Roanoke County Public Schools

Pre-AP Computer Programming (6640P)

Curriculum Guide

June 2015

Mathematics & CTE Curriculum Guide

Revised 2015. Available at www.rcs.k12.va.us.

Acknowledgements

The following people have made tremendous contributions to the completion of this curriculum guide and all are appreciated.

Phifer Herrala
William Byrd High School

Skip Larrington
Hidden Valley High School

Bob Powers
Cave Spring High School

Stephanie Schilling
Northside High School

Roanoke County Public Schools Administration

Dr. Lorraine Lange
Superintendent

Dr. Ken Nicely
Director of Secondary Instruction

Dr. Linda Wright
Director of Elementary Instruction

Linda Bowden
Mathematics Coordinator

Preface

This curriculum guide is written for the teachers to assist them in using the textbooks/resources in a most effective way. This guide will assist the mathematics teacher in preparing students for the challenges of the twenty-first century. As established by the National Council of Teachers of Mathematics Principles and Standards for School Mathematics, educational goals for students are changing. Students should have many and varied experiences in their mathematical training to help them learn to value mathematics, become confident in their ability to do mathematics, become problem solvers, and learn to communicate and reason mathematically. This guide, along with the available textbook resources, other professional literature, alternative assessment methods, and varied instruction in-service activities will assist the mathematics teacher in continuing to integrate these student goals into the curriculum.

Table of Contents

Introduction/General Comments

1. As always, the instructor should exercise their own judgment on what exercises or activities to use from this curriculum guide as well as their judgment when it comes to the pacing. It is suggested that the instructor spends ample time on loops as it is a difficult concept for students to master. Students MUST be familiar with a while loop, and standard for loop, and an enhanced for loop (also known as a for-each loop).
2. The concept behind this guide is that the instructor moves through the sections as outlined in the guide. They have the option of assigning the exercises listed in the guide, assigning others from the back of the chapters, or finding other assignments from other texts. Plenty of exercises are listed in the back of each chapter.
3. As you look through the guide, you will notice that there are no SOL guidelines noted and the last page which discusses the SOL blueprints is notably blank. The reason for this is because there are no SOL guidelines for computer programming classes.
4. This course is specifically designed as a precursor to the AP Computer Science class. This class will introduce many of the basic concepts from the AP Computer Science class, but will not cover them in as much depth as they will need to be covered for the AP exam.
5. The Magpie Chatbot Lab is suggested by the College Board as foundational material for the AP exam. It is highly recommended.
6. When students have completed this class, they should have a solid foundation for the upcoming AP Computer Science Class.
7. The instructor may find that some programs seem redundant, or that time is limited and they want to move on. It is perfectly acceptable to skip some programming assignments that are listed here. If this is done, the instructor should keep a record of the programs that are skipped. These would constitute an excellent review of these concepts when planning the AP class for the following year.
8. The instructor might find that certain programs they feel are excellent projects that combine all the concepts of a chapter are missing from this curriculum guide. This is intentional! These programs are left for the AP Computer Science curriculum. The AP Computer Science curriculum will use these programs during the first nine weeks to review and enhance understanding of the material in the six chapters covered in this class.

Textbook/Resources Overview

Book Title: Java Software Solutions for AP* Computer Science (Third Edition)

Authors: John Lewis, William Loftus, Cara Cocking

Publisher: Pearson

Year: 2011

This book is specifically designed to allow a student to prepare for the AP Computer Science Exam. It is not intended to be strictly an introductory Java textbook. For this reason, this material can be supplemented with other material of an introductory nature from other textbooks.

The textbook comes with a companion website that the instructor is urged to register with. The website contains a variety of video lessons, bonus chapters, and source code to utilize with the class. The website can be found at http://www.pearsonschool.com/Access_Request and select "access to online instructor resources."

Sequence of Instruction and Pacing Suggestions

First Nine Weeks		
SOL	**Chapter/Sections/Topic**	***Time Frame**
	Chapter 1 – Computer Systems	4.00 blocks
	Chapter 2 – Objects and Primitive Data	16.00 blocks
	Chapter 3 - Program Statements	2.50 blocks
***Time Frame is based on 95 minutes of instruction per block.**	**First Nine Weeks Total**	**22.5 blocks**

Second Nine Weeks		
SOL	**Chapter/Sections/Topic**	***Time Frame**
	Chapter 3 - Program Statements	11.50 blocks
	Chapter 4 – Writing Classes	11.00 blocks
***Time Frame is based on 95 minutes of instruction per block.**	**Second Nine Weeks Total**	**22.5 blocks**

Sequence of Instruction and Pacing Suggestions

Third Nine Weeks		
SOL	**Chapter/Sections/Topic**	***Time Frame**
	Chapter 4 – Writing Classes	14.00 blocks
	Chapter 5 – Enhancing Classes	8.50 blocks
***Time Frame is based on 95 minutes of instruction per block.**	**Third Nine Weeks Total**	**22.5 blocks**

Fourth Nine Weeks		
SOL	**Chapter/Sections/Topic**	***Time Frame**
	Chapter 5 – Enhancing Classes	2.50 blocks
	Chapter 6 – Arrays (Including sorting and searching)	15.0 blocks
	Magpie Chatbot Lab	4.00 blocks
	Final Exam	1.00 block
***Time Frame is based on 95 minutes of instruction per block.**	**Fourth Nine Weeks Total**	**22.5 blocks**

Mapping for Instruction - First Nine Weeks

Chapter: 1 Computer Systems

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
1.0: Introduction	Short Answer: 1.2, 1.3	All references to problems are from the textbook unless stated otherwise.
1.1: Hardware Components	Short Answer: 1.1, 1.4	
1.2: Networks	Short Answer: 1.5, 1.6	
1.3: Programming	Short Answer: 1.7, 1.8 Programming Projects: 1.1, 1.2, 1.3, 1.4, 1.5, 1.6	
1.4: Programming Languages	Short Answer: 1.9	

Chapter: 2 Objects and Primitive Data

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
2.0 An Introduction to Objects 2.1 Using Objects	Supplemental Programs: 2.1, 2.2, 2.3	Supplemental Programs are attached as part of this curriculum guide.
2.2 String Literals 2.3 Variables and Assignment 2.4 Primitive Data Types 2.5 Arithmetic Expressions 2.6 Enumerated Types	Programming Projects: 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7 Supplemental Programs: Painting a Room	Programming Projects are listed at the end of every chapter in the book.
2.7 Creating Objects 2.8 Class Libraries and Packages 2.9 Interactive Programs 2.10 Formatting Output	Programming Projects: 2.8, 2.9, 2.10, 2.11, 2.12, 2.13	Always have students use Math.random() since this is what is used on the AP exam. Note: Program 2.14 is an excellent project for the purpose of putting everything in this chapter together; however, it is intentionally omitted from this class as it will be used as a review in the AP class..

Chapter: 3 Program Statements

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
3.0 Program Development 3.1 Control Flow 3.2 The if Statement	Programming Projects: 3.1, 3.2	Cover DeMorgan's Law during this unit.

Mapping for Instruction - Second Nine Weeks

Chapter: 3 Program Statements

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
3.3 Boolean Expressions Revisited 3.4 More Operators	Supplemental Programs: Rock, Paper, Scissors	
3.5 The while Statement	Programming Projects: 3.3	Make sure to cover the while statement. The do-while statement is optional.
3.6 Iterators 3.7 The for Statement	Programming Projects: 3.4, 3.5, 3.6, 3.7, 3.8, 3.9	Note that this now includes the for-each loop which is also referred to as the enhanced for loop.
3.8 Program Development Revisited	Programming Projects 3.10, 3.11, 3.12, 3.13, 3.15	Note that program 3.14 is intentionally omitted so it can be used as a review program in the AP Computer Science class.

Chapter: 4 Writing Classes - Part 1

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
4.0 Objects Revisited 4.1 Anatomy of a Class	Programming Projects 4.1, 4.2	
4.2 Anatomy of a Method	Programming Projects 4.3, 4.4,	Notice that programming project 4.3 allows the students to use the Die class from the book. Some teachers may choose to interactively write the Die class with their students in class. This will help the students learn the structure of a class. Also note that program 4.5 will be used as a review program in the AP class and is intentionally omitted from this section.

Mapping for Instruction - Third Nine Weeks

Chapter: 4 Writing Classes - Continued

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
4.3 Method Overloading	Programming Projects 4.6, 4.8	
4.4 Method Decomposition	Programming Projects 4.4	
4.5 Object Relationships	Programming Projects 4.8 Supplemental Programs: Tracking Grades, Band Booster Class	Cover an “is-a” relationship versus a “has a” relationship.

