Sample Syllabus 1 (AP Computer Science A)

Bekki George
James E. Taylor High School
Katy, Texas
rebeccageorge@katyisd.org
http://schools.katyisd.org/campus/ths/index.htm

School Profile

Location and Environment: James E. Taylor High School is located 25 miles west of downtown Houston in a suburban area near Katy, Texas, a town with a population of 12,000. The Katy Independent School District itself encompasses a large area consisting of approximately 65,000 households and 41,500 students. The socioeconomic level is predominantly middle to upper-middle class.

Grades:	9–12	
Type:	Public school	
Total Enrollment:	2,770	
Ethnic Diversity:	Asian American	10.0 percent
	Hispanic/Latino	7.2 percent
	African American	3.4 percent
	Native American	0.1 percent
College Record:	71 percent attend four-year colleges	
	17 percent attend community colleges	

Personal Philosophy

I want my students to enjoy computer science as much as I do. I get very excited when they understand what I'm teaching. Every day, I hear "All right!" or a squeal of delight when one of my kids gets something difficult to work. Their enthusiasm is contagious. My reward comes when students can apply the programming tools they have learned to real-life examples on their own. Computer science is more than just programming. Students should leave my class with a clear understanding of Java and the ability to adapt to any new programming language that they are taught in college. I want them to have the confidence to tackle any problem-solving obstacles they encounter.

Class Profile

Five sections (27 students per section)
Classes meet for 50 minutes each school day.

Course Overview

AP Computer Science 1 (the official name for my course) is both a college-prep course for potential computer science majors and a foundation course for students planning to study in other technical fields such as engineering, physics, chemistry, and geology. The course emphasizes programming methodology, procedural abstraction, and in-depth study of algorithms, data structures, and data abstractions, as well as a detailed examination of a large case study program. Instruction includes preparation for the AP Computer Science A Exam.

Texts

Bergin, Joseph et al. *Karel J Robot: A Gentle Introduction to the Art of Object-Oriented Programming in Java*. Redwood City, Calif.: Dreamsongs Press, 2005. http://csis.pace.edu/~bergin/KarelJava2ed/Karel%2B%2BJavaEdition.html. Introduces objects and inheritance.

College Board. *AP Marine Biology Simulation Case Study.* New York: College Entrance Examination Board, 2002. Download from the Course Home Pages: apcentral.collegeboard.com/compscia or apcentral.collegeboard.com/compsciab.

Horstmann, Cay. *Big Java*. Hoboken, N.J.: Wiley, 2002.

Lambert, Ken, and Martin Osborne. *Fundamentals of Java, Comprehensive Course.* 2nd ed. Boston: Course Technology, 2002.

Course Planner

The Resources listings include the following text references: *Karel J Robot* (KJR), *Marine Biology Simulation Case Study* (MBS), *Big Java* (BJ), and *Fundamentals of Java* (FJ).

Unit (Weeks)	Title, Topics, and Student Objectives	Resources, Assessments, and Strategies
1 (0–3)	**Karel J Robot** **Topics:** • Objects • Classes • Looping • Conditionals **Objectives:** • Write and use simple classes with Karel J Robot • Learn the basics of conditionals and looping	**Resource:** • KJR **Assessments:** • Program-specific tasks for Karel • Create a SmartRobot Class to teach Karel more commands: turnRight(), turnAround(), climbStair(). • Clear a field of beepers (using loops). • Redistribute a field of beepers (using loops and conditionals). • Run a hurdle race: o same height and equally spaced; o same height and unequally spaced; o different heights and unequally spaced.
2 (4)	**Java Basics** **Topics:** • Java basics • Using the compiler • Input and output **Objectives:** • Understand terminology: compiler, IDE, JVM • Edit, compile, and run a simple program in Java • Understand the different compile time errors, runtime errors, and logic errors • Use BufferedReader for input • Use output with System.out and format output to look nice	**Resource:** • FJ: lesson 3, Critical thinking **Assessments:** • Labs: Triangle, Rectangle, Square: Area, and perimeter program • Get input for the registrar's office program. **Strategies:** Assign a lot of small programs that illustrate different types of input and output—make sure students have used every type of input and displayed it in different ways.

