AP® Computer Science AB Syllabus

Course description:

This course is comprised of college material, and is conducted in a rigorous manner. It is designed for students who have a strong background in mathematics and interest in the field of computer science.
The course emphasizes proper programming methodology, algorithmic development, data structures, and object oriented programming (OOP) and design. Students will become proficient in creating and implementing classes. This includes creating new classes using inheritance and implementing interfaces.

Topics studied in this course include a review of computer platforms and architecture; objects and primitive data; program statements; writing and enhancing classes; inheritance; one and two-dimensional arrays; advanced data structures such as: trees, linked lists, sets, and maps; recursion; and searching and sorting techniques. A detailed scope and sequence is shown below. Every area that is referenced in the AP® Computer Science AB Course Description is studied.

Students will be expected to use their knowledge of proper programming and design techniques when creating programs in this course. Programs are implemented using the JAVA programming language. Upon completion of the course, students will be able to take the College Board AP® Computer Science AB exam.

All AP® Computer Science classes are taught in a computer lab. Each student has a computer workstation. Lessons are taught using Socratic methods. Portions of programs are presented and discussed. Students create sample classes with the instructor. About two-thirds of each class period is spent on the computer.

Major Texts

J. Lewis, W. Loftus, and C. Cocking, *Java Software Solutions for AP Computer Science*, 1st ed., Prentice Hall, 2004.

K. Lambert and M. Osbourne, *Fundamentals of Java, Comprehensive Course*, 2nd ed., South-Western, 2003.

Wu, C. Thomas. *An Introduction to Object –Oriented Programming with Java* –2nd ed. New York, NY: The McGraw-Hill companies, Inc., 2001

College Board. *AP GridWorld Case Study*. New York: College Entrance Examination Board, 2006.

K. Lambert and M. Osbourne, *Activities Workbook for Fundamentals of Java*, South-Western, 2003.

Teukolsky, Roselyn. *Barron's AP® Computer Science Levels A and AB 2007,* 3rd ed. Hauppauge, N.Y.: Barron's Educational Series, 2006.

S. Horowitz, *Addison-Wesley's Review for the AP® Computer Science Exam in Java*, Addison-Wesley, 2004.

Course Planner

The resource list includes the following text references *Fundamentals of Java* (FJ), *Java Software Solutions for AP Computer Science* (SS), *An Introduction to Object –Oriented Programming with Java –Thomas C. Wu* (TCW)

Unit (Weeks)	Title, Topics and Student Objectives	Resource, Assessment, and Strategies
1 (0-2)	**Thomas C. Wu** *(Introduction to Object-Oriented Programming and Software Development)* **Topics: [C1], [C2], [C7]** • Classes and Objects • Messages and Methods • Class and Instance Data Values • Inheritance **Objectives:** • Name the basic components of object-oriented programming. • Differentiate classes and objects. • Differentiate class and instance data values. • Draw object diagrams using icons for classes and objects • Describe the significance of inheritance in object-oriented programs.	**Resources:** • TCW – Lesson 1 **Assessment:** • Draw an object diagram for a Person class and two Person objects. • Draw an object diagram for Account Object with instance methods deposit and withdraw. • Draw object diagram with instance variables. • Vocabulary Inheritance, superclass, subclass. • Exercise 1.7 pg 34
1 (3)	**Fundamentals of Java** *(First Java Programs)* **Topics: [C4], [C5]** • The Java Virtual Machine and Byte Code • Hello World • Edit, Compile and Execute • Turtle Graphics **Objectives:**	**Resources:** • FJ – Unit 1 – Lesson 2 **Assessment:** • Write a program that displays you name, address and telephone number • Write a program that displays a yield sign enclosed in a triangle. • Write a program that calculates and