Chapter: 5 Enhancing Classes

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
5.0 References Revisited 5.1 The static Modifier	Programming Projects: 5.1	Discuss both static methods and static variables.
5.2 Exceptions		Students need to recognize exceptions, not throw them.
5.3 Interfaces	Programming Projects: 5.2, 5.3, 5.4	Students need to be able to recognize and interface. Students should also be able to create their own interfaces.

Mapping for Instruction - Fourth Nine Weeks

Chapter: 5 Enhancing Classes

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
5.3 Interfaces	Programming Projects: 5.5, 5.6	Project 5.6 is complex. Make sure you have done this first before asking your students to do it.
5.4 Identifying Classes and Objects		Discuss white box testing, black box testing, and debugging.

Chapter: 6 Arrays

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
6.0 Arrays	Programming Projects: 6.1, 6.2, 6.4, 6.5	It is entirely possible that some of the assignments in this chapter might be put off until the AP Computer Science class. It all depends on the students in the class. If that is done, the teacher will have to cover them in the first part of the AP class.
6.1 Arrays of Objects	Programming Projects: 6.6, 6.7, 6.8, 6.9, 6.10	
6.2 Searching	Supplemental Programs: Search Projects	
6.3 Sorting	Supplemental Programs: Sort Projects	
6.4 Comparing Sorts	Supplemental Programs: Comparing Sorts Project	This section may be moved completely to the AP class if necessary.

Magpie Chatbot Lab

Textbook Chapters/Sections/Topics	Supporting Materials	Comments
Strings / Random Numbers / Conditional Statements / Arrays	Magpie Chatbot Lab – Activities 1-5	This lab is provided in its entirety from AP Central at College Board. Teachers may choose to make modifications to the lab and/or the software as desired. Teachers may choose to modify the lab in an effort to make it easier to work with in Eclipse.

Supplemental Resources

Chapter 1 - Computer Systems – Supplemental Materials

Chapter Notes – Relation to AP subset:

Students must know how to count in binary, both forward and backward.

They must understand the difference between a compiler and an interpreter and how Java fits into that picture. They also need to understand that Eclipse is neither, but is an Integrated Development Environment (IDE). They must understand the concept of Java byteCode.

From a programming perspective, students must understand how to output data to the screen (System.out.println).

They need to understand how to write and use comments. Comments can be designated in many ways such as // or /* */ or /** */. Javadoc comments @param and @return are part of the AP subset.

They need to understand the difference between syntax and semantics.

They also need to understand the difference between compile-time, run-time, and logic errors.

Supplemental Resources:

Some teachers use a video by Mark Gugnor about Men's Brains / Women's Brains. It can be found on YouTube by searching for the author and title or by going here: https://www.youtube.com/watch?v=0BxckAMaTDc. There are many different versions, the 5 ½ minute version is sufficient to make the point of how men and women think differently, and it is very important to have both type of thinking in the workplace.

Some teachers reference trailers for the One Million Minutes (or 2 or 4) video to demonstrate global competition and diversity. Some are: https://www.youtube.com/watch?v=vFQ6j-BPwTs or https://www.youtube.com/watch?v=9bFmWlyBcrs.

Chapter 1 - Activity 1

History Project

Some teachers use a history project to meet some of the CTE competencies as well as give the students experience researching a topic and presenting on it.

There are many people / topics that can be researched. The list below is not an exhaustive list, so feel free to add to it. Also, depending on the size of the class, you may choose to remove some of the items from the list.

People	Equipment / Innovations	Languages / Software	Personal Computers
Charles Babbage	Abacus	Machine Language	Intel 8080
Ada Byron Lovelace	Napier's Bones	Assemblers	Altair
Herman Hollerith	Slide Rule	Compilers	Apple I
Vannevar Bush	Pascaline Engine	Interpreters	Apple II
Tommy Flowers	Stepped Rekoner	Basic	Macintosh
Alan Turing	Calculator (les Xavier de Colmar)	Fortran	IBM PC
Claude Shannon	Vacuum Tubes	COBOL	Compaq Portable PC
George Stibitz	Atanasoff / Berry Computer	C, C+, C++, C#	Gateway
William Hewlett & David Packard	Colossus Computer	Pascal	Dell
Konrad Zuse	Mark I	Java	
Grace Hopper	ENIAC		
John Mauchly	Transistor	LINIX / UNIX	Machine Components
Von Neumann	UNIVAC I	MS DOS	Core Memory
Paul Baran	Semi-Conductor	Windows	IC Memory
Ray Tomlinson	Integrated Circuit	Mac OS	Disk Drives (Floppy, hard, SSD)
Bob Metcalfe	IBM 360		Tape Drives
Vint Cerf & Bob Kahn	Cray I	VisiCalc	Keyboards, Mice, Touch Screens
Richard Stallman	Hayes Modem	WordStar	CPU / ALU / Registers
Larry Page & Sergey Brin		WordPerfect	CDs/ DVDs
Jimmy Wales & Larry Sanger		Word / Excel / PowerPoint	Paper Tape / Punch Cards
		TurboTax	Bus / Serial / Parallel ports

Chapter 1 - Activity 2

Communication Activity

Arrange students so that they paired up but they cannot see one another's desks using boxes or some other item. There are two different activities that the students can participate in. One is a drawing activity while the other is a building activity. For the drawing activity, one student will have a plain sheet of paper and a pencil/pen. The other student will have a picture of basic shapes. Neither student can see the other student's paper. Only the student with the picture is allowed to speak. The other partner has to follow the instructions they are given. The student with the picture will describe in detail to the other student how to draw the picture without telling the student what the final picture is supposed to look like.

Similarly, this activity can be done using legos/k'nex. One student would have a picture of something to build while the other student has the exact number of parts needed to build what¹s in the picture. As before, only the student with the picture can talk and neither student can see what the other student has.

Purpose of this activity - this is to mimic what it is like for a student to talk to a computer. The computer will follow the directions as given and will not be able to ask questions. This forces the students to be very detail oriented and they quickly see how order of commands and how you say something matters.

NOTE - You may want to contact Stephanie Schilling or Phifer Herrala for some of the pictures they have their students draw.

Chapter 1 - Activity 3

Introduction to Algorithms

Have the students write down directions for how to make a peanut butter and jelly sandwich. The students are not to write their names on this page as this is a participation activity and not for an accuracy grade. You can either pick at random or find the worst ones and then begin to follow the algorithm to make a peanut butter and jelly sandwich following their instructions literally.

Talk to the students about how there are a variety of solutions to this problem. However, some of the solutions are better than the others. It's the same with writing algorithms for code.

Now, have the students again write down the instructions for making a peanut butter and jelly sandwich. Display a few of them on the board to demonstrate that there are multiple ways to solve the problem.

Purpose of this activity - students will realize how important the little details are. They cannot make any assumptions when writing code.

For fun - show the students the friendship algorithm clip from The Big Bang Theory. (this can easily be found on YouTube) This is a good pictorial demonstration of a flowchart.

Chapter 2 - Objects and Primitive Data - Supplemental Materials

Chapter Notes – Relation to AP subset:

Students are responsible for 3 primitive data types: int, double, and boolean. While the AP subset ignores chars, stating that they might confuse students because of their close relationships to Strings, the book uses them extensively. Also, if this confusion issue were true, then the same issue exists for ints versus Integers and doubles versus Doubles. Because knowledge of the char data type can be extremely useful when answering free response questions on the AP exam, it is recommended that the students be extremely familiar with the char primitive data type. Warning - the AP people like to test the substring method heavily. They almost always have a question that walks off the end of the String because the ending position in a substring method is always confusing. Also, they like to ask unique things about the substring method. Compare the following three things ... what do you expect to happen? By the way, try them - what you get may not be what you would expect.

```
String myString = new String("abcde");

String s1 = myString.substring(myString.length()-1);

String s2 = myString.substring(myString.length());

String s3 = myString.substring(myString.length()+1);
```

Students must understand operator precedence. Casting should be covered in this chapter, especially as related to ints and doubles. This is often a question on the AP exam and usually relates to how averages are calculated (see page 73 of the textbook). Enumerated types are new to the AP subset and must be covered here. Students MUST understand truncation towards zero when casting from a double to an int (i.e. 4.5 → 4 and -4.5 → -4). Students need to know how to round instead of truncate (i.e. if (x>0) (int)(x + 0.5)) else (int)(x - 0.5))

Students need to understand String literals, String concatenation, and the String methods listed on page 78 of the textbook. They need to understand constants and variables, and how to manipulate variables. They must understand the main arithmetic operators (+, -, /, *, and %). They must understand the dual nature of the plus sign(+). They must understand escape sequences for use in Strings, highlighted on page 60 of the textbook.