Unit (Weeks)	Title, Topics, and Student Objectives	Resources, Assessments, and Strategies
3 (5)	**Defining Variables, Arithmetic Expressions** **Topics:** • Using and understanding variables • Comments • Arithmetic expressions in Java programs **Objectives:** • Understand terminology: comments, variables, constants, reserved words, literals • Declare and initialize variables and constants in Java • Understand mathematical expressions in Java and their precedence • Use casting to make their data more accurate • Use the assignment operator correctly	**Resource:** • FJ: lesson 3, Projects **Assessments:** Labs: • Paycheck program; have employee information entered and calculate pay. • Modify the paycheck program to also include any overtime hours in the calculations. **Strategies:** • Students need practice with how the different types, double and int, relate when they are used in mathematical operations. • Worksheets are helpful here. • Present a lot of small program examples in which they have to find the errors.
4 (6–7)	**Introduction to Classes and OOP** **Topics:** Creating and using classes **Objectives:** • Understand terminology: constructor, accessor, mutator, instance variable, and more • Understand the difference between public and private access in a class • Use and comprehend the DecimalFormat class and the Random class • Write classes from scratch	**Resource:** • BJ: chapter 3 **Assessments:** Labs: Purse class and StampMachine class **Strategies:** • Go slowly and show as many examples as possible. • Start with very simple examples. • Give students classes to complete.
5 (8–12)	**Conditionals and Looping** **Topics:** if, if-else, while, for **Objectives:** • Understand terminology: control statements, counter, infinite loop, iteration, nested loops, logical operators, truth tables • Construct syntactically correct loops and conditional statements • Understand the different errors that may occur with loops • Use logical operators to make programs more robust • Construct truth tables	**Resources:** • FJ: lessons 4 and 6, Projects **Assessments:** Labs: • Approximate PI using Leibniz's method • Base Conversion; convert from base 10 to base 2 • Guess My Number game • Euclidean algorithm program • Perimeter and area of rectangles using all combinations of certain range **Strategies:** • This unit needs a lot of programs. • Students need practice writing different types of loops and conditionals. • Worksheets that focus on the output of certain statements are very helpful here.

Unit (Weeks)	Title, Topics, and Student Objectives	Resources, Assessments, and Strategies
6 (13–14)	**The String Class** **Topic:** • String class **Objectives:** • Instantiate String objects • Understand that Strings are immutable • Use appropriate String methods to solve problems	**Resource:** • FJ: lesson 10.1 **Assessments:** • FJ: exercise 10.1 • Lab: LineEditor Class (*AP CS Course Description*) **Strategies**: Work several examples using the substring method.
7 (15–17)	**ArrayList** **Topic:** • Using ArrayList class **Objective:** • Use the ArrayList methods	**Resources:** • FJ: lesson 10.7 • BJ: 13.1 and 13.2 **Assessments:** • BJ: exercise p.13.1 • WordList (2004 AP CS A Exam, Free-Response Question 1, AP Central) **Strategies:** • Stress the difference between add and set. • Draw pictures of the ArrayList after add, set, and remove have been performed.
8 (18)	**Arrays** **Topics:** • Declaring and initializing arrays • Manipulating arrays with loops • Creating parallel arrays **Objectives:** • Understand terminology: array, element, index, logical size, physical size, parallel arrays • Declare one-dimensional arrays in Java • Use initializer lists when declaring arrays • Manipulate arrays using loops and array indices • Use the physical and logical size of an array together to guarantee they do not go beyond the bounds of their array • Understand how parallel arrays can be useful when processing certain types of data • Work with arrays of primitive data types as well as arrays of objects	**Resource:** • FJ: lesson 8, Projects **Assessments:** Lab: For one-dimensional arrays, read in numbers and place each one in an even, odd, and/or negative list. **Strategies:** • Students need practice manipulating loops that work with arrays. • Worksheets where they show the output from sample programs are always helpful here. • Students also need to be reminded about the indexing of arrays beginning at zero.

