AP® Computer Science A Syllabus

Syllabus Number:

**Course Philosophy and Overview**

The primary goal of AP Computer Science class is to challenge and improve the students’ critical thinking and problem solving skills through the Java programming language. The required Java skill sets are documented in the AP® Java subset, which is found in Appendices A and B of the AP Computer Science Course Description. The course will follow an object-oriented approach to learning Java. Major topics include class construction, recursion, inheritance and polymorphism, as well as significant investigation of a variety of searches (binary and sequential) and sorts (merge, insertion, selection).

The acquisition of these programming skills will be demonstrated through a myriad of assessments. Students will be required to produce flowcharts to show the flow of their program logic. They will perform peer assessments to discuss quality and efficiency of their classmates’ program designs. Students will predict program output given written code. However, the most common assessment students will see is the actual creation of Java programs.

One particularly powerful resource frequently used is LiveLab. Each student is given an account that is tied to a class, tied to my instructor account. Through LiveLab, I can assign program that students upload their code (.java files) through the LiveLab website. The website will compile and execute their program using test data of my choosing and provide immediate feedback as to whether the program is fully functional. I can also create online object quizzes and tests through LiveLab to make sure the students are acquiring the proper programming vocabulary and concepts.

Students will be given programs of varying degrees of size and rigor. I believe it important to give students several different formats for the required programs. From timed Java programming competitions to cooperative group projects, students will learn to work collaboratively, as part of a team, but will also have significant individual responsibility for programs.

My goal for this AP Computer Science course is to challenge students to be active learners and critical thinkers. I follow the “guide on the side, not the sage on the stage” approach, which gently introduces the students to each new concept, encouraging student-driven learning via questioning and experimentation. I believe in hands-on learning. In addition to working on programs at home, students receive instant feedback during in-class lab time once or twice a week. During lab classes, I can easily look at each student’s code, informally tracking his or her progress, and assisting accordingly. I can talk to students about their programs, and they can ask specific questions about problems they are having.

Our class meets in a computer lab, which gives each student constant and immediate access to Java and the Eclipse IDE. Students use a combination of our textbooks, the GridWorld case student, current events, periodicals and a variety of other materials, to facilitate learning.

**Textbooks and Resources**

Liang, Daniel. Introduction to Java Programming. (9th edition). Upper Saddle River, NJ. Prentice Hall, 2012.

Horstmann, Cay. Big Java. (2nd Edition). Hoboken, NJ. John Wiley & Sons, Inc. 2006.

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

**General Course Overview**

Our AP Computer Science course meets for the entire school year, which is composed of three 12 week trimesters. The follow table shows the curricular course our program follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit #** | **Topic** | **Weeks** | **AP Computer Science**  **Scoring Components Focused** |
| 1 | Introduction to Computer Science | 1 | [C6], [C8] |
| 2 | Java Program Fundamentals | 2-7 | [C3], [C4] , [C6] |
| 3 | Strings | 8 | [C4], [C6] |
| 4 | Methods | 9-10 | [C3], [C4], [C8] |
| 5 | Arrays and ArrayLists | 11-13 | [C2], [C3], [C4] |
| 6 | Searches and Sorts | 14-15 | [C2], [C3], [C8] |
| 7 | Coding Efficiencies | 16 | [C4] |
| 8 | Object Composition | 17-18 | [C5], [C7] |
| 9 | Object Design | 19-20 | [C5], [C6] |
| 10 | Inheritance, Polymorphism, Interfaces | 21-22 | [C5], [C6] |
| 11 | Recursion | 23-24 | [C2], [C3], [C4] |
| 12 | GridWorld Case Study | 25-27 | [C5], [C7] |
| 13 | AP Test Review, AP Test | 28-30 |  |
| 14 | Final Project-Build and Presentation | 31-36 |  |