The Integer and Double classes (the Objects!) are covered in this chapter. While auto boxing and unboxing are not part of the AP subset, they should be covered since this is something the students will run into when they are testing casts. This can be very confusing as something is going on behind the scenes that is not intuitively obvious to the students. This is an excellent time to discuss class libraries and packages. Refer to the book to know which ones the students have to be familiar with (Integer, Double, String, Math, etc.) Note: the Random class is NOT part of the AP subject. The Math class is! Suggestion - use Math.random() and not

Random.nextInt(). The NumberFormat class and DecimalFormat class are not part of the AP subset.

Note: user input (Scanner class) is NOT tested on the AP Exam.

When creating objects, students MUST use the keyword "new". This is a common error on the AP Exam and is **ALWAYS** tested.

Students must understand and be familiar with the terminology of a "method signature". The method signature depends on the number, types, and order of its parameters ONLY, and does not include the return type. This terminology is frequently used on the AP Exam.

In the AP subset, **ALMOST ALL CLASSES SHOULD BE DECLARED AS PUBLIC**. In the AP subset, **ALL INSTANCE VARIABLES SHOULD BE DEFINED AS PRIVATE**. Methods, constants (static final variables), and constructors may be either public or private.

Students need to understand how to make a variable into a constant (final).

Static variables are part of the subset. Using static methods is also part of the subset (i.e. Math.random()).

The rules for default initialization are not in the AP subset.

Supplemental Programs:

Something that is used by a number of teachers is www.codingbat.com. This includes hundreds of practice problems for students (done like puzzles) that challenge students to solve a problem. If their answer is incorrect, the computer tells them what test cases they did not prepare for and gives them the opportunity to try again. This program also tracks, by student, their completion rate and success. The assignments given can correlate to the chapters the students are studying.

See other resources below.

Chapter 2 - Activity 1

Computer Corporation Activity

CTE Skills List: 1 – 29 (with regards to the simulation); other skills apply depending on the programming project(s) assigned

Creating the company

Students will work in small groups of two to three people. The group will come up with a name for their corporation, a mission statement, where their headquarters are located, as well as stats for their company. Within the group, they will assign the following roles to every person (one person may hold more than one role): Chief Executive Officer (CEO), Chief Financial Officer (CFO) and Chief Information Officer (CIO). Each role has a specific function with relation to the client, which is being role-played by the teacher. The CEO is the only one who can make bids with the client. The CFO is only the one who can make purchases for the company and authorize any financial transaction. The CIO is only the one who can ask questions about the specification and/or client modifications as well as turns in the program to the client. After the company is set up, the teacher verifies the information and should everything check out, the company is allowed to make bids.

Making bids

The company simply looks through the chapter exercises and makes bids on the various programs to do. The goal for each chapter is a set amount of money, adjustable by the teacher. The baseline cost for each program, also set by the teacher, can be adjusted accordingly to difficulty level (i.e., harder programs are worth more, easier ones are cheaper). When the CEO for the company is making the bid for the chapter exercise, the teacher/client can role-play the interactions with the student and depending on their interactions, can affect the value of the bid and/or whether the bid is even accepted.

Turning in projects

CIOs are responsible for turning in the bids to the client. Again, teachers role-play the client and test the program for any bugs and whether or not they fulfill the original specifications. Should the program meet with the client's satisfaction, the client can award them the agreed upon amount in the initial bid. Should the project not meet the client's specification or requirements, the client may have the options of: a) rejecting the attempt and ask the company to fix the mistakes; b) accept the attempt however award them less than the stated amount of the initial bid; or c) completely deny the bid. No money is awarded and the company loses the bid.

Accumulated Revenue

The money that is earned by the company is accumulated and can be used in various ways in this simulation. First, and foremost, it can be used as a grade for the individuals in the company. Reaching a set goal, or a percentage thereof, can determine a grade. Secondly, the revenue that the companies have accrued can be used to spend on various aspects of their simulation. For example, they may purchase computer equipment or hire lawyers to boost various stats with respect to their company. You'll find that some students may think outside of the box and try to use their revenue in different ways. It's entirely up to the teacher on whether or not it plays in the simulation.

Contact

This activity was provided by Bob Powers from Cave Spring High School. It is an involved activity, but well worth the time and effort. If you would like more information on this activity, please contact him directly.

Names and Places

The goal of this exercise is to develop a program that will print a list of student names together with other information for each. The tab character (an escape sequence) is helpful in getting the list to line up nicely. A program with only two names is in the file Names.java.

a. Save Names.java to your directory. Compile and run it to see how it works.

b. Modify the program so that your name and hometown and the name and hometown of at least two classmates sitting near you in class are also printed. Save, compile and run the program. Make sure the columns line up.

c. Modify the program to add a third column with the intended major of each person (assume Sally's major is Computer Science and Alexander's major is Math). Be sure to add a label at the top of the third column and be sure everything is lined up (use tab characters!).

Chapter 2 - Activity 3

Two Meanings of Plus

In Java, the symbol + can be used to add numbers or to concatenate Strings. This exercise illustrates both uses. When using a String literal (a sequence of characters enclosed in double quotation marks) the complete String must fit on one line. The following is NOT legal (it would result in a compile-time error).

```
System.out.println ("It is NOT okay to go to the next line
                    in a LONG string!!!");
```

The solution is to break the long String up into two shorter Strings that are joined using the concatenation operator (which is the + symbol). This is discussed in Section 2.2 in the text. The following would be legal:

```
System.out.println ("It is OKAY to break a long string into " +
                    "parts and join them with a + symbol.");
```

When working with Strings, the + symbol means to concatenate the Strings (join them). When working with numbers the + means what it has always meant -- add! Remember operator precedence. Work from left to right. This can impact whether the “+” sign means add or concatenate.

1. Observing the Behavior of +.

To see the behavior of + in different settings do the following:

a. Study the program below, which is in file PlusTest.java.

```
// ********************************************************************
//   PlusTest.java
//
//   Demonstrate the different behaviors of the + operator
// ********************************************************************

public class PlusTest
{
   // -----------------------------------------------
   // main prints some expressions using the + operator
   // -----------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("This is a long String that is the " +
                          "concatenation of two shorter Strings.");

      System.out.println ("The first computer was invented about" + 55 +
                          "years ago.");

      System.out.println ("8 plus 5 is " + 8 + 5);

      System.out.println ("8 plus 5 is " + (8 + 5));

      System.out.println (8 + 5 + " equals 8 plus 5.");
      }
```

```
}
```

b. Save PlusTest.java to your directory.

c. Compile and run the program. For each of the last three output statements (the ones dealing with 8 plus 5) write down what was printed. Now for each explain why the computer printed what it did given that the following rules are used for +. Write your explanations on one of the programs you have already printed out.

If both operands are numbers + is treated as ordinary addition. (NOTE: in the expression a + b the a and b are called the operands.)

If at least one operand is a String, the other operand is converted to a String and + is the concatenation operator.

If an expression contains more than one operation, expressions inside parentheses are evaluated first. If there are no parentheses, the expression is evaluated left to right.

d. The statement about when the computer was invented is too scrunched up. How should that be fixed?

2. Writing Your Own Program With +.

Now write a complete Java program that prints out the following sentence:

Ten robins plus 13 canaries is 23 birds.

Your program must use only one statement that invokes the println method. It must use the + operator both to do arithmetic and String concatenation. In other words, there must be a 10+13 somewhere in your println statement.

Chapter 2 - Activity 4

A Table of Student Grades

Write a Java program that prints a table with a list of at least 5 students together with their grades earned (lab points, bonus points, and the total) in the format below.

```
//////////////////\\\\\\\\\\\\\\\\\\
==          Student Points          ==
\\\\\\\\\\\\\\\\\\//////////////////


Name            Lab     Bonus   Total
----            ---     -----   -----
Joe             43      7       50
William         50      8       58
Mary Sue        39      10      49
```

The requirements for the program are as follows:

1. Print the border on the top as illustrated (using the slash and backslash characters).
2. Use tab characters to get your columns aligned. You **MUST** use the + operator both for addition and String concatenation.
3. Make up your own student names and points -- the ones shown are just for illustration purposes. You need 5 names.

