AP® Computer Science A Sample Syllabus 4

Syllabus 1172780v1

Curricular Requirements		Page(s)
CR1	The course teaches students to design and implement computer-based solutions to problems.	3, 4, 5, 6, 7, 8, 10
CR2a	The course teaches students to use and implement commonly used algorithms.	4, 9
CR2b	The course teaches students to use commonly used data structures.	4, 5, 7
CR3	The course teaches students to select appropriate algorithms and data structures to solve problems.	6, 8, 9
CR4	The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.	3, 7, 10
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.	3, 6
CR6	The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.	1
CR7	The course teaches students to recognize the ethical and social implications of computer use.	6, 7

Course Philosophy:

Students enter AP Computer Science A with varying backgrounds. In order to meet the diverse needs and appeal to the interests of my students, I try whenever possible to teach concepts inside of contexts. This keeps students motivated and allows me to spiral my instruction. Working with contexts allows me the chance to arrange topics to maximize student excitement at important times, such as the beginning and end of first semester, and just before students choose their courses for the following year.

Texts:
Kölling, Michael. *Introduction to Programming with Greenfoot*. Upper Saddle River, NJ: Pearson Higher Education, 2010. www.greenfoot.org/book

Litvin, Gary and Litvin, Maria. *Java Methods, 2nd AP Edition*. Andover, MA: Skylight Publishing, 2011. www.skylit.com/javamethods

Programming Environments: BlueJ (bluej.org) and Greenfoot (greenfoot.org)

Online Resources:
Kölling, Michael . *Joy Of Code*. http://blogs.kent.ac.uk/mik/category/joy-of-code/.

Parlante, Nick . *CodingBat*. http://codingbat.com/java.

Syllabus at a Glance:

Semester 1:	Semester 2:
• Introduction to AP Computer Science A with Greenfoot • Java Basics, with Magpie Lab and Text-Based Game • 1D Arrays with Bioinformatics or Finch Robot • Writing Classes using Encapsulation • Lists and Interacting Objects with Greenfoot • Semester Final Project with Greenfoot Competition	• 2D Arrays with Picture Lab • Interfaces, Abstract Classes, and Polymorphism • Recursion with Flood Fill • Searching and Sorting Algorithms • Review for AP Exam with Elevens Lab • Projects

Learning Environment:
In a typical class period (52 minutes) students work at the computer in 3 different kinds of tasks:
Notes, in which students take notes "in code" and play around with that code.
Labs, in which students spend at least 20 hours of hands-on time, and are given small methods to complete inside a larger programming project. **[CR6]**
Major programs, in which pairs of students work together to make creative programs over 4-7 class periods. Major programs always have extensions for students who finish early.

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

I try to schedule plenty of work days so that almost all students finish their programs in class. If students miss class, need extra time, or would like extra help, they may come into the classroom before school or during lunch. Most students need to go somewhere after school, so the classroom is usually not open.

Approximately once every 10 days we leave the classroom (to avoid the distraction of the computers) and have an "unplugged" lesson. I need to find an available room in the school to do this.

The majority of a student's grade comes from completion of the Labs and Major programs, which I check off during class time. For me, Labs serve as formative assessments and Major programs serve as summative assessments. Students demonstrate to me that their programs work to specifications, and I look at their code to make sure the programs demonstrate good style.

Quizzes: At the end of each unit, students take a quiz which measures their understanding of not only Java, but also the context of that unit. Also, if tasks from CodingBat.com were given during a certain unit, similar questions will appear on the quiz. During second semester, these quizzes have at least one question taken from previous AP Computer Science A Free-Response Questions.

Know, Understand, Do: Our school expects each teacher to write KUDOs (Know, Understand, and DO) statements for student outcomes of each unit. At the beginning of each unit, students are asked to complete a form with all of that unit's objectives in the form "I know the meaning of . . ." , " I understand that . . ." and "I can do the following: " At the end of the unit, and before any summative test, students complete the form again to note how much they have learned and what they still need to learn.

Unit 1: Introduction to AP Computer Science A with Greenfoot (4 weeks)

Know, Understand, Do	Students know the meaning of algorithm, programming language, compiler, subclass, superclass, constructor, API, compiler, Javadoc, type, return type, method call, parameter, comment, assignment statement, `class`, `new`, and `extends`. Students understand that a class is a template for making objects, and a subclass inherits the methods of its superclass. Students can create a subclass, create and call a method, use an API, debug and test programs, and use the `new` command to instantiate an object.
Resources, Labs, Tasks	*Java Methods*: Parts of Chapters 1-5. Part of each lesson in the first week of class is devoted to topics such as algorithms, levels of programming languages, and converting from decimal to binary and hex.