	• Explain the Java virtual machine and byte code. • Describe the structure of a simple Java Program • Edit, compile, and run a program using a Java development environment. • Understand compile-time errors. • Write a simple turtle graphics program.	prints the number of minutes in a year • Write a program that draws two vertical line segments showing Muller-Layer Illusion.
1 (4)	**Fundamentals of Java** *(Syntax, Errors and Debugging)* **Topics: [C4] , [C5]** • Language Elements • Basic Java Syntax and Semantics • Terminal I/O for Different Data Types. • Comments • Programming Errors • Debugging **Objectives:** • Construct and use numeric and string literals. • Name and use Variables and constants. • Create arithmetic expressions. • Concatenate two strings or a number and a string. • Know how to use comments in a program. • Tell the difference between syntax errors, runtime errors and logic errors.	**Resources:** • FJ – Unit 1 – Lesson 3 **Assessment:** • Write a program that takes the length of an edge as input and prints the cube's surface area as output. (Remember to analyze, design, implement and test) • Write a program that prompts user for input of regular and over time hours. After that computes employee's total weekly pay. • Use the TurtleGraphics package to display the employee's wages in a histogram. After taking inputs and computing results, your program should create a pen that displays five horizontal lines of varying length.
2 (5-6)	**Fundamentals of Java** *(Introduction to control Statements)* **Topics:[C1]** • Addition Operators	**Resources:** • FJ – Unit 1 – Lesson 4, Unit 2 Lesson 6 **Assessment:**

	• Standard Classes and Methods • The If and if-else statements. • The while, the for statement. • Nested Control Statements and the break Statements. • Errors in Loops • Logical operators • Nested if statements. **Objectives:** • Use the increment and decrement operators. • Use if and if-else statements to make choices. • Use while and for loops to repeat a process. • Construct appropriate conditions for control statements using relational operators. • Detect and correct common errors involving loops. • Construct complex Boolean expressions using the logical operations. • Understand the logic of nested if statements and extended if statements.	• A 2-minute telephone call to Lexington, Virginia, costs $1.15. Each additional minute costs $0.50. Write a program that takes the total length of a call in minutes as input and calculates and displays the cost. • Write a program that prompts user for input of base integer number. Then prompts user to enter exponent, the program displays base number to the power. The program halts when the user enters -1 for the power. • Project Unit 1-4 pg 128 • Written questions pg 121
2 (7)	**Fundamentals of Java** *(Introduction to Defining Classes)* **Topics: [C1] , [C2], [C4], [C6]** • The internal structure of Classes and Objects • The Structure and Behavior of Methods. • Scope and Lifetime of Variables. **Objectives:** • Design an implement a simple class from user requirements. • Use visibility modifiers to make methods visible to clients and restrict access to data within a class. • Use instance variables, local variables and parameters appropriately.	**Resources:** • FJ – Unit 2 – Lesson 5 **Assessment:** • Modify the smiling faces program of this lesson by completing the method drawLine. Add the instance variable radius and color and the mutators setRadius and SetColor. • Develop a new class called Car. A car displays two wheels, a body (a long rectangle), and a passenger compartment (a shorter rectangle). Be sure to include the appropriate helper methods. Test-drive your new class in an interface similar to that of the smiling faces program.
3	**FJ – Arrays, Recursion, and Complexity**	**Resources:**