Painting a Room

File Paint.java contains the partial program below which, when complete, will calculate the amount of paint needed to paint the walls of a room of the given length, width, and height. It assumes that the paint covers 350 square feet per gallon. Use the Scanner class to read in values from the keyboard.

```
//*************************************************************
//File: Paint.java
//
//Purpose: Determine how much paint is needed to paint the walls
//of a room given its length, width, and height
//*************************************************************
public class Paint
{
   public static void main(String[] args)
   {
        final int COVERAGE = 350;  //paint covers 350 sq ft/gal
       //declare integers length, width, and height;
       //declare double totalSqFt;
       //declare double paintNeeded;

       //Prompt for and read in the length of the room

       //Prompt for and read in the width of the room
```

```
        //Prompt for and read in the height of the room


        //Compute the total square feet to be painted—think about the dimensions of each wall


        //Compute the amount of paint needed


        //Print the length, width, and height of the room and the number of gallons of paint needed.


    }
}
```

Save this file to your directory and do the following:

1. Fill in the missing statements (the comments tell you where to fill in) so that the program does what it is supposed to. Compile and run the program and correct any errors.

2. Suppose the room has doors and windows that don't need painting. Ask the user to enter the number of doors and number of windows in the room, and adjust the total square feet to be painted accordingly. Assume that each door is 20 square feet and each window is 15 square feet.

Chapter 3 - Program Statements - Supplemental Materials

Chapter Notes – Relation to AP subset:

Flowcharting is an excellent concept to cover. While flowcharting is not part of the AP subset, it is very helpful when trying to understand a process.

In this chapter, students must master the if statement, if-else statement, nested if-else statements, and block statements (statements within curly braces {}). They must understand the difference between assignment operators and relational operators. They must understand how to use relational operators and their precedence (and, or, not). They must be able to create and read truth tables. They need to understand when to use “=” versus "==" versus ".equals". They need to understand the inherent problems with comparing floating point numbers for equality, and learn to use tolerances instead. Increment operators, decrement operators, and special java assignment operators must be understood (page 135).

Note that students might run into the following problem. In the early days of Java, the statement String s1 = new String(“abc”); was functionally equivalent to String s1 = “abc”:. This is no longer true.

```
String s1 = new String("abc");
String s2 = new String("abc");
if (s1==s2)
      System.out.println ("This does not print");
String s3 = "abc";
String s4 = "abc";
if (s3==s4)
   System.out.println ("This does print");
```

It is imperative that students understand the concept of short-circuiting. It almost always appears on the AP Exam in some form (either an “and” or an “or” relational operator can short circuit). Also ensure that students understand DeMorgan’s law. It also appears every year somewhere in the AP exam.

Students must learn how to use the while statement. Note that the do-while statement is NOT part of the AP subset although it might be helpful to introduce the concept. They must understand the for loop and the enhanced for loop (also called the for-each loop). They must understand when to use, and when NOT to use, each of the looping constructs mentioned above. They must understand infinite loops and nested loops. They must learn about iterators, something that is new to the AP subset in the past couple of years, and how to use these with loops. The break and continue statements are NOT part of the AP subset.

Supplemental Programs: See below.

Chapter 3 - Activity 1

Rock, Paper, Scissors Program

Program Rock.java contains a skeleton for the game Rock, Paper, Scissors. Open it and save it to your directory. Add statements to the program as indicated by the comments so that the program asks the user to enter a play, generates a random play for the computer, compares them, and announces the winner (and why). Some teachers may choose to not give the students the skeleton but make them create it on their own.

For example, one run of your program might look like this:

Enter your play: R, P, S, or Q to quit

r

Computer play is S

Rock crushes scissors, you win!

Note that the user should be able to enter either upper or lower case r, p, s, and q. The user's play is stored as a String to make it easy to convert whatever is entered to upper case. Use an if-else statement to convert the randomly generated int for the computer's play to a String or to convert the character the user entered into an int. If the user enters an invalid letter, tell them that their entry was invalid and force them to enter a valid character.

This program can be enhanced to address more of the CTE competencies by keeping score and reporting the number or wins, losses, and ties at the end of the game (or along the way).

Chapter 3 - Activity 2

Pig Program

Using the PairOfDice class, design and implement a class to play a game called Pig.

In this game, the user competes against the computer. (I found it easier to write this as a two person game first, and then extend it to make one of the players the computer.)

Players alternate turns (I call them rounds) until one player reaches 100 points. The first player to 100 points wins the game. The game should stop IMMEDIATELY when either player earns 100 points.

Each player's round is made up of one or more rolls of the dice. You may assume the user rolls at least one time. The player rolls the pair of dice and adds up their points.

- If the player rolls snake eyes (two 1's), they lose all the points they have earned so far in the game (their "gameScore"), they lose all the points they have earned so far this round (their "roundScore"), and the dice go to the other player.
- If the player rolls a 1 on either of the two die (but not both), they lose all their points for this round (their "roundScore"), their total for the game so far (their "gameScore") remains the same, and the dice go to the other player.
- If the player does not roll any 1's, then the points they received on the pair of dice are added to their score for this round ("roundScore").
    - If the player's total points for the game so far ("roundScore" plus "gameScore") is 100 or greater, the game immediately ends and this player is declared the winner.
    - If the player's total points for the game so far ("roundScore plus "gameScore") are less than 100 points, then the player has a choice to make. The player is given the option of continuing to roll (being a pig), or passing the dice to the other player. If the player passes the dice to the other player, their "roundScore" is added to their "gameScore", and their "roundScore" is reset to zero.

The first player to 100 wins. (In other words, play stops automatically when either player gets to 100!)

When you decide to make this a human against a computer, the only thing you have to change is in the third bullet above. It has to do with whether the computer chooses to pass the dice or not. Assuming the computer's round has not ended because they rolled either a single or double 1, then the computer has to choose whether to pass the dice or not. Here is how you make that choice –

- If the computer has earned less than 20 points on this round, they roll again (pig)
- If the computer has earned 20 or more points on this round, then the round ends and the computer gives the dice to the player.

Chapter 4 - Writing Classes - Supplemental Materials

Chapter Notes – Relation to AP subset:

This chapter focuses on writing classes. Students need to be very skilled at writing classes, defining instance data, and writing methods. They need to be extremely clear on the difference between a class, an object, instance data, and methods. They need to understand the concept of private versus public data (protected data is not covered in the AP subset). They also need to understand the difference between public and private methods, and when to use each. The use of the return statement (for a non-void method) and the passing of parameters needs to be stressed. This is something that students often miss on the AP Exam since almost every free response question has to do with writing a class, a method, or both. The difference between a formal parameter and the actual parameter is often confusing. The concept of constructors needs to be hit hard. Students should hear the following statement almost every day: "The purpose of a constructor is to initialize the instance variables." Overloaded methods and method decomposition (breaking a large method into smaller ones) need to be learned here. Also, having objects made up of other objects should be dealt with here (a student object contains an Address object which contains String objects).

Supplemental Programs: See below.

Chapter 4 - Activity 1

Tracking Grades Program

A teacher wants a program to keep track of grades for students and decides to create a Student class for this program as follows:

Each Student will be described by three pieces of data: name, score on test #1, and score on test #2.

There will be one constructor, which will have one argument -- the name of the Student.

There will be three methods: printName, which will print the student's name; inputGrades, which will prompt for and read in the student's test grades; and getAverage, which will compute and return the student's average as a double.

1. File Student.java contains an incomplete definition for the Student class. Save it to your directory and complete the class definition as follows:

a. Declare the instance data (name, score for test1, and score for test2).

b. Add the missing method headers.

c. Add the missing method bodies.

2. File Grades.java contains a shell program that declares two Student objects. Save it to your directory and use the inputGrades method to read in each student's test scores, then use the getAverage method to find their average. Print the average with the student's name, e.g., "The average for Joe is 87.5" You can use the printName method to print the student's name.

3. The printName method is rather cumbersome for producing the output described above. Add a method getName to your Student class that, instead of printing the name, returns it as a String. Now modify your Grades program to use getName instead of printName in printing each student's average.

4. Modify the inputGrades method of the Student class so that it validates the grades it reads in. That is, if a grade entered is less than 0 or greater than 100, it should print a warning message and ask for another grade. This should be repeated (for each grade entered) until the grade is between 0 and 100. Note that this means you can't use an if statement, but you will need to use a while statement.

5. Add statements to your Grades program that print the values of your Student variables directly, e.g.:

```
System.out.println("Student 1: " + student1);
```

This should compile, but notice what it does when you run it -- nothing very useful! When an Object is printed, Java looks for a toString method for that Object. This method must have no parameters and must return a String. If such a method has been defined for this object, it is called and the String it returns is printed. Otherwise the default toString method, which is inherited from the Object class, is called. It simply returns a unique hexadecimal identifier (the address of the Object) for the Object such as the ones you saw above.

Add a toString method to your Student class that returns a String containing the student's name and test scores, e.g.:

Name: Joe Test1: 85 Test2: 91

Note that the toString method does not call System.out.println -- it just returns a String.

Recompile your Student class and the Grades program (you shouldn't have to change the Grades program -- you don't have to call toString explicitly). Now see what happens when you print a student Object -- much nicer!