**Detailed Course Outline**

The following table covers five pages and shows the specific topics covered each week and the related programming assignments. The AP Computer Science Scoring Components addressed each week is located in the last column.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Weeks** | **Unit** | **Topic(s)** | **Programming Assignments [C1]** | **AP Comp.**  **Satisfied** |
| **1** | 1 | * Ethics– Privacy, Piracy, Property * Hardware and Software Components * Java Language Specification (API, JDK) * Discuss and distribute *AP Java Subset* * Introduction - Eclipse IDE * Parts of a Java program (import, class statement, main method, etc..) * Indents and Block Styles * Javadoc comments - @param, @return * Compiling & Executing a Java Program * *System.out.println()* – Console output * Types of Errors (syntax, runtime, logic) | HelloWorld  MyName – Outputs full name  ASCIIArt - Using *println* statements  ApproximatePI (Output 22/7)  OutputArea (Static formula) | **[C6]**  **[C8]** |
| **2** | 2 | * Input from Console (*Scanner* class) * Primitive data types (int vs. *double*) * Boundaries of types *int* and *double* * Constants (*final* keyword) * Naming Conventions (what and why) * Math Operations & Operator Precedence * Increment and Decrements (++ and --) * *Math* class (*pow, sqrt, abs, random*) * *Null* reference * Exception Types | ComputeArea (based on user input)  ComputeAverage(3 numbers  SalesTax  PayCheckv1-No overtime  FahrenheitToCelsius  ComputeLoan(Formula uses Math.pow)  ComputeChange(Uses % operator)  CalculateRunwayLength  CalculateBMI  CalculateHypotenuse | **[C3]**  **[C4]**  **[C6]** |
| **3** | *2* | * *Math* Class & related functions * Base Conversion * Binary and Hexadecimal * Primitive data type Conversions (Casting) | ConvertToBinary – From Decimal  ConvertFromBinary – To Decimal  ConvertToHex – From Decimal  ConvertFromHex – To Decimal | **[C3]**  **[C4]**  **[C6]** |
| **4** | *2* | * *boolean* primitive data type * *char* primitive data type * Counters * Condition *statements (if, else, else if)* * Nested *if* statements | Paycheckv2 – Rate increase for 40+ and 80+  OutputLetterGrade – Given percent (else if)  ComputeTax(tax rate changes with income)  LeapYear – Is inputted year a leap year?  SortThreeNumbers – Sort 3 integers  ZellersCongruence(given date, finds day of week) | **[C3]**  **[C4]**  **[C6]** |
| **5** | *2* | * Logical Operators (&&, ||, ^, !) * Truth Tables * De Morgan’s Law * *Random* Class | MathQuiz – 5 question random addition quiz  RockPaperScissors(Random, multiple *if*s)  QuadraticFormula – Calculates root(s) | **[C3]**  **[C4]**  **[C6]** |
| **6-7** | *2* | * *while* Loops * *for* Loops * Nested Loops * Sentinels * *printf* statement – Output formatting * Escape Sequences ( \\, \” and \n) * *break* and *continue* keywords | GuessingGame – Guess random number 1 to 100  MultiplicationTable – Outputs formatted 1 to 9  GreatestCommonDivisor(Given 2 integers)  PrimeNumbers – Outputs all primes 1 to 100  OutputPatterns – Nested for loops  FindPIN – Brute Force a PIN number - Ethics  PerfectNumber – Output perfect numbers <10000  DecimalToBinary – Converts base 10 to base 2  **Project-AlgebraTest** – User is given random algebra test (10 ?s) – Questions are in the format 5x -10 = 20. Letter grade is given at end. | **[C3]**  **[C4]**  **[C6]**  **[C8]** |
| **8** | 3 | * The *String* class * Further *char* type analysis * Immutable Concept * Concatenation of Strings * Substrings * Built-in String Methods * Using .*equals* method to Compare Strings * Object Equality vs. Object Identity * Converting to and from Strings | FullName – Concats First, Middle and Last Names  Palindrome – Checks if a String is a palindrome  CheckSSN – Determines if given SSN is valid  Telephone – Letters are converted to numbers (Ex: 1-800-FLOWERS = 1-800-3569377)  CheckPassword – Password criteria must be met | **[C4]**  **[C6]** |
| **9** | 4 | * Defining a Method * Modular Programming Benefits * Functional Decomposition * Calling Methods * Method Returns * Void methods * Passing Parameters – By value/reference * Method Abstraction (Encapsulation) * Static Methods | SumDigits- sum of 2 integers is returned  MeterToFoot – Converts meters to feet  PalindromicPrime-Given an integer, call two methods to see if it is a palindrome and prime  CreditCardValidation – Use the Luhn validity check formula to determine if user entered cc# is valid  Computer Ethics paper – We use the obvious application of our CreditCard program to discuss ethical use of computer programming  TwinPrimes –prime pairs (2 difference) < 1000  Emirps - Find primes whose reverse is also prime. Ex: 17 and 71 | **[C3]**  **[C4]**  **[C8]** |
| **10** | 4 | * Overloading Methods * Private vs. Public Methods * Method Abstraction (Stepwise Refinement) * Scope of Variables | Craps – Dice game simulation  TemperatureConvert – Menu program converts between Fahrenheit, Celsius and Kelvin  Java Programming Competition-4 Programs given to groups of 4 – Have 75 minutes to build as much as possible(After we discuss collaboration)  **Project – Blackjack** | **[C3]**  **[C4]**  **[C8]** |
| **11** | 5 | * Single Dimensional Arrays * Traversals, insertions, deletions * Passing Arrays to Methods * Returning an Array from a Method * *Arrays* class and associated methods * Parallel arrays | DeckOfCardsv1 – No repeats  Lotto – Pick and save 6 nonrepeated random ints  EliminateDuplicates – Deletes repeated values in an array of random integers  Poker – Write methods to calculate a poker hand.  EightQueens – 8 queens on board w/o attack  PlinkoSimulation – Galton box simulation  LockerPuzzle | **[C2]**  **[C3]** |