	Chapters 1-4 in the Greenfoot book describe the Little Crab scenario. In this scenario students create a game where a Crab moves around the screen via keyboard commands. The Crab tries to eat Worms while avoiding Lobsters. Students will read the chapter, perform all exercises in the book, and then fill out a worksheet made by the instructor which reviews the concepts taught in the book. When they are done, they have a working game that they can customize and extend. **[CR1]**
Teaching Points	Some students in class may have trouble reading the text, so I informally assess their ability to read the text and learn from it. Using Greenfoot is a really fun way to start the year because every student, at any level, can find something interesting to do. I create a forum on our school's online classroom management system entitled "Post your Greenfoot questions here." Students ask a lot of "how do I . . ." questions. Most of the questions require essential knowledge from weeks or months ahead of where we are right now, but I still try to give the students snippets of code that plant a seed when we discuss those topics later in the course.

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

Unit 2: Java Basics with Magpie Lab (5 weeks) [CR4]

Know, Understand, Do	Students know the meaning of instantiate, parameter, `String`, and `else`. Students understand that every method has a return type, Strings variables are objects, in computer science we often start counting at 0, and object variables store a reference to the object. Students can use the mod operator (%) to solve problems, use methods in the `Scanner`, `Math`, `Random`, `String` classes, create loops using `for` and `while` statements, create nested logic using `if/else if/ else` statements, create and use a constant, create a text-based menu-driven program, and compare different coding solutions to determine which are more efficient. **[CR5]**
Resources, Labs, Tasks	*Java Methods*: Parts of Chapters 6-8. CodingBat.com: "Warmup-1", "String-1", "Logic-1" Week 1: Students receive a set of notes regarding the `Scanner`, `Random`, `Math`, and `String` classes. Week 2: Students do a Conditional Statements Lab and a Looping Lab.

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.

	Week 3: Students do Activities 1-4 in the College Board Magpie Lab. **[CR1]** Weeks 4-5: Students do their first **Major program**, a text-based game with a menu. **[CR1]**
Teaching Points	It is important to balance the fun of Unit 1 with some core knowledge and consistent practice in Unit 2.

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

Unit 3: 1D Arrays with Bioinformatics or Finch Robot (3 weeks) [CR2a] [CR2b]

Know, Understand, Do	Students know the meaning of array, element of an array, index of an array, and ArrayIndexOutOfBoundsException. Students understand that an array variable holds a reference to the array's contents, arrays are always passed to methods as parameters, and the first index of an array is 0 and the last index of an array is "length - 1." Students can instantiate an array in two different ways; traverse an array; implement methods to find the maximum, minimum, and average of an array; implement a linear search of an array; and write methods to perform other array computations.
Resources, Labs, Tasks	*Java Methods:* Section 12-1 and 12-2: Exercises 1-4. CodingBat.com: "Array-1" FinchRobot.com: Array Assignments Labs with Finch Robot: Students work in pairs using the inexpensive and easy-to-use Finch Robot. The labs for this unit involve learning about the Finch's capabilities, followed by labs where the students design an experiment to collect Finch sensor data into arrays and process that data using introductory array algorithms. The finchrobot.com web site has some premade lesson plans for different topics. **[CR1]** Lab Alternative to Finch Robot - Bioinformatics: Students simulate working with DNA by writing programs that fill an array with characters 'a', 'c', 't', 'g.' Students can perform some simple array manipulations such as finding the ratio of similarity between two arrays, copying one array into another, creating the "reverse complement" of an array, and looking for a 3-character sequence. **[CR1]**
Teaching Points	We save 2D arrays, sorting, and binary search for later. The Finch's sensors allow students to create a series of `while` loops, giving them solid understanding of event-driven programming.

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

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

	While groups of 3 or 4 are possible, groups of 2 are ideal for robotics programming. Both the Finch Lab and the Bioinformatics Lab offer a chance for students to see the connections between Computer Science and Science. Even if a class uses Finch Robots, the Bioinformatics Lab might be interesting as a series of homework assignments.

Unit 4: Writing Classes using Encapsulation (2 weeks)

Know, Understand, Do	Students know the meaning of encapsulation, instance variable, getter and setter methods, default constructor, `NullPointerException`, `this`, static methods, static fields. Students understand that a class can have more than one constructor or no constructor, every non-void method requires a reachable return statement; and private methods and instance variables can be accessed only within the code of that class. Students can design and create a class and store objects of that class in an array.
Resources, Labs, Tasks	*Java Methods*: Chapter 9. Exercises. Notes are taken in BlueJ by creating one class per day. Rather than focus on complex algorithms, students write basic classes that represent data objects. Labs include writing classes such as `Student`, `Fraction`, `ComplexNumber`, `Quadratic Equation` and `Movie`. **[CR1]** Major program: Working with a partner, students complete the SnackBar assignment from *Java Methods* Chapter 9. **[CR1]**
Teaching Points	A series of homework worksheets give students some short, light practice creating classes that have multiple constructors, accessor methods, and mutator methods.