Chapter 4 - Activity 1

Band Booster Class

In this exercise, you will write a class that models a band booster and use your class to update sales of band candy.

1. Write the BandBooster class assuming a BandBooster Object is described by two pieces of instance data: name (a String) and boxesSold (an int that represents the number of boxes of band candy the booster has sold in the band fundraiser). The class should have the following methods:
   - A constructor that has one parameter -- a String containing the name of the BandBooster. The constructor should set boxesSold to 0.
   - A method getName that returns the name of the BandBooster (it has no parameters).
   - A method updateSales that takes a single int parameter representing the number of additional boxes of candy sold. The method should add this number to boxesSold.
   - A toString method that returns a String containing the name of the BandBooster and the number of boxes of candy sold in a format similar to the following:
   - Joe: 16 boxes
2. Write a program that uses BandBooster objects to track the sales of 3 boosters over several weeks. Your program should do the following:
   - Read in the names of the three band boosters and construct an Object for each.
   - Read in the number of weeks for the current fundraising campaign.
   - Have a count controlled loop that, for each week, gets the number of boxes of candy sold by each booster. Your prompts should include the booster's name. For example,

   Enter the number of boxes sold by Joe this week:
   - For each member, after reading in the weekly sales, invoke the updateSales method to update the total sales by that member.
   - After the loop, print the name and total sales for each member (you will implicitly use the toString method here).

Chapter 5 - Enhancing Classes - Supplemental Materials

Chapter Notes – Relation to AP subset:

Using "null" needs to be discussed in detail. The use of automatically created "getters" and "setters" can be used to help with this concept. Creating stub methods that return null for the object as a temporary fix can be useful. The use of the word “this” does not need to be discussed as it is not part of the subset; however, if you automatically generate getters and setters, then an explanation if “this” will be required as it is used in the setter methods.

Static variables need to be learned here.

Understanding an exception is an important concept that must be grasped. The students DO NOT need to be able to throw an exception - they only need to understand the concept behind one and why one might want to "catch" one. They DO NOT need to use the "try" and "catch" features of Java.

Using an Interface is another important concept that must be grasped. Students are expected to implement an Interface. They are NOT expected to create one. Implementing the Comparable Interface, Interator, and ListIterator Interfaces is expected.

Nested classes are covered in this chapter. Nested classes are NOT part of the AP subset.

Supplemental Programs:

There are no supplemental programs for this chapter.

Chapter 6 - Arrays - Supplemental Materials

Chapter Notes – Relation to AP subset:

This is an absolutely HUGE chapter when it comes to the AP Exam - probably the most important chapter in the book. Arrays and ArrayLists are hit repeatedly on the AP Exam. At this point, the Die class must implement the Comparable interface (if you have not done this, do it now). Students need to be fully conversant with one dimensional arrays, two dimensional arrays, and ArrayLists. They need to know when to use them and how to access their elements. They need to fully understand how to use a for loop, and a for-each loop, to look at the data. They also need to be very aware of when to use which type of loop (for example, you can't remove an element from a list if using the for-each loop). One big thing the AP people like to test is where array subscripts start and end. Watch this one carefully!

Students need to be able to create an array using an initializer list (i.e. int[] foo = {2,5,8};)

Students need to be able to pass arrays (or ArrayLists) as parameters and determine the length (size) of them inside the methods. It is especially important that they are able to determine the number of rows and/or the number of columns in a two dimensional array inside a method! They need to keep track of whether an array is full or potentially partially empty. Note that all two dimensional arrays in the AP subset are rectangular. There are no ragged arrays in the subset, although they are completely permissible in Java.

Students need to be fully conversant with Selection Sorts, and Insertion Sorts. They need to know what happens in each pass of the sorts, and be able to recognize a sorting type from the appearance of the data. (Note: Merge Sort will be covered in the recursion unit.)

Supplemental Programs: See below.

Chapter 6 - Activity 1

BlackJack Program

Write a program that implements BlackJack. This program must use a Card Class, Deck class, and BlackJack Player class.

Write this assuming that an Ace is always worth 11 points. Eventually, you may enhance this to allow a 1 or 11.

Write this assuming that you have three players (last one being the dealer). Eventually, allow the user to choose the number of players.

Write this showing all cards of all players. Eventually, hide the first card dealt.

Write this as a text based program. Eventually, add graphics to the program using Java or Greenfoot.

Chapter 6 - Activity 2

Searching and Sorting Program:

The following are all portions of a large program. The methods are all static and are called from a main program that changes many times for testing purposes.

Program a – Create a random int array

Design and implement a static method called createIntArray that takes three parameters – the size of an array, the lowest value of the range, and the highest value of the range. This method will create an array of that size containing random ints between the starting and ending values of the range (inclusive). Return this array to the calling method. If any parameters are invalid, return null. Design and implement a second static method called printIntArray that takes one parameter – the array – and prints the array, 1 int per line. Create a main program that creates an array and prints it using the two methods above.

Program b – Linear Search for an int in the above array

Design and implement a static method called intLinearSearch that takes an array of int values as a parameter and also takes a single int as a second parameter. This method should perform a linear search of the unsorted array to see if the number can be found. If found, return the value to the user of the location of the found number. If not found, indicate this by returning -1. Create a main program that tests this by prompting the user for which item to search for in an array of your choosing.

Program c – Bubble Sort (Optional – easy sort to understand and code; however, not tested on the AP exam).

Design and implement a static method called intBubbleSort that takes an array of int values as a parameter. This method should perform a bubble sort on the array. Test this by creating and printing an array, sorting it, and printing it again.

Program d – Selection Sort

Design and implement a static method called intSelectionSort that takes an array of int values as a parameter. This method should perform a selection sort on the array. Test your method.

Program e – Insertion Sort

Design and implement a static method called intInsertionSort that takes an array of int values as a parameter. This method should perform an insertion sort on the array. Test your method.

Program f – Binary Search for an int in the above array

Design and implement a static method called intBinarySearch that takes an array of sorted int values as a parameter and also takes a single int as a second parameter. This method should perform a binary search of the sorted array to see if the number can be found. If found, return the value to the user of the location of the found number. If not found, indicate this by returning -1. Test your method.

Program g – Repeat steps a, c, d, e, and f for Die

Design and implement static methods called createDieArray, printDieArray, dieBubbleSort, dieSelectionSort, and dieInsertionSort. Note that the createDieArray will only take two parameters, the size of the array and the number of sides on each Die. For example, you could create an array of 10 Die where each Die had 1,000 sides. dieBinarySearch that takes an array of Die as a parameter and also takes a single int as a second parameter.

Program h – Compare Searches

Using the methods and programs developed above, compare the speed of searches above. Use the following code to help determine speed. Even if you use a billion items, you may have to do something similar using the Die class to get meaningful differences in the numbers.

```
long startTimeMs = System.currentTimeMillis( );
// do your search or sort here
long taskTimeMs  = System.currentTimeMillis( ) - startTimeMs;
    long milliseconds = taskTimeMs % 1000;
    long seconds = taskTimeMs / 1000 % 60;
    long minutes = taskTimeMs / 60000 % 60;
    long hours = taskTimeMs / 3600000 % 24;
    System.out.println(       hours + " hours, " +
            minutes + " minutes, " +
            seconds + " seconds, " +
            milliseconds + " milliseconds");
```

Program i – Compare Sorts

Using the methods and programs developed above, compare the speed of sorts above. Use the above code to help determine speed. Start with 1,000 ints and double each time. You can then extrapolate for higher numbers if necessary. You must also try this for the Die class, but your numbers will not have to be as high for valid comparisons. Again, start at 1,000 Die.