(8-9)	*(Introduction to Arrays)* **Topics: [C1] , [C2], [C4]** • Conceptual Overview • Simple Array Manipulations • Looping through Arrays • Declaration Arrays • Parallel Arrays • Two-Dimensional Arrays • Arrays and Methods • Arrays of objects **Objectives:** • Write programs that handle collections of similar items. • Declare array variables and instantiate array objects. • Use loops and write methods to manipulate arrays. • Create parallel arrays and two-dimensional arrays.	• FJ – Unit 3 – Lesson 8 **Assessment:** • Write a program that takes 10 floating-point numbers as inputs. The program displays the average of the numbers followed by all of the numbers that are greater than the average. As part of your design, write a method that takes an array of doubles as a parameter and returns the average of the data in the array. • Modify the program of Project 8.4 so that it displays not only the median and mode of the list of numbers but also a table of the numbers and their associate frequencies.
(10-11)	**Grid World Case Study** *(parts 1-4)* **Topics: [C6]** • GridWorld Case Study part 1 – 4 • Work on GridWorld Case Study part 1-4 **Objectives:** • Review Testing • Understand the Bug class, Runner class, Grid interface. • Extend the Bug class by creating a specialized bug with new characteristics. • Class Modifications • Inheritance	**Resources:** • Grid World Case Study part 1 - 4 **Assessment:** • Exercise from within the case study. • Read the GridWorld case study thoroughly and get familiar with classes, methods and interfaces.
3 (12)	**FJ Arrays, Recursion and Complexity** *(Classes)* **Topics: [C1], [C6]** • Class (static) Variables and Methods. • Code Reuse through Inheritance • Inheritance and Abstract classes • Acceptable Classes for Parameters and Return Values. • Exceptions **Objectives:**	**Resources:** • FJ – Unit 3 – Lesson 9 **Assessment:** • Design a hierarchy of classes that represents the classification of artifacts, such as vehicles. • Add a method perimeter to the shape hierarchy from this lesson. This method should return the circumference of a circle and a wheel and the perimeter of a rectangle.

	• Know when it is appropriate to include class (static) variables and methods in a class. • Understand the use of inheritance by extending a class. • Understand the use of polymorphism and know how to override methods in a superclass. • Understand the implications of references types of equality, copying and mixed-mode operations.	• Using Compute Weekly Pay Case Study form this lesson, modify the payroll system of the case study so that taxes are deduced from gross pay. The tax rate is a flat 10 percent.
3 (13)	**FJ Arrays, Recursion and Complexity** *(Arrays)* **Topics: [C1], [C2]** • Advance operations on Strings • Searching Arrays • Sorting Arrays • Insertions and Removals • Working with Arrays of Objects • The Class java.util.ArrayList **Objectives:** • Use string methods appropriately. • Write a method for searching an array. • Write methods to perform insertions and removals at given positions in an array. • Understand the issues involved when working with arrays of objects.	**Resources:** • FJ – Unit 3 – Lesson 10 **Assessment:** • Write a program that inputs 10 integers into an array, sort the array with a selection sort, and displays its contents before and after the sort. • Modify Student Test Score Case Study for this lesson so that it displays a list of the words in the text area and their associated frequencies. A word's frequency is the number of times it appears in the text. You should use an array of String of the worlds and a parallel array of int for the frequencies. The program should sort the list of worlds before output. If the number of unique worlds exceeds 50, sop the processing of input and display a message in the message box.
3 (14)	**FJ Arrays, Recursion and Complexity** *(Recursion, complexity and Searching and Sorting)*	**Resources:** • FJ – Unit 3 – Lesson 11 **Assessment:**

	Topics: • Recursion • Complexity Analysis • Binary Search • Quicksort **Objectives:** • Design and implement a recursive method to solve a problem • Understand how computer executes a recursive method. • Understand the behavior of a complex sort algorithm such as the quicksort.	• Write a recursive method that returns a string with the characters in reverse order and test the method with a Tester program. • Write a tester program to help assess the efficiency of the Tower of Hanoi program. This program should be similar to the one developed from the Fibonacci method.
4 (15-18)	**FJ Using Abstract Data Types** *(Object Oriented Analysis and Design, Linear Collections: Lists and Stacks, Unordered collection Sets and Maps)* **Topics: [C1], [C2], [C4]** • Analysis , Design and Implementation • Overview of Collections • Lists and Iterators • Stacks • Queues and Priority Queues • Sets and sorted Sets • Maps and sorted maps **Objectives:** • Understand the general role of analysis and design in the software development process. • Understand the difference between aggregation, inheritance and other relationships among classes. • Distinguish fundamental categories of collections, such as linear, hierarchical, graph and unordered. • Understand the basic features of lists and their application • Recognize the difference between index-based operations and content-based operations on lists. • Understand behavior of a stack and recognize applications in which a stack would be useful.	**Resources:** • FJ – Unit 4 – Lesson 12, 13, 14, 15 **Assessment:** • Modify case study Maintain a List of Tasks from this lesson so that it allows the user to scroll through all the tasks in a message box. When the user selects a new button named View All, the TaskListView responds by sending the message toString to the TaskListModel, which returns a string consisting of all the tasks. This string is then displayed in a message box. • Write a program that tests its inputs for palindromes. (Hint: Use a string tokenizer and a stack.) • Write a program that maintains a wait list of students for admission to a college course. Seniors are given first priority, followed by juniors, sophomores and freshmen. The program should allow the user to perform the following tasks: a. Enter a Student's name and class, adding that student to the wait list b. Remove the next student from the wait list c. Clear the wait list. When any of these operations is performed, a display is updated with

