, event handling, image format, menu bar, content pane, event listener, anonymous inner classes. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam
Weeks 29 - 31	**Pictures Lab [6 hours]** One- and two-dimensional arrays (creation, insertions, deletions, traversals, algorithms); searching algorithms and comparison (sequential and binary); and choosing appropriate data representation and algorithms. • Reading: PictureLab Introduction and Activities 1-9 (APCS A Labs) • Lab: Picture Lab Activities 1-9 (APCS A Labs) **[CR4] [CR6]**
Weeks 32 – 33	**Handling errors** This chapter discusses the difficult issue of how to deal with errors. Several possible problems and solutions are discussed, and JavaOs exception handling mechanism is discussed in detail. We extend and improve an address book application to illustrate the concepts. • Reading: *Objects First with Java™* Sections 12.1 – 12.10 (Defensive programming, client-server interaction, parameter checking, notifying the user, notifying the client object, exception-throwing principles, throwing an exception, checked and unchecked exceptions, preventing object creation, exception handling, defining new exception classes, using assertions, the assert statement, consistency checks, error recovery and avoidance, file-based input/output, readers, writers and streams, Scanner: parsing input)

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

CR4 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.

CR6 - The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.

	• Chapter Lab Exercises: 12.1 – 12.54. During these exercises we will work with the **address-book** program which is an implementation of an address book with an optional GUI interface. Lookup is flexible: entries can be searched by partial definition of name or phone number. This project makes extensive use of exceptions. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]** • Review: Vocabulary summary: exception, unchecked exception, checked exception, exception handler, assertion, serialization. Code review exercises as assigned by instructor. • Test: True/False, Multiple Choice Exam

CR1 - The course teaches students to design and implement computer-based solutions to problems.

CR3 - The course teaches students to select appropriate algorithms and data structures to solve problems.

Week 34	**Designing applications** This chapter steps back to discuss in more detail the next level of abstraction: how to structure a vaguely described problem into classes and methods. In previous chapters we have assumed that large parts of the application structure already exist, and we have made improvements. Now it is time to discuss how we can get started from a clean slate. This involves detailed discussion of what the classes should be that implement our application, how they interact, and how responsibilities should be distributed. We use class-responsibilities-collaborators (CRC) cards to approach this problem, while designing a cinema booking system. • Reading: *Objects First with Java™* Sections 13.1 – 13.8 (Analysis and design, verb/noun method, discovering classes, using CRC cards, class design, designing class interfaces, user interface design, documentation, cooperation, prototyping, software growth, using design patterns, decorator, singleton, Factory method, Observer) • Chapter Lab Exercises: 13.1 – 13.18. During these exercises we will work with the **cinema-booking-system** program, a system to manage advance seat bookings in a cinema. This example is used in a discussion of class discovery and application design. No code is provided, as the example represents the development of an application from a blank sheet of paper. **[CR1]** • Hand-written review code exercises as assigned by instructor. **[CR3]**

	• Review: Vocabulary summary: verb/noun, scenarios, prototyping, design pattern. Code review exercises as assigned by instructor. Test: True/False, Multiple Choice Exam
Weeks 35 - 36	**Elevens Lab [10 hours]** Review and more: Writing Classes, Lists and ArrayLists (creation, insertions, deletions, traversals, algorithms); **[CR2b] [CR5]** sorting algorithms and comparison (selection and insertion) **[CR2a] [CR3];** and choosing appropriate data representation and algorithms. **[CR3]** Inheritance, using class member, polymorphism and class hierarchy design. • Reading: Elevens Activities 1-11 (APCS A Labs) • Lab: Elevens Activities 1-11 (APCS A Labs) **[CR4] [CR6]**

CR2a - The course teaches students to use and implement commonly used algorithms.

CR2b - The course teaches students to use commonly used data structures.

CR4 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.

CR5 - The course teaches students to use elements of the standard Java library from the AP Java subset in Appendix A of the AP Computer Science A Course Description.

CR6 - The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.

CR7 - The course teaches students to recognize the ethical and social implications of computer use.

Teaching Strategies

General Approach

AP Computer Science A is a substantial course that requires a meticulous approach from both my students and me. The limited class time is fully utilized for discussion and activities, labs, quizzes, review, and multiple-choice tests. Outside reading and hand-written exercises are essential to the success of the students. Reading and comprehending technical material is a new skill for most students. They need to learn active reading techniques, including how to take notes, which will also be utilized by commenting notes next to new statements of code. I engage my students in a number of activities and discussions focused on the ethical and social implications of computer use such as protection of privacy, intellectual property, and public safety. I will introduce them to both the ACM and IEEE and their published Codes of Ethics. Dr. Jody Paul has an excellent site listing many resources that we also use to facilitate discussion and activities focused on computer ethics. **[CR7]**

I typically begin each new unit of material with reading, reinforcement exercises and hand-written assignments as well as review projects. This is followed by classroom discussion and related activities. Students complete review projects often during each chapter. Finally, I have a review and a multiple-choice test for each chapter's end. Reviews include Quizlet banks as well as review games.

References and Reading
Dr. Jody Paul http://jodypaul.com/sweng.html
ICSE_ISC_CBSE_Computer Science http://amitasuri.com/2011/04/23/merge-sort-program/

Differentiated Instruction
Different students learn in different ways. I use a variety of pedagogical teaching techniques including student presentations, group work, and various multiple response strategies to engage students. Students also learn at different rates, so I utilize tutoring, individual challenges, and extra credit assignments to address student needs at both ends of the spectrum.

Tutoring
In order to be successful in AP Computer Science A, it's critical that students learn the material in a timely fashion. Students who don't grasp earlier material don't have the foundation necessary for later material. In addition to making myself available before and after school, I offer time after class to help students who have a difficult time grasping certain concepts.

Extra Credit
It's important to keep all students engaged and learning, so I provide extra credit challenges and labs for students interested in this.

Review
After completing the course material, I give and review a practice AP Computer Science A exam. I then conduct a comprehensive review during the last few weeks prior to the AP Exam. Students work through the reading and multiple choice questions in Barron's AP Computer Science A as well as a bank of Multiple Choice and Free Response sample AP questions from the Leon Schram book. We then discuss this reading and multiple-choice questions in class.

Lab Component
Writing computer programs is critical to understanding the course material. I assign many labs and review projects per unit. These assignments are completed on an individual basis. Many students complete their lab assignments during class. I provide ample open lab time before and after school for students who need or want it.

I have integrated the AP Computer Science A Labs into my course at appropriate times based on their content, which account for a minimum of 20 hours of hands-on lab work (e.g., four hours on Magpie labs, six hours on PictureLab labs, and ten hours on the Elevens labs). **[CR6]** Students complete the Magpie labs to help them develop their conditional statement skills. They complete the PictureLab labs to practice two-dimensional array algorithms. I have distributed the Elevens

CR6 - The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.

labs across the school year to complement different portions of my object-oriented curriculum. Students complete all of the required activities of the AP Computer Science A Labs.

My lab computers have the Oracle Java SDK and the BlueJ Integrated, Interactive Development Environment, tailored for our use. All of the Java-specific software we use in the classroom is available at no cost. Since all software is installed on our district’s Virtual Machine servers, students may access their files and this software at any time on any device, which many students take advantage of.