CTE Correlation

Number	Competency	Section / Activity
6640.001	Demonstrate positive work ethic.	Every Project, all year. Computer Corporation Activity.
6640.002	Demonstrate integrity.	Discuss Day 2 of class. Maintain focus on every project. Computer Corporation Activity.
6640.003	Demonstrate teamwork skills.	Discuss Day 1 of class. Virtually all projects are group projects. Computer Corporation Activity.
6640.004	Demonstrate self-representation skills.	Chapter 1.2: History Project. Computer Corporation Activity.
6640.005	Demonstrate diversity awareness.	Discuss as part of “Men’s Brains, Women’s Brains” video. Computer Corporation Activity.
6640.006	Demonstrate conflict-resolution skills.	Discuss Day 1 of class. Virtually all projects are group projects that require collaboration and conflict resolution. Computer Corporation Activity.
6640.007	Demonstrate creativity and resourcefulness.	Chapters 4 & 6: Pig and Blackjack Projects. Computer Corporation Activity, Magpie Lab.
6640.008	Demonstrate effective speaking and listening skills.	Chapter 1.2: History Project. Computer Corporation Activity.
6640.009	Demonstrate effective reading and writing skills.	Chapter 1.2: History Project. Computer Corporation Activity, Magpie Lab.
6640.010	Demonstrate critical-thinking and problem-solving skills.	Chapter 1.3: Required for all projects. Computer Corporation Activity, Magpie Lab.
6640.011	Demonstrate healthy behaviors and safety skills.	Discuss Week 1. Computer Corporation Activity.
6640.012	Demonstrate an understanding of workplace organizations, systems, and climates.	Chapter 1.2: Discuss as part of jobs in the Computer Industry. Computer Corporation Activity, Magpie Lab.
6640.013	Demonstrate lifelong-learning skills.	Chapter 1.2: Discuss as part of jobs in the Computer Industry. Computer Corporation Activity, Magpie Lab.
6640.014	Demonstrate job-acquisition and advancement skills.	Chapter 1.2: Discuss as part of jobs in the Computer Industry. Computer Corporation Activity, Magpie Lab.
6640.015	Demonstrate time-, task-, and resource-management skills.	Discuss weeks 1 & 2. Required in all Projects. Computer Corporation Activity, Magpie Lab.
6640.016	Demonstrate job-specific mathematics skills.	Chapter 2.5. Computer Corporation Activity.
6640.017	Demonstrate customer-service skills.	Chapter 1.3: Company Profile Project. Computer Corporation

		Activity, Magpie Lab.
6640.018	Demonstrate proficiency with technologies common to a specific occupation.	Chapter 1.3: Company Profile Project. Computer Corporation Activity, Magpie Lab.
6640.019	Demonstrate information technology skills.	Chapter 1.3 and continue throughout the year. Computer Corporation Activity, Magpie Lab.
6640.020	Demonstrate an understanding of Internet use and security issues.	Chapter 1.2. Computer Corporation Activity.
6640.021	Demonstrate telecommunications skills.	Chapter 1.3: Company Profile Project. Computer Corporation Activity.
6640.022	Examine aspects of planning within an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity, Magpie Lab.
6640.023	Examine aspects of management within an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity.
6640.024	Examine aspects of financial responsibility within an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity.
6640.025	Examine technical and production skills required of workers within an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity, Magpie Lab.
6640.026	Examine principles of technology that underlie an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity, Magpie Lab.
6640.027	Examine labor issues related to an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity.
6640.028	Examine community issues related to an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity.
6640.029	Examine health, safety, and environmental issues related to an industry/organization.	Chapter 1.3: Company Profile Project. Computer Corporation Activity.
6640.030	Identify the purposes and goals of the student organization.	Week 1 & 2: Discuss as part of the study of involvement in clubs / organizations with the school and the work place
6640.031	Explain the benefits and responsibilities of membership in the student organization as a student and in professional/civic organizations as an adult.	Week 1 & 2: Discuss as part of the study of involvement in clubs / organizations with the school and the work place
6640.032	Demonstrate leadership skills through participation in student organization activities, such as meetings, programs, and projects.	Week 1 & 2: Discuss as part of the study of involvement in clubs / organizations with the school and the work place
6640.033	Identify Internet safety issues and procedures for complying with acceptable use standards.	Chapter 1.2
6640.034	Describe the development of computers and current industry trends in the programming field.	Chapter 1.4, History Project, Magpie Lab.

6640.035	Describe the development of programming languages and applications.	Chapter 1.4, History Project.
6640.036	Describe the functions of computer hardware, computer software, and computer system components.	Chapter 1.0, 1.1, History Project.
6640.037	Compare computer operating systems.	Chapter 1.0, 1.4, History Project.
6640.038	Identify the software development life cycle (SDLC).	Chapter 1.3, 1.4, Magpie Lab.
6640.039	Describe the development environment for a specific programming language.	Week 2: Discuss as part of Eclipse IDE, Magpie Lab.
6640.040	Analyze the problem statement.	Chapter 1.3 and continues throughout the course, Introduction to Algorithms Activity, Magpie Lab.
6640.041	Create possible solutions to the problem.	Chapter 1.3 and continues throughout the course, Introduction to Algorithms Activity, Magpie Lab.
6640.042	Determine the best solution to the problem.	Chapter 6.4 and continues throughout the course, Introduction to Algorithms Activity, Magpie Lab.
6640.043	Design a program, using an algorithm, pseudocode, a flowchart, and/or a decision table.	Chapter 3.1 and continues throughout the course, Magpie Lab.
6640.044	Code the program, using a programming language.	Chapter 1.3 and continues throughout the course, Magpie Lab.
6640.045	Test the program with sample data.	Chapter 2.2 and continues throughout the course, Magpie Lab.
6640.046	Debug the program.	Chapter 2.2 and continues throughout the course, Magpie Lab.
6640.047	Document the program.	Chapter 1.3 and continues throughout the course, Magpie Lab.
6640.048	Describe maintenance procedures.	Chapter 1.4 and in Dice programs, Magpie Lab.
6640.049	Identify syntax errors of a given programming language.	Chapter 1.4 and continues throughout the course, Magpie Lab.
6640.050	Identify industry standards for a graphical user interface (GUI).	Chapter 2.11. Also discuss with Hour of Code, Alice, and Greenfoot. Also discuss as part of Excel, Word, PowerPoint, etc.
6640.051	Create a graphical user interface that adheres to industry standards.	Alice, Scratch, Code.org, Greenfoot, or Visual Basic
6640.052	Code a program that will produce formatted output.	Chapter 2.10, Magpie Lab.
6640.053	Code an application that uses mathematical operations and built-in functions.	Chapter 2.8 and repeated throughout the course, Magpie Lab.

6640.054	Write a program that uses variables and constants.	Chapter 2.3 and repeated throughout the course, Magpie Lab.
6640.055	Write a program that accepts user input.	Chapter 2.9 and repeated throughout the course, Magpie Lab.
6640.056	Write a program that uses arrays.	Chapter 6.0 and repeated throughout the course, Magpie Lab.
6640.057	Write a modular program that uses functions or methods.	Chapter 2.1 and repeated throughout the course, Magpie Lab.
6640.058	Write a program that uses conditional structures.	Chapter 3.2 and repeated throughout the course, Magpie Lab.
6640.059	Write a program that uses looping structures.	Chapter 3.7 and repeated throughout the course, Magpie Lab.
6640.060	Write a program that uses counters and accumulators.	Chapter 3.5 and repeated throughout the course
6640.061	Identify the purpose of an executable file.	Chapter 1.4: Discuss with Interpreters versus Compilers
6640.062	Create an object-oriented program.	Chapter 2.0 and repeated throughout the course, Magpie Lab.
6640.063	Code a program to display graphics.	Chapter 1.5, Alice, Greenfoot, Visual Basic
6640.064	Code a program to incorporate multimedia.	Chapter 3,9, Alice, Python
6640.065	Code a program to animate objects.	Chapter 4.7, Code.org
6640.066	Examine the history of game design and development.	Chapter 4 and 6. Discuss once students have completed writing some of their own games.
6640.067	Analyze the impact of intellectual property law on game design.	Chapter 4 and 6. Discuss once students have completed writing some of their own games.
6640.068	Identify the target markets for game applications.	Chapter 4 and 6. Discuss once students have completed writing some of their own games, Magpie Lab.
6640.069	Identify game genres.	Chapter 4 and 6. Discuss once students have completed writing some of their own games.
6640.070	Examine a variety of game programming platforms.	Alice, Scratch, Code.org, Greenfoot, or Visual Basic
6640.071	Create a storyboard.	Do as part of 70 above
6640.072	Code a program from the storyboard.	Alice, Scratch, Code.org, Greenfoot, or Visual Basic
6640.073	Create an object within the context of a game.	Pig, BlackJack
6640.074	Specify behaviors of an object within the context of a game.	Pig, BlackJack
6640.075	Develop a game program that uses a scoring method.	Pig, BlackJack