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

Unit 5: Lists, `ArrayLists`, and Interacting Objects with Greenfoot (4 weeks) [CR2b]

Know, Understand, Do	Students know the meaning of `List`, `ArrayList`, casting, abstract class, overloading, and `null`.

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

	Students understand that: an abstract class cannot be instantiated, and should contain code that is common to all subclasses; an `ArrayList` is a `List`, and can only hold objects; an `ArrayList` is more flexible than an array; sometimes it is better to store data in an array and sometimes in an `ArrayList` **[CR3]**; and one has to be careful when adding to or removing from an `ArrayList` inside a loop. **[CR5]** Students can instantiate an `ArrayList` and populate it, create a complicated program with multiple classes that interact with each other, check for collision of objects, write a `for-each` loop to traverse and process a `List`, and add to and delete from a `List`.
Resources, Labs, Tasks	*Java Methods:* Chapter 13. Exercises 1-4. Greenfoot Chapters 6 & 7. Most exercises are step-by-step, but as Chapter 7 progresses; there is less and less direction and students need to figure out how to implement the advanced features of an Asteroids game on their own. The final product requires a large amount of work that is not specified in the book, and counts as a **Major program**. **[CR1]**
Teaching Points	The Greenfoot web site allows students to upload their projects to greenfoot.org for others to play. After students upload their versions of the Asteroids game, they take time to play each others' games and give feedback.

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

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.

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

Semester 1 Final Project with Greenfoot - Greeps Competition

The Greenfoot book, as well as the web site(http://www.greenfoot.org/competition/greeps/), explain directions for a Greeps competition, which we use as our semester **final project**. In lieu of an exam, students work in pairs to program an alien Actor that will collect tomatoes during a timed run. Grading is generous, with an "A minus" awarded if the team has well-written code that follows the rules of competition, meets a reasonable score threshold, and is completed on time. We hold a competition during our semester exam period to determine a ranking of scores. Higher grades are awarded depending on a team's score.

Because the Greeps game is public on the web, I slightly modify the rules and the code so that students do not gain an advantage by using code written by others. We discuss the ethical concerns of using another person's code without their permission. **[CR7]**

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

Unit 6: 2D Arrays with Picture Lab (3 weeks) [CR2b] [CR4]

Know, Understand, Do	Students know the meaning of two-dimensional array, and pixel. Students understand that a 2D array is actually an array of 1D arrays; a picture is made of pixels, which store Red, Green, and Blue components; and using pictures from public web sites comes with responsibility for both legal and ethical use. **[CR7]** Students can manipulate pictures by writing methods in Java, process an individual row or column of a two-dimensional array, and process all elements of a two-dimensional array.
Resources, Labs, Tasks	*Java Methods:* Chapter 12. Students complete the Chomp Lab. (see http://en.wikipedia.org/wiki/Chomp) and do Exercises 12-13. **[CR1]** Students do all activities (A1 through A9) of the AP Computer Science A Picture Lab, as well as the Extensions (E1 and E2), and complete teacher-created worksheets that summarize each activity. **[CR1]**
Teaching Points	I encourage students to choose their own pictures for Picture Lab. We discuss the social and ethical implications of using copyrighted photos, as well as the implications of using a photo of someone in ways that they do not approve of. **[CR7]** In class discussion, the students make comparisons to Photoshop, and I challenge the students to create an effect that Photoshop can't do. I'll try to place the units that are the most fun at the end of first semester and the start of second semester, as this is the time when students are making the class choices for the following year. Students often bring their friends into the lab to show off what they have created. We come back to Picture Lab at the end of the school year; see Unit 8 Recursion and Unit 11 Projects.

Unit 7: Interfaces, Abstract Classes, and Polymorphism (3 weeks)

Know, Understand, Do	Students know the meaning of abstract method, abstract class, polymorphism, interface, and `super`. Students understand that an interface can be thought of as a set of specifications that a class must fulfill, an interface is different from an abstract class, an interface acts as a superclass and as a type, and all classes in Java are a subclass of `Object`.

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.

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

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

	Students can design and implement an interface, design and implement an abstract class with subclasses, create an `ArrayList` of an Interface type and use a for-each loop to iterate through it, and use the keyword `super` correctly.
Resources, Labs, Tasks	*Java Methods:* Chapter 11. Exercises 1-5. LibraryItems Lab. `LibraryItem` is an abstract class, each pair of students in the class creates a subclass and shares their code with their class. Students create an `ArrayList<LibraryItem>` and perform functions on that `ArrayList`. **[CR1]** Interface Lab (design an interface that can be implemented by several classes that were previously written in the course). **[CR1]** Polymorphism Lab. This lab is different each year I teach the course, depending on the class. The appropriate lab is photocopied, dynamically, at the time the class executes the lab. **[CR1]**
Teaching Points	At this point in the school year we, start to get ready for the AP Exam. For this subject in particular, it's helpful for students to work out practice multiple choice questions.