| **12** | 5 | * 2 Dimensional Arrays * 3D Arrays and Beyond * Nesting for loops to populate 2D arrays * 1st Tri - Final Exam Review and Final Exam | ConnectFour – 2D String and 2D boolean arrays  MarkovMatrix – Sum of all columns is 1  DungeonLevel Discussion – 3D Array utility  Substituion – Replacing values in a char array | **[C2]**  **[C3]**  **[C4]** |
|  |  |  |  | **Trimester 2 Begins** |
| **13** | 5 | * ArrayLists – Holds Objects, Unbounded * ArrayList methods(*get, set, size, add, remove)* * Arrays vs. ArrayLists | CompareAndContrast – Arrays vs. ArrayLists  FindLargest – Finds largest integer in an ArrayList  **Project-TicTacToe** – Class Contest to Build the best TicTacToe Artificial Intelligence (AI) | **[C2]**  **[C3]**  **[C4]** |
| **14** | 6 | * Searching Arrays * Binary Search * Sequential Search | BinarySearch Logic – Document variable values  BinarySearch – find a specific name in an array  SequentialSearch – find a specific value in an array  SearchPredictions – Count iterations to pick the more efficient search method | **[C2]**  **[C3]** |
| **15** | 6 | * Sorting Arrays * Merge Sort * Insertion Sort * Selection Sort | SortLogic – Follow each sorts logic step by step  SelectionSort – Sort an array of random doubles  InsertionSort – Random insert 100 prime numbers into an array, then sort with insertion sort  MergeSort – Sort an array of Strings  SortPredictions – Count iterations to pick the more efficient sort method | **[C2]**  **[C3]** |
| **16** | 7 | * Big Oh Concept * Logarithms * Run time tests on each search and sort * Calculate number of loop iterations for each search and sort studied * Revisit *break* statement to reduce wasteful looping | \***Class contest - Palindrumber** – Find the smallest paldindrome number that is the product of 3 three digit numbers as quickly as possible (fewest loop iterations) | **[C4]** |
| **17** | 8 | * Objects Introduction * Defining Classes for Objects * Constructors * Instantiation * Instance vs. Static Data Fields * Instance vs. Static Methods * Packages – Portability concept & import | Circle – Create the class with getArea and getCircumference methods.  Student – Create a student class and instantiate it.  TV – Create a TV class with a variety of “controls”  DeckofCardsv2 – Design Card class and populate an ArrayList with all 52 cards and no repeats  **Project Hero**-Design a *hero* class for an RPG. | **[C5]**  **[C7]** |
| **18** | 8 | * Test Classes (Driver Programs) * Data Field Encapsulation (*private*) * Accessor and Mutator Methods * Passing Objects to Methods * Arrays of Objects | TenCircles – Define circle class, then instantiate 10 circle object with random radius, then calculate the sum of all ten circles’ areas.  Computer Class – Data Fields are the computer’s components (processor, RAM, etc..)  PokerHands – Deal out four hands of poker, using the *Card* class, no repeats | **[C5]**  **[C7]** |
| **19-20** | 9 | * Class Design Guidelines * Immutable Objects * *this* reference * Data Abstraction and Encapsulation * Object-Oriented Design | Polygon Class – getArea for any # of sides  BankAccount – Several data fields and methods | **[C5]**  **[C6]** |
| **21** | 10 | * Inheritance * Class Hierarchy * Superclasses and subclasses * *super* Keyword * Overriding Methods * Overriding vs. Overloading * Wrapper Classes (Integer and Double) * Boxing and unboxing | Develop hierarchy of classes stemming from the *person* superclass. Ex: student, teacher  *shape* superclass  *circle* subclass – inherits shape , defines its own area method  square subclass – inherits shape, defines its own area method | **[C5]**  **[C6]** |
| **22** | 10 | * The Object class * Polymorphism * Dynamic Binding * Casting Objects * Protected Data and Methods * Abstract Classes * Interfaces * Comparable Interface * *List* interface (implemented by ArrayList) | *AbsoluteValue* - works for *int* and *double*  *Family* Class – various subclasses include mom, dad, brother, sister, etc.. Create a protected method in *family* that only allows its subclasses to access its methods.  Shapesv2 – Polymorphic calls to shape superclass from circle and rectangle subclasses | **[C5]**  **[C6]** |
| **23-24** | 11 | * Recursion * Recursive Helper Methods * Problem Solving with Recursion * Recursion vs. Iteration * Random Mazes with Recursion * Final Exam Review and Final Exam | * Factorials – Iterative vs. Recursive * Fibonacci – Generate the sequence * RatInMaze – Using a recursive Method – Find a path out of maze * Recursive BinarySearch * Recursive SelectionSort * FloodFill | **[C2]**  **[C3]**  **[C4]** |
|  |  |  |  | **Trimester 3**  **Begins** |
| **25** | 12 | * GridWorld Introduction (Part 1) * GridWorld Continuation (Part 2) | * Examine Default classes, methods * Instantiation and Testing of classes * Bug Variations | [C5]  [C7] |
| **26** | 12 | * GridWorld Continuation (Part 3) | * Critter Design and Implementation * Implementing Grid interface | [C5]  [C7] |
| **27** | 12 | * GridWorld Continuation (Part 4) | * Use inheritance to produce new Critters | [C5]  [C7] |
| **28-29** |  | * Review for AP exam * Practice AP Exam - Objective Questions * Practice AP Exam – Case Studies |  |  |
| **30** |  | * Take AP Exam * Choose Final Project |  |  |
| **31-34** |  | * Work on Final Project |  |  |
| **35** |  | * Present Final Project |  |  |
| **36** |  | * Final Exam Review and Final Exam |  |  |
|  |  |  |  | **End of Year** |

