AP Computer Science A
Syllabus

Computer Facilities

The course is taught in a computer lab. There are 24 student stations plus a teacher station. I use an overhead projector that projects onto a Smart board. I have a whiteboard at the front of the class and each student faces the front of the class. We have a school-wide network and each student has access to their own (h:) drive where they can store their personal files. Each student also has access to a shared (p:) drive in which they have access to resource files that I store on it. I also employ a program called Vision6, which is a classroom management program in which I can freeze their screens in order to teach a lesson.

Texts

Horstmann, Cay. *Java Concepts for AP Computer Science.* Hoboken, N.J.: Wiley, 2007.

Pre-AP Coursework

This course is the third in a sequence of computer science courses offered at our high school.

The first course, Computer Science 1, is a semester course where the students are introduced to the concepts of computer science. Currently, Alice 2.2 is the introductory language. The environment is drag-and-drop so the student doesn't have to deal with syntax. The students learn about variables such as numeric, String, Boolean, and objects. They also learn about conditionals such as if-then. They also learn about loop structures such as the for-loop and the while-loop. The student is introduced to the idea of an object and its properties. They also learn about methods and functions. The purpose of the course is to teach the student that the computer can be controlled by the programmer. They learn that program design becomes essential as programs increase in size. It is during this course that I teach basic hardware and software essentials, computer architecture, computer history, and binary numbers.

The second course, Computer Science 2, is a semester course where the students are introduced to the java language. Currently, DrJava is the IDE that I employ to teach them these basics. In this course I teach the students how to code by typing their code into the IDE. The concepts of variables, objects, methods, conditionals, loop structures are reviewed and extended from the Computer Science 1 course. I also teach them arrays. Proper programming conventions are taught early on and adhered to throughout the entire computer course sequence. I use swing GUI's primarily for input and output as students find this relevant, engaging and fun. The students extend their problem solving ability and learn the syntax of the Java language.

AP Computer Science is the third course in this sequence. It's a full year course and Computer Science 2 is a pre-requisite.

Teaching Strategies

I tend to teach my courses with a basic assignment outline but willingly adapt my lessons based on the interests of my students. My assignments easily vary from year to year since my students propose creative projects and I will try my best to accommodate these ever-changing interests.

The course is based solely on problem solving. What I mean by this is that I am constantly presenting "problems" in which either we or they have to solve. I begin each new idea by presenting a problem and this makes clear the reason *why* we need the concept that will follow. That is, I don't teach the topic and then explain why we need to know it. Every topic has a purpose *because* of the way that I teach. This technique has proven to challenge the students to think *on their own* as each student becomes quite comfortable with thoughts like "I wonder how would I do this…" or "I want to be able to that, I think I'm going to do …" This is an essential component of problem solving and design. When students begin designing their own classes and subclasses, it is vital that they can think critically. The project 12.1 "There's an App for That!" requires the student to create an entirely original app from start to finish.

For each topic, I lecture for a while and explain the basics of the concept. I then demonstrate how the code is written by using my computer and the overhead projector. Next, I have the students mimic me at their own computers, and then finally, I have them do an assignment in class. This is followed by further assignments that extend the concept. Most of the time, the assignments start very simply and then progress to something more challenging. Sometimes the assignments require the student to extend a class that has been written for them and other times the assignment requires them to create a class from scratch.

I like to have an ongoing larger project that we work on as a group. This is normally the place where we add the features that are inherent in the current concept. I like doing this because it's a way to incorporate new ideas in with old ideas an weave them together into one large project. For example, we will start out with a Student class. This class is extended throughout the entire year as we learn new ideas such as collections, data files, searching, sorting, etc. The purpose of this technique is to teach how to extend a project by adding components.

Every Monday, I lead a discussion in class called "Tech Talk." The purpose of the time is to bring to light any new technology that the students or I have heard. This often provides me with the chance to discuss the ethical and social consequences of computers. For example, we have had in-depth discussions on the social programming aspects of Facebook, Twitter, information privacy, etc.

Course Outline – AP Computer Science - Overview

Unit	Description	Weeks
0	Gridworld	6*
1	Introduction	1
2	Using Objects	1
3	Implementing Classes	1
4	Fundamental Data Types	1
5	Decisions	2
6	Iteration	2
7	Arrays and ArrayLists	2
8	Designing Classes	2
9	Interfaces and Polymorphism	2
10	Inheritance	2
11	Input/Output	2
12	Object-Oriented Design	2
13	Recursion	2
14	Sorting and Searching	2

Unit 0: Gridworld

Purpose:
*The Gridworld case study, which is tested on the AP exam will be explored and expanded throughout the entire course. It will not be continuously studied for 6 weeks. Instead, I appropriately assign Parts 1, 2, 3, and 4 of the case study student manual as we progress throughout the year.