6640.076	Create a game program with multiple levels.	Pig, BlackJack, Magpie Lab.
6640.077	Explain how to locate resources and references to aid program development.	Chapter 2.8: API for Java
6640.078	Evaluate the validity of sample code obtained from the Internet and/or other sources.	Tic-Tac_Toe: Pull From Internet and Test, Magpie Lab.
6640.079	Develop a Web page, using HTML and/or JavaScript. (Optional)	N/A
6640.080	Publish a program link on a Web page. (Optional)	N/A
6640.081	Describe the process and requirements for obtaining industry certifications related to the Programming course. (Optional)	N/A
6640.082	Identify testing skills/strategies for a certification examination. (Optional)	N/A
6640.083	Demonstrate ability to successfully complete selected practice examinations (e.g., practice questions similar to those on certification exams). (Optional)	N/A
6640.084	Successfully complete an industry certification examination representative of skills learned in this course (e.g., MCP, IC3, NOCTI). (Optional)	N/A
6640.085	Identify careers in the information technology industry.	Chapter 1.2: Discuss as part of jobs in the Computer Industry, Magpie Lab.
6640.086	Describe ways that computer programs can be used in business and industry.	Chapter 1.2: Discuss as part of jobs in the Computer Industry, Magpie Lab.
6640.087	Create or update a résumé.	Chapter 1.2: Discuss as part of jobs in the Computer Industry
6640.088	Investigate information technology educational and job opportunities.	Chapter 1.2: Discuss as part of jobs in the Computer Industry
6640.089	Assemble a professional portfolio.	This is done automatically by Eclipse. Students will save all their projects on a regular basis throughout the year culminating with the final portfolio at the end of the year
6640.090	Describe basic employment activities.	Chapter 1.2: Discuss as part of jobs in the Computer Industry
6640.091	Deliver an oral presentation of the professional portfolio. (Optional)	N/A
6640.092	Identify potential education and employment barriers for nontraditional groups and ways to overcome those barriers.	Chapter 1.2: Discuss as part of jobs in the Computer Industry

The Student Competency Record is given on the next few pages for reference. The most recent version should be downloaded annually by the teacher from: http://www.cteresource.org/verso/courses/6640/programming-scrs

2015/2016 Student Competency Record

Programming
6640 - 36 weeks

Student	**School Year**
School	**Teacher Signature**

Traditional letter or numerical grades do not provide adequate documentation of student achievement in competency-based education; therefore, the Virginia Standards for CBE require a recording system to provide information about competencies achieved to employer, student-employee, and teacher. The Student Competency Record provides a means for keeping track of student progress. Ratings are assigned by the teacher for classroom competency achievement and by the teacher-coordinator in conjunction with the training sponsor when competence is evaluated on the job.

Tasks/competencies designated "Required" are considered essential statewide and are required of all students. In some courses, all tasks/competencies have been identified as required. Tasks/competencies marked "Optional" are considered optional; they and/or additional tasks/competencies may be taught at the discretion of the school division. Tasks/competencies marked with an asterisk (*) are considered sensitive, and teachers should obtain approval by the school division before teaching them.

Note: Students with an Individualized Education Program (IEP) or an Individualized Student Alternative Education Plan (ISAEP) will be rated, using the following scale, only on the competencies identified in their IEP or ISAEP.

Students will be expected to achieve a **satisfactory rating** (one of the three highest marks) on the Student Competency Record (SCR) rating scale on at least 80% of the required (essential) competencies in a CTE course.

...RATING SCALE...
1 - Can teach others
2 - Can perform without supervision
3 - Can perform with limited supervision
4 - Can perform with supervision
5 - Cannot perform

6640 36 weeks		Programming TASKS/COMPETENCIES	Date	Rating
	Demonstrating Workplace Readiness Skills: Personal Qualities and People Skills			
Required	1	Demonstrate positive work ethic.		
Required	2	Demonstrate integrity.		
Required	3	Demonstrate teamwork skills.		
Required	4	Demonstrate self-representation skills.		
Required	5	Demonstrate diversity awareness.		
Required	6	Demonstrate conflict-resolution skills.		
Required	7	Demonstrate creativity and resourcefulness.		
	Demonstrating Workplace Readiness Skills: Professional Knowledge and Skills			

6640 36 weeks	Programming TASKS/COMPETENCIES		Date	Rating
Required	8	Demonstrate effective speaking and listening skills.		
Required	9	Demonstrate effective reading and writing skills.		
Required	10	Demonstrate critical-thinking and problem-solving skills.		
Required	11	Demonstrate healthy behaviors and safety skills.		
Required	12	Demonstrate an understanding of workplace organizations, systems, and climates.		
Required	13	Demonstrate lifelong-learning skills.		
Required	14	Demonstrate job-acquisition and advancement skills.		
Required	15	Demonstrate time-, task-, and resource-management skills.		
Required	16	Demonstrate job-specific mathematics skills.		
Required	17	Demonstrate customer-service skills.		
	Demonstrating Workplace Readiness Skills: Technology Knowledge and Skills			
Required	18	Demonstrate proficiency with technologies common to a specific occupation.		
Required	19	Demonstrate information technology skills.		
Required	20	Demonstrate an understanding of Internet use and security issues.		
Required	21	Demonstrate telecommunications skills.		
	Examining All Aspects of an Industry			
Required	22	Examine aspects of planning within an industry/organization.		
Required	23	Examine aspects of management within an industry/organization.		
Required	24	Examine aspects of financial responsibility within an industry/organization.		
Required	25	Examine technical and production skills required of workers within an industry/organization.		
Required	26	Examine principles of technology that underlie an industry/organization.		
Required	27	Examine labor issues related to an industry/organization.		
Required	28	Examine community issues related to an industry/organization.		
Required	29	Examine health, safety, and environmental issues related to an industry/organization.		
	Addressing Elements of Student Life			
Required	30	Identify the purposes and goals of the student organization.		
Required	31	Explain the benefits and responsibilities of membership in the student organization as a student and in professional/civic organizations as an adult.		
Required	32	Demonstrate leadership skills through participation in student organization activities, such as meetings, programs, and projects.		
Required	33	Identify Internet safety issues and procedures for complying with acceptable use standards.		

6640 36 weeks	Programming TASKS/COMPETENCIES		Date	Rating
	Exploring Programming Concepts			
Required	34	Describe the development of computers and current industry trends in the programming field.		
Required	35	Describe the development of programming languages and applications.		
Required	36	Describe the functions of computer hardware, computer software, and computer system components.		
Required	37	Compare computer operating systems.		
Required	38	Identify the software development life cycle (SDLC).		
Required	39	Describe the development environment for a specific programming language.		
	Using Algorithmic Procedures			
Required	40	Analyze the problem statement.		
Required	41	Create possible solutions to the problem.		
Required	42	Determine the best solution to the problem.		
	Implementing Programming Procedures			
Required	43	Design a program, using an algorithm, pseudocode, a flowchart, and/or a decision table.		
Required	44	Code the program, using a programming language.		
Required	45	Test the program with sample data.		
Required	46	Debug the program.		
Required	47	Document the program.		
Required	48	Describe maintenance procedures.		
	Mastering Programming Fundamentals			
Required	49	Identify syntax errors of a given programming language.		
Required	50	Identify industry standards for a graphical user interface (GUI).		
Required	51	Create a graphical user interface that adheres to industry standards.		
Required	52	Code a program that will produce formatted output.		
Required	53	Code an application that uses mathematical operations and built-in functions.		
Required	54	Write a program that uses variables and constants.		
Required	55	Write a program that accepts user input.		
Required	56	Write a program that uses arrays.		
Required	57	Write a modular program that uses functions or methods.		
Required	58	Write a program that uses conditional structures.		
Required	59	Write a program that uses looping structures.		
Required	60	Write a program that uses counters and accumulators.		
Required	61	Identify the purpose of an executable file.		

6640 36 weeks	Programming TASKS/COMPETENCIES		Date	Rating
	Developing Interactive Multimedia Applications			
Required	62	Create an object-oriented program.		
Required	63	Code a program to display graphics.		
Required	64	Code a program to incorporate multimedia.		
Required	65	Code a program to animate objects.		
Required	66	Examine the history of game design and development.		
Required	67	Analyze the impact of intellectual property law on game design.		
Required	68	Identify the target markets for game applications.		
Required	69	Identify game genres.		
Required	70	Examine a variety of game programming platforms.		
Required	71	Create a storyboard.		
Required	72	Code a program from the storyboard.		
Required	73	Create an object within the context of a game.		
Required	74	Specify behaviors of an object within the context of a game.		
Required	75	Develop a game program that uses a scoring method.		
Required	76	Create a game program with multiple levels.		
	Using Web Technology			
Required	77	Explain how to locate resources and references to aid program development.		
Required	78	Evaluate the validity of sample code obtained from the Internet and/or other sources.		
Optional	79	Develop a Web page, using HTML and/or JavaScript.		
Optional	80	Publish a program link on a Web page.		
	Preparing for Industry Certification			
Optional	81	Describe the process and requirements for obtaining industry certifications related to the Programming course.		
Optional	82	Identify testing skills/strategies for a certification examination.		
Optional	83	Demonstrate ability to successfully complete selected practice examinations (e.g., practice questions similar to those on certification exams).		
Optional	84	Successfully complete an industry certification examination representative of skills learned in this course (e.g., MCP, IC3, NOCTI).		
	Developing Employability Skills			
Required	85	Identify careers in the information technology industry.		
Required	86	Describe ways that computer programs can be used in business and industry.		
Required	87	Create or update a résumé.		