	• Understand the behavior of a queue and recognize application in which a queue would be useful.	the appropriate information. The structure and user interface of this program can be similar to that of the emergency room scheduler.
(19-20)	**GridWorld Case Study** *(part 5)* **Topics: [C6]** • Explain girds data structure **Objectives:** • Understand the BoundedGrid Implimentations. • Consider alternate implementation for BoundedGrid class • Understand implementation of UnboundedGrid class.	**Resources:** • GridWorld Case Study, The College Board **Assessment:** • New variations to the bug class. • Display of Classes extended from Critter class. • Understanding of grid data structure.
5 (21-22)	**FJ Implementing Abstract Data Types** *(Implementing Lists, Stacks, and Queues)* **Topics: [C3]** • Interfaces, Multiple Implementations and Prototypes. • The Fixed-Size Array Implementation of Indexed Lists • Complexity Analysis of Indexed List Implementation • Fixer-Size Array Implementation of Positional Lists • Iterators • Implementation of Stacks and Queues **Objectives:** • Use an array to implement an indexed list. • Use an array to implement positional list. • Use arrays or linked structures to implement stacks and queues. • Understand the run-time and memory trade-offs of array-based and link-based implementations of linear collections.	**Resources:** • FJ – Unit 5 – Lesson 16 **Assessment:** • Complete the classes FSAIndexedList, SLIndexedList, FSAPositionalList, DLPositionalList and test them with an appropriate tester program. • Add an iterator method to the interface IndexedList. Then implement this method in the class FSAIndexedList using the techniques described in the lesson. Test you iterator with an appropriate tester program.
5 (23-24)	**FJ Implementing Abstract Data Types** *(Implementing Sets and Maps)* **Topics: [C1] , [C3]**	**Resources:** • FJ – Unit 5 – Lesson 17 **Assessment:**

	• The Set and Map Prototypes • List Implementation of Sets and Maps • Hashing Implementation of Maps and Sets. **Objectives:** • Explain why a list implementation of sets and maps is simple but inefficient. • Develop hash function to implement sets and maps. • Understand why a hashing implementation of sets and maps can be very efficient.	• Modify the program TESTCODES from this lesson to implement the linear method of resolving collisions. The code for this method should be packaged in a private Java helper method. Use an array of sufficient capacity to observer any clusters that develop. Track and display the number of comparisons required to resolve collisions as well. Test this method will load factors of .33, .5, .75 and 1.
5 (25-26)	**FJ Implementing Abstract Data Types** *(Implementing Trees and Priority Queues)* **Topics: [C1] , [C3]** • An overview of Trees • Binary Tree Traversals • A linked Implementation of Binary Trees • An Array Implementation of Binary Trees • Implementing Heaps **Objectives:** • Use the appropriate terminology to describe trees. • Distinguish different types of hierarchical collections, such as general trees, binary search trees, and heaps. • Use binary search trees to implement sorted sets and sorted maps. • Use heaps to implement priority queues.	**Resources:** • FJ – Unit 5 – Lesson 18 **Assessment:** • Write the implementation of a sorted set class using a binary serch tree. Your solution should consist of an interface, SortedSetPT, an implementing class, TreeSetPT, and a tester program. You should decide wich methods should be included in the sorted set's interface. Be sure to throw an appropriate exception if a client's item is not comparable.
6 (27)	**FJ Graphics, Files, Apples and Swing** *(Files)* **Topics: [C1], [C6]** • File Class • File input • File output • Other Input/Output Situations **Objectives:** • Understand the use of the classes for the processing files in the	**Resources:** • FJ – Unit 6 – Lesson 20 **Assessment:** • Write a program that reads names from a text file. The names are separated by new line characters, are in sorted order, and some are duplicate names. The program should write the names to a different text file without the duplicate names. Allow the use to