The Gridworld case study is used to teach students how to read and understand a large program consisting of several classes and interacting objects. It is also an excellent resource for extending classes, such as the Box class or the Critter class.

Apps:
App G.1 Gridworld Part 1
App G.2 Gridworld Part 2
App G.3 Gridworld Part 3
App G.4 Gridworld Part 4

Unit 1: Introduction

Purpose:
The purpose of this unit is to familiarize the student with the hardware and software involved in computer science. In this unit, the student will learn how the computer hardware and software interact with each other to produce a computer program.

Key Terms:

• CPU
• RAM
• Memory
• ALU
• JVM
• High-level language
• Source Code
• Machine code
• Compiler
• IDE
• Console Screen
• Library
• Case sensitive
• Comments
• Classes
• Methods
• Syntax error
• Logic error
• Editor

Apps:
App 1.1 NamePrinter
App 1.2 FacePrinter
App 1.3 DialogViewer

Unit 2: Using Objects

Purpose:

The purpose of this unit is to learn how the computer uses variables, classes and methods to complete tasks. We will learn how to use objects that have been made by other programmers. We will learn how to test simple programs and learn the java vocabulary.

Key Terms:

• Identifiers
• String
• Variables
• Primitive Type
• int
• double
• boolean
• Classes
• Objects
• Methods
• Constructors
• Parameters
• Accessor methods
• Mutator methods
• Return values
• Void
• API documentation
• import
• Objects references
• Frames
• Components

Apps:

App 2.1 AreaRectangle Class

App 2.2 PerimeterRectangle Class
App 2.3 DieSimulator Class
App 2.4 LotteryPrinter Class
App 2.5 ReplaceTester Class
App 2.6 HollePrinter Class
App 2.7 AreaDialogBox Class

Unit 3: Implementing Classes

Purpose:

The purpose of this unit is to become familiar with the process of implementing classes. We will learn how to implement simple methods, understand the purpose and use of constructors, and understand how to access instance fields and local variables. We will also learn to appreciate the importance of documentation comments. The students are taught how to read the Java API.

Key Terms:

• Class methods
• Method definition
• Access specifier
• Return type
• Parameters
• Constructors
• Documentation comments
• javadoc
• Objects
• Instance fields
• Encapsulation
• "Black Box"
• Tester Class
• Implicit Parameter
• This

Apps:

App 3.1 BankAccountTester Class
App 3.1 Adding a method
App 3.3 Adding a constructor
App 3.4 BasketballPlayer Class
App 3.5 Modifying the BasketballPlayer Class
App 3.6 VotingMachine Class

Unit 4: Fundamental Data Types

Purpose:

The purpose of this unit is to learn how to manipulate numbers and character strings. We will learn how to read program input and produce formatted output. We will learn about the numeric and string types such as integer, floating-point and String. We will also learn the role of constants and the final type.

Key Terms:

• Primitive types
• Integer
• Floating-point
• Overflow error
• Rounding errors
• Casting
• Integer division
• Final variables (Constants)
• Magic numbers
• The ”=” operator
• ++ and - -
• % operator
• The Math class and it’s methods abs(), sqrt(), pow()
• Static methods
• Concatenation
• The “+” operator
• Parsing
• Substring
• Scanner class
• Escape sequences
• Formatting numbers
• Dialog Boxes for input and output

Apps:
App 4.1 CashRegister Class
App 4.2 A Counting Cash Register Class
App 4.3 Calculator Class
App 4.4 SodaCan Class
App 4.5 QuadraticEquation Class

Unit 5: Decisions

Purpose:
The purpose of this unit is to understand how to implement decision-making into the programming process. We will learn how to compare integers, floating-point numbers, strings, and objects in order to change the flow of the program. This important concept is what allows a program to change depending on its values.