Unit (Weeks)	Title, Topics, and Student Objectives	Resources, Assessments, and Strategies
9 (19–21)	**Searching and Sorting Arrays** **Topics:** • Bubble, Selection, Insertion sorts • Sequential and Binary searches **Objectives:** • Write a method for searching an array • Perform insertions and deletions at given positions in arrays • Trace through sorting and searching algorithms • Understand the algorithms behind each of the following searching and sorting techniques: bubble, selection, and insertion sorts; sequential search and binary search • Understand the efficiency of each sort and search and when it is desirable to use each one	**Resource:** • FJ: lesson 10 **Assessments:** Lab: Students make their own "utility" class that includes all of these sorts and searches. **Strategies:** • Students need practice tracing through sorts and searches. • Worksheets: show the sort or search at a certain "pass." • Students also do well with a worksheet that talks about the efficiency of each of the strategies they have learned, efficiency for a sorted versus unsorted list, and "best," "worst," and "average" efficiency. (See appendix B for lab and worksheets.)
10 (22–24)	**MBS (chapters 1–3)** **Topics:** • Experimenting with a large program • Using classes • Modifying classes **Objectives:** • Run the case study and analyze output • Experiment with the Simulation • Understand the Fish Class, Simulation Class, and the Environment Interface • Modify the Fish Class	**Resource:** • MBS: chapters 1–3 **Assessments:** Exercises and analysis from the text **Strategies:** • Read the manual thoroughly. • Be familiar with all the classes and interfaces discussed.
11 (25–27)	**More on Classes, Inheritance, Interfaces** **Topics:** • Classes • Inheritance • Abstract classes • Interfaces **Objectives:** • Demonstrate inheritance by extending a class • Understand polymorphism and know when it is appropriate to override methods in a super class • Create and extend an abstract class • Implement an interface	**Resources:** • BJ: chapter 11 • FJ: lessons 9.5 and 9.6 **Assessments:** • Create an abstract Shape class. • Pet Parade (2004 AP CS A Exam: Free-Response Question 2, on AP Central) **Strategies**: Draw pictures of the inheritance hierarchy. Note: This unit could be moved to after unit 12 if you wish to use the MBS to introduce inheritance.

Unit (Weeks)	Title, Topics, and Student Objectives	Resources, Assessments, and Strategies
12 (28–29)	**MBS (chapter 4)** **Topic:** • Inheritance **Objective:** • Use inheritance to extend the Fish Class	**Resource:** • MBS: chapter 4 **Assessments:** Exercises and analysis from the text **Strategies:** • Have fun with this chapter. • Allow the students to be creative after working through the exercises and analysis. • Create different kinds of fish or other objects.
13 (30–31)	**Recursion (and Merge Sort)** **Topics:** • Recursion • Merge Sort **Objectives:** • Create a recursive method to solve a problem • Understand the difference between recursive and iterative solutions to a problem • Understand and use the Merge Sort.	**Resources:** • FJ: lesson 11.1 • BJ: section 18.4 **Assessments:** • Factorial program • Rewrite loop programs with recursion. **Strategies:** • Students need lots of practice on recursion. • Mathematical worksheets on recursion help introduce this topic. • Do a lot of examples and ask, "What is returned by this method?" (See appendix B for worksheets and notes.)
14 (32–36)	**Review** **Topics:** • Review AP Computer Science A topics **Objective:** • Prepare for the AP CS A Exam by reviewing material and taking practice exams	**Resources:** • Previous free-response questions from AP Central • Barron's test-prep book for the AP Exams **Assessments:** Practice exams **Strategies:** Give as many practice exams as possible.

Teaching Strategies

I try to create a learning environment that is comfortable for all students. Those who have never touched a computer should be as at ease in my class as those who have taught themselves how to program. I aim to foster critical thinking, a lifelong skill, and I accomplish this by giving challenging, yet not impossible, assignments. When new topics are introduced, I like to use a hands-on approach of having students see and run examples. While the novices ask questions, more experienced students can make changes to the examples and experiment with different outcomes.

I measure the effectiveness of my teaching by monitoring student work and giving many short quizzes, in order to gauge students' knowledge before I test them on a topic. Experienced programmers help the novices in a mentoring program after school. This promotes student leadership and propels in-class learning.