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

Unit 8: Recursion with Flood Fill (2 weeks)

Know, Understand, Do	Students know the meaning of recursion, flood fill algorithm, and `StackOverflowException`. Students understand that every recursive method must have a base case (or terminating case). Students can give an example of real-world recursion, act out a recursive method, write a recursive method, and give an example of when not to use recursion. **[CR3]**
Resources, Labs, Tasks	*Java Methods:* Chapter 4. Case Study "File Manager." **[CR1]** CodingBat.com Section "Recursion-1." Worksheet with problems tracing recursive algorithms. There are several web sites that explain "Recursion with Flood Fill" or "paint can flood fill."
Teaching Points	Students learn recursion better if they can first act it out. One way to do this is to photocopy dozens of copies of a short recursive method. As the class is tracing a method call, give each student one copy of

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

	itself have the students mark where they are in the current method and pass out another version of the method, which students "stack" on top of the first one. As one method ends, have students remove that paper from the stack and return to the place they left off in the previous method.

Unit 9: Searching and Sorting Algorithms, Comparable Interface (2 weeks)

Know, Understand, Do	Students know the meaning of swap, sorting, binary search. Students understand that a binary search can be performed in log2(n) time, an interface can make your code much more useful, and different sorting algorithms have different efficiencies. **[CR3]** Students can perform a trace of a binary search algorithm, explain in words the selection, insertion, and merge sort algorithms; and write a sorting or searching method that takes an array of `Comparable` as a parameter. **[CR2a]**
Resources, Labs, Tasks	*Java Methods:* Chapter 14. Benchmarks Lab helps students compare different sorting algorithm efficiencies. Exercise 4, and 11-14. **[CR3]** CodingBat.com Section "Arrays-2" (this section is not about sorting, but gives good practice with array manipulation via loops). We write the sorting and binary search algorithms as a class, together. We sort arrays of `ints`, `doubles`, and `Strings`. After having a discussion about why we needed to write different methods depending on the type of data in the array, we discuss how writing a method to sort arrays of type `Comparable` makes our work much easier.
Teaching Points	Before students study any particular sorting algorithms, we do a "You Sort" class activity away from the computers. Pairs of students are given small cards with numbers on them, face down. One person can look at the numbers and tell the other which of two is larger. The other cannot look at the numbers, but can point to two numbers, ask which is larger, and move numbers around (face down). I ask the students to write their algorithms down, and sometimes we make videos of the algorithms and post them for the class to see. When its time to name standard algorithms, the students enjoy saying "that was my algorithm!" If their algorithm is not selection, insertion, or merge, the students enjoy researching what named algorithm most closely matches theirs.

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

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

Unit 10: Review for AP Exam with Elevens Game (4 weeks) [CR4]

Know, Understand, Do	Students know the meaning of Model-View-Controller programming pattern. Students understand that separating the Graphical User Interfaces (GUI) from the logic for a game makes it more modifiable and more portable. Students can implement a complicated program with many interacting classes.
Resources, Labs, Tasks	Students will complete all the activities (#1-11) in the Elevens Game Lab, and complete teacher-created worksheets that summarize each activity. **[CR1]**
Teaching Points	During this time, students spend about one day per week answering previous free-response questions. Students who want to learn about Graphical User Interfaces can look at the GUI code of this project and apply what they learn to their final project.

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

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

Unit 11: Projects: (4 weeks)

Know, Understand, Do	Students know the meaning of software engineering. Students understand that creating software with others takes planning and coordination. Students can design, implement, and test a complicated piece of software.
Resources, Labs, Tasks	Students do three projects after the AP Exam. 1. **Responsible Use of Computers**. Students research some current topic that interests them in this area. They may write a paper of 3-5 pages, create a Power Point or Prezi, or make a video to share with the class. 2. **Fine Arts Week Pictures**. Each May our entire school has a week devoted to fine arts. All students can attend concerts, performances, and shows. My students get into the spirit by creating an artistic work using PictureLab. The students write up a paragraph explaining why they chose this picture and how they created the picture using their methods of PictureLab. I develop 5"x7" photos and put the pictures as well as the paragraphs in a place that the entire student body can see them during the week. 3. **Final Programming Project**. Students create a final project, in lieu of a final exam, with a partner.

Teaching Tips	Students often have other AP exams around this time, so it helps to have Tasks that are self-directed. There is no final exam for the course, so during our final we have a "show and tell" of everyone's work.