**AP Computer Science Curricular Requirement Evidence [C1] through [C8]**

\*Note: All keywords specified in the Syllabus Development Guide are underlined in each Evidence section

**Scoring Component 1 [C1]** - The course teaches students to design and implement computer-based solutions to problems.

**[C1] Evidence**

The Programming Assignments Column in the Course Outline lists the numerous assignments (with descriptions) my students are required to complete for each unit. I have cross-referenced the units covered with those lists in the current AP Computer Science Course Description, from the AP College Board website. Every unit has several opportunities for the students to implement and design object-oriented constructs using the various data structures and algorithms specified in the AP Computer Science Java Subset.

**Scoring Component 2 [C2]** - The course teaches students to use and implement commonly used algorithms.

**[C2] Evidence**

Depending on the unit, I give my students several different program requirements. In most cases, the students will start “from scratch” and design and implement their solution entirely on their own. Examples include the *GuessingGame* and *AlgebraTest* projects.

There are also several programs in which the students are given a “starting point” and they must complete, troubleshoot or create an additional feature for those programs. An example of this would be the *Poker* program. In this program, I give the students a program that deals out 5 random cards from a deck. Students are then tasked with writing methods to calculate as many possible poker hands, such as pairs, flushes, straights, etc… To do this, they must be able to understand and use my existing *card* class and its associated methods.

Students will also have assignments which make them analyze and develop code without the benefit of a computer. For example, when discussing the binary search method, students are asked to calculate and document the values of all variables utilized as an array is traversed and a specific value is sought. Students will be asked to make predictions about which searches and sorts are more efficient before we discuss such topics as “Big O” and logarithms. We will then code each search and sort and test each (with varying sized arrays and ArrayLists) and document the time each takes. We also revisit the binary search and selection sort in week 23, when we discuss and implement recursive versions of this programs.

**Scoring Component 3 [C3]** - The course teaches students to use and implement commonly-used data structures

**[C3] Evidence**

In weeks 11 to 13, our topics are focused on arrays and ArrayLists. We focus on both single dimension, as well as two-dimensional arrays. The students will implement a variety of programs that make use of these data structures. For example, in our *Lotto* program, students will create a program that takes 6 user inputted numbers and saves them into an array. A second array is created and filled with 6 random values. The students will then use a looping structure of their choosing to compare the values in each array to see how many lotto numbers they matched.