6640 36 weeks	Programming TASKS/COMPETENCIES		Date	Rating
Required	88	Investigate information technology educational and job opportunities.		
Required	89	Assemble a professional portfolio.		
Required	90	Describe basic employment activities.		
Optional	91	Deliver an oral presentation of the professional portfolio.		
Required	92	Identify potential education and employment barriers for nontraditional groups and ways to overcome those barriers.		

	Locally Developed Tasks/Competencies			

SOL Correlation by Task

Task/Competency Number	Task/Competency Statement	Standards
6640.001	Demonstrate positive work ethic.	**History and Social Science** CE.4, GOVT.17
6640.002	Demonstrate integrity.	**History and Social Science** CE.3, CE.4, GOVT.17
6640.003	Demonstrate teamwork skills.	**History and Social Science** CE.4, GOVT.17
6640.004	Demonstrate self-representation skills.	**History and Social Science** CE.4, GOVT.17
6640.005	Demonstrate diversity awareness.	**History and Social Science** CE.3, CE.4, GOVT.3, GOVT.11, GOVT.17, VUS.14
6640.006	Demonstrate conflict-resolution skills.	**History and Social Science** CE.4, GOVT.17
6640.007	Demonstrate creativity and resourcefulness.	**History and Social Science** CE.4, GOVT.17
6640.008	Demonstrate effective speaking and listening skills.	**English** 6.2, 7.1, 7.2, 8.2, 9.1, 10.1, 11.1, 12.1
6640.009	Demonstrate effective reading and writing skills.	**English** 6.6, 6.7, 6.8, 7.6, 7.7, 7.8, 8.6, 8.7, 8.8, 9.5, 9.6, 9.7, 10.5, 10.6, 10.7, 11.5, 11.6, 11.7, 12.5, 12.6, 12.7
6640.010	Demonstrate critical-thinking and problem-solving skills.	**History and Social Science** CE.1, CE.4
6640.011	Demonstrate healthy behaviors and safety skills.	**History and Social Science** GOVT.16, GOVT.17, VUS.15
6640.012	Demonstrate an understanding of workplace organizations, systems, and climates.	**History and Social Science** CE.12, GOVT.15, VUS.15
6640.013	Demonstrate lifelong-learning skills.	**History and Social Science** CE.3, CE.14
6640.014	Demonstrate job-acquisition and advancement skills.	**History and Social Science** CE.12, CE.14
6640.015	Demonstrate time-, task-, and resource-management skills.	**History and Social Science** CE.4, CE.11, GOVT.17
6640.018	Demonstrate proficiency with technologies common to a specific occupation.	**History and Social Science** CE.14, VUS.15
6640.019	Demonstrate information technology skills.	**History and Social Science** CE.14, VUS.15
6640.020	Demonstrate an understanding of Internet use and security issues.	**History and Social Science** CE.14, VUS.15
6640.021	Demonstrate telecommunications skills.	**History and Social Science** CE.14, VUS.15

Task/Competency Number	Task/Competency Statement	Standards
6640.022	Examine aspects of planning within an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5 **Mathematics** PS.8
6640.023	Examine aspects of management within an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5 **History and Social Science** GOVT.18
6640.024	Examine aspects of financial responsibility within an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5 **Mathematics** MA.14
6640.025	Examine technical and production skills required of workers within an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5
6640.026	Examine principles of technology that underlie an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5
6640.027	Examine labor issues related to an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5 **History and Social Science** GOVT.16, GOVT.17
6640.028	Examine community issues related to an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5 **History and Social Science** GOVT.16, GOVT.18
6640.029	Examine health, safety, and environmental issues related to an industry/organization.	**English** 9.5, 10.5, 11.5, 12.5 **History and Social Science** GOVT.9, GOVT.16
6640.034	Describe the development of computers and current industry trends in the programming field.	**English** 10.5, 11.5, 12.5 **History and Social Science** VUS.15, WHII.16
6640.035	Describe the development of programming languages and applications.	**English** 10.5, 11.5, 12.5 **History and Social Science** VUS.15
6640.036	Describe the functions of computer hardware, computer software, and computer system components.	**English** 10.5, 11.5, 12.5
6640.038	Identify the software development life cycle (SDLC).	**History and Social Science** GOVT.1
6640.040	Analyze the problem statement.	**English** 10.5, 11.5, 12.5 **Mathematics** COM.1, COM.4
6640.041	Create possible solutions to the problem.	**History and Social Science** GOVT.1 **Mathematics** COM.1, COM.3, COM.4
6640.042	Determine the best solution to the problem.	**History and Social Science** GOVT.1 **Mathematics** COM.1, COM.4
6640.043	Design a program, using an algorithm, pseudocode, a flowchart, and/or a decision table.	**Mathematics** COM.1, COM.2, COM.4

Task/Competency Number	Task/Competency Statement	Standards
6640.044	Code the program, using a programming language.	**Mathematics** COM.1, COM.2, COM.4
6640.045	Test the program with sample data.	**Mathematics** COM.2, COM.18
6640.046	Debug the program.	**Mathematics** COM.2, COM.18, COM.19, COM.20
6640.047	Document the program.	**Mathematics** COM.2, COM.20
6640.048	Describe maintenance procedures.	**English** 10.5, 11.5, 12.5
6640.049	Identify syntax errors of a given programming language.	**Mathematics** COM.2, COM.18, COM.19, COM.20
6640.050	Identify industry standards for a graphical user interface (GUI).	**English** 10.5, 11.5, 12.5
6640.051	Create a graphical user interface that adheres to industry standards.	**Mathematics** COM.6
6640.052	Code a program that will produce formatted output.	**Mathematics** COM.7
6640.053	Code an application that uses mathematical operations and built-in functions.	**Mathematics** COM.1, COM.11, COM.12, COM.13
6640.054	Write a program that uses variables and constants.	**Mathematics** COM.9, COM.10, COM.11
6640.055	Write a program that accepts user input.	**Mathematics** COM.6
6640.056	Write a program that uses arrays.	**Mathematics** COM.15, COM.16
6640.057	Write a modular program that uses functions or methods.	**Mathematics** COM.5
6640.058	Write a program that uses conditional structures.	**Mathematics** COM.3, COM.13, COM.14, COM.15, COM.16
6640.059	Write a program that uses looping structures.	**Mathematics** COM.3, COM.14, COM.15
6640.060	Write a program that uses counters and accumulators.	**Mathematics** COM.3, COM.15
6640.062	Create an object-oriented program.	**Mathematics** COM.3, COM.5, COM.13, COM.16, COM.20
6640.063	Code a program to display graphics.	**Mathematics** COM.7, COM.8
6640.064	Code a program to incorporate multimedia.	**Mathematics** COM.6, COM.7, COM.8
6640.065	Code a program to animate objects.	**Mathematics** COM.6, COM.8, COM.15, COM.16
6640.066	Examine the history of game design and development.	**English** 10.5, 11.5, 12.5
6640.067	Analyze the impact of intellectual property law on game design.	**English** 10.5, 11.5, 12.5
6640.071	Create a storyboard.	**English** 10.5, 11.5, 12.5

Task/Competency Number	Task/Competency Statement	Standards
6640.072	Code a program from the storyboard.	**Mathematics** COM.3, COM.4
6640.073	Create an object within the context of a game.	**Mathematics** COM.8
6640.074	Specify behaviors of an object within the context of a game.	**Mathematics** COM.5, COM.16
6640.075	Develop a game program that uses a scoring method.	**Mathematics** COM.3, COM.15
6640.076	Create a game program with multiple levels.	**Mathematics** COM.13, COM.20
6640.078	Evaluate the validity of sample code obtained from the Internet and/or other sources.	**Mathematics** COM.2, COM.18
6640.079	Develop a Web page, using HTML and/or JavaScript.	**Mathematics** COM.19
6640.085	Identify careers in the information technology industry.	**English** 10.5, 11.5, 12.5
6640.086	Describe ways that computer programs can be used in business and industry.	**English** 10.5, 11.5, 12.5 **History and Social Science** VUS.15, WHII.16
6640.087	Create or update a résumé.	**English** 10.6, 10.7, 11.6, 11.7, 12.6, 12.7
6640.088	Investigate information technology educational and job opportunities.	**English** 10.5, 10.8, 11.5, 11.8, 12.5, 12.8
6640.091	Deliver an oral presentation of the professional portfolio.	**English** 10.1, 11.1, 12.1