	java.io.package. • Open and use input file streams for reading one character, one line, or one work at a time from text files. • Use data input and output streams for the transfer of specific types of data to and from files. • Use file dialogs to open connections to input and output files.	specify the file names in text fields. • Assume that a text file or names has each name on a separate line and that names are in random order. Write a program that inputs the names, sorts them, and write them back to the same file.
(28-31)	**Preparation for the AP Exam** **Topics: [C6], [C7]** • Revisit GridWorld Case Study • Appropriate free-response • Appropriate multiple choice answers. **Objectives:** • Review GridWorld Case study and gain complete understanding of it's classes and methods. • Review multiple choice and open ended questions.	**Resources:** • Barron's AP® Computer Science Levels A and AB • Review for the AP® Computer Science Exam in Java **Assessment:** • Exercise from the text • Previous free response questions from AP Central. • Practice Exams
6 (32-33)	**FJ Graphics, Applets and Swing** *(Simple Two-Dimensional Graphics, Introduction to HTML and Applets, Swing and AWT)* **Topics: [C1], [C6]** • The Conceptual Framework for compute Graphics. • The Method repaint • The Method getGraphics • Responding to Mouse Events. • Simple Text elements • Linking to other documents. • Applets. • The Swing and AWT philosophy • GUI components. **Objectives:** • Understand difference between Cartesian coordinates and screen coordinates. • Use methods to draw images in two-dimensional graphics. • Create a program that graphs data with charts. • Use the appropriate markup tags to include images in Web pages.	**Resources:** • FJ – Unit 5 – Lesson 18 **Assessment:** • Write a program that allows the user to change the pen color and the background color of the application window and to view its size. The program should provide two menus of colors. When the user selects a color from the Pen menu, the program sets the window's graphics context to that color. When the user selects a color from the Background menu, the program sets the window's background to that color. The program also displays the height and width of the window's graphics context at the center of the window. These values should be updated whenever the user resizes the window. • Redo the Fahrenheit to Celsius temperature conversion application with Swing/AWT so that your program adheres to the

	• Convert Java applications to an applet and embed the applet in a web page. • Understand how to structure a GUI-based program using Java's Swing and AWT packages.	model/view/controller pattern as described in the case study.
(34-36)	**Final Group Project** **Topics: [C6], [C7]** • Will vary from Group to Group **Objectives:** • As part of there final grade students will work in groups to create an application, based upon their interest and teacher's approval.	**Resources:** • Workbook for Fundamentals of Java **Assessment:** • All steps taken to create the application. • Snap of final run of student application.

[C1] The course teaches students to design and implement computer-based solutions to problems in a variety of application areas.

[C2] The course teaches students to use and implement commonly used algorithms and data structures.

[C3] The course teacher students to develop and select appropriate algorithms and data structures to solve problems.

[C4] The course teaches students to code fluently in an object-oriented paradigm using the programming language Java. The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendixes A and B of the AP Computer Science Course Description.

[C5] The course teaches students to identify the major hardware and software components of a computer system, their relationship to one another, and the roles of these components within the system.

[C6] The course teaches students to read and understand a large program consisting of several classes and interacting objects, and enables students to read and understand the current AP Computer Science Case Study posted on AP Central.

[C7] The course teaches students to recognize the ethical and social implications of computer use.

Computer Facilities:

There are two computer labs available to students enrolled in computer science courses. Each lab contains 20 PCs. Each lab also contains scanners, CD-RW drives, a data projector, and laser printers.

Each student has an account on a networked server. They store their program files in their individual accounts. They have access to a public drive for instructor-supplied programs and files. The computer labs are open before and after school and during some lunch periods.