On most assignments, students are allowed to choose the data structure they will implement. However, they soon learn how to best to choose between arrays and ArrayLists. For example, there are two versions of DeckOfCards the students will implement. (Weeks 11 and 17) The first version has the student store a cards value using integers in an array. The second version has students first creating a *card* class, then in a separate driver program, they populate an ArrayList with instances all 52 possible cards. We examine how to randomly shuffle the elements of both arrays and ArrayLists and how to remove elements from each, which helps students for when they must write a program to deal out four hands of poker.

**Scoring Component 4 [C4]** - The course teaches students to select appropriate algorithms and data structures to solve problems.

**[C4] Evidence**

Much of the evidence requested is provided in my C2 evidence, which focuses on the various searches and sorts and how students must, on paper and without Java, calculate the number of loop iterations and variable value changes that occur over the course of a specific search or sort.

In week 13, ArrayLists are introduced. Part of our initial discussion involves comparing the two and deciding when it is more advantageous to use one over the other. For example, if the number of elements to be stored is not static, an ArrayList would be favorable and would allow students to avoid “ArrayOutOfBounds” exceptions.

In week 23, recursion is introduced. One of the programs students are responsible for is a factorial calculator. We discuss the benefits and drawbacks to using recursion, based on the size of the integer be used.

**Scoring Component 5 [C5]** - The course teaches students to code fluently in an object-oriented paradigm using Java.

**[C5] Evidence**

Starting in week 17, students are introduced to objects and encouraged develop program solutions that embrace an object-oriented approach. Students are shown the benefits and modularly of developing multiple, portable class files and how we can “reuse” our class files, if they are thoughtfully designed. For example, the *card* class that we design, in isolation, in week 17 (DeckOfCardsv2) is later used in our *PokerHand* project in week 18.

We invest three weeks in the study of the GridWorld Case Study. In this time, there are numerous tweaks the students will be asked to perform within the Case Study. For example, in week 27, students will show their ability to use inheritance by extending Bug and Critter classes to create new versions of these superclasses.

All advanced topics are thoroughly examined in GridWorld. We examine the Grid Interface in Week 26. (Part 3). We also study polymorphism through the methods used. Given the class hierarchy of GridWorld and the fact that several methods are identically named in both super and subclasses (Examples include *act()* and *isValid*() ), students will have to understand how polymorphism works to understand the GridWorld construction.

**Scoring Component 6 [C6]** -The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendices A and B of the AP Computer Science Course Description.

**[C6] Evidence**

As mentioned in [C1], all classes, methods and interfaces listed in both Appendix A and Appendex B of the AP Computer Science Course Description, as well as those listed in the AP Computer Science Quick Reference Guide are specifically listed in the Detailed Course Outline and therefore covered over the course of the school year. The student application of these concepts is inherent in the various programs they write. These, again, are listed in the Programming Assignments column of the Detailed Course Outline.

**Scoring Component 7 [C7]** - The course teaches students to read and understand a large program consisting of several classes and interacting objects. In particular, the course enables students to read and understand the current AP Computer Science Case Study posted on AP Central.

**[C7] Evidence**

As previously mentioned in [C5], three weeks of class time (Weeks 25-27) is spent working with the GridWorld case study. Students will be comfortable working with multiple interdependent classes. They will know the inner workings of original code to the degree that they will be able to modify it to produce new, anticipated results. For example, students will create new Critters and modifying existing critters.

In addition to the time spent with GridWorld, time is also given, at the end of the course, for students to develop their own Java programming project. One of the many requirements of this project is the use of multiple classes.

**Scoring Component 8 [C8]** - The course teaches students to recognize the ethical and social implications of computer use.

**[C8] Evidence**

A portion of the first week is spent discussion the various ethical and social issues surrounding technology today. The students, in the first week, will choose from a variety of discussed topics (torrents (piracy), identity theft (privacy), plagiarism (intellectual property), etc…) to write a persuasive essay that details their opinion and ethical view.

At various time throughout the school year we will examine how new programming skill could be used in an unethical manner. For example, in week 9, students create a program called *CreditCardValidation*, in which they have to use the Luhn validity check formula to determine if a user-entered credit card number is valid. This prompt excellent discussion about how this could be used for illicit purposes, but more importantly how we can add additional safeguards to our programs to prevent such acts. We then discuss when Expiration Dates and CIV codes are also found on credit cards now.

We also discuss brute force hacking methods and how looping structures could be used to guess 4 digit pins. (Week 9)