Key Terms:

• Decisions
• Condition
• If
• If-else
• Body of a statement
• Branching
• Block statement
• Relational operators
• Lexicographic comparison
• Null reference
• Sequences of comparisons
• Switch statement
• Nested branches
• “Dangling” else
• Enumerated types
• Boolean expressions
• Boolean operators
• && (And)
• || (Or)
• ! (Not)
• Boolean variables
• Test Coverage

Apps:
App 5.1 Enhanced Bank Account Class
App 5.2 Paycheck Class
App 5.3 Improving the Quadratic Equation Class
App 5.4 Deck of Cards Class

Unit 6: Iteration

Purpose:
The purpose of this unit is to understand how looping works. We will learn the *while, for, and enhanced for loop* statements. We will also learn how to *nest* loops, process input and avoid infinite loops and off-by-one loops.

Key Terms:

• Iteration
• Loop
• While
• Do
• For
• "Off by One"
• Symmetric loop
• Asymmetric loop
• Spaghetti code
• Counter
• Nested Loop
• Sentinel value
• Random Numbers
• Simulations
• Debugger

Apps:
App 6.1 Currency Converter Class
App 6.2 RandomDataAnalyzer Class

Unit 7: Arrays and Array Lists

Purpose:
The purpose of this unit is to familiarize ourselves with the use of arrays and array lists. These constructs are used to keep track of a list of data. We learn how to fill an array and an arrayList, add data to and delete data from an each. We will also study common array algorithms such as nesting loops in order to access each element of a 2D array.

Key Terms:

• Array
• ArrayList
• Array vs. ArrayList
• Reference
• Array elements
• Index
• Length of an array
• Bounds error
• Initialization
• Enhanced for loop
• Finding a value
• Inserting a value
• Deleting a value
• Finding max or min
• Two-Dimensional arrays
• Copying arrays
• Parallel arrays
• ArrayLists of Objects

Apps:
App 7.1 MyFriends Class
App 7.2 HaveMoreFriends Class
App 7.3 CD Class
App 7.4 Tic-Tac-Toe Class

Unit 8: Designing Classes

Purpose:
The purpose of this unit is to learn how to create and choose appropriate classes to implement. We will carefully analyze the concepts of cohesion and coupling and learn how to minimize the use of side effects. We will take a closer look at the responsibilities of methods and their callers in terms of preconditions and postconditions. We will learn the difference between instance methods and static methods and introduce the concept of static fields while understanding the scope rules for local variables and instance fields.

Key Terms:

• Single Concept
• Cohesion
• Coupling
• Consistency
• Accessor method
• Mutators method
• Immutable class
• Side effects
• Precondition
• Postcondition
• Static method
• Static Field
• Scope
• Local variables
• Packages

Apps:
App 8.1 Payroll Department Class

Unit 9: Interfaces and Polymorphism

Purpose:
As the student reads and writes more complex code that contain multiple classes, it becomes appropriate to talk about interfaces and polymorphism. In this unit, the student will learn how to implement interfaces and to write their own interface. The student will learn how to define objects using the interface and therefore call methods on all objects of the same interface.

Key Terms:

• Interface type
• Class vs. interface
• Defining an interface
• Implementing an interface

Apps:
App 9.1 Geometry FindArea Class
App 9.2 Company Pay Class

Unit 10: Inheritance

Purpose:
Inheritance is a topic that is critical to object-oriented programming. Students will learn that classes can be created that inherit behavior from a more general class. The student will learn how to override superclass methods and invoke superclass constructors. In particular, the student will learn how to override the toString and equals methods.

Key Terms:

• Inheritance
• Extend
• Superclass
• Subclass
• Object class
• Reusing code
• Overriding a method
• Hierarchies
• Private fields of the subclass
• Keyword super
• Converting between subclass and superclass types
• Abstract classes
• Private vs. public

Apps:
Gridworld Apps

Unit 11: Input/Output

Purpose:
Most real world computer programs use files to store metadata created during the course of running the program. For example, a game may store the current score and state of a user if the user chooses to quit the game and come back later. Large amounts of data are stored for programs in the banking industry, education, and medical records. In this unit, the student will learn how to read and write text files. They will learn how to throw exceptions in order to appropriately deal with errors that may occur with opening and closing these files.

Key Terms:

• Text files
• Scanner class
• PrintWriter class
• Opening and closing a file
• Backslash vs. forward slash
• JFileChooser
• Exceptions
• Throwing an exception
• Catching an exception
• The try block
• "Throw early, catch late"

Apps:
Unit 11.1 HighScore Class
Unit 11.2 LunchAccount Class

Unit 12: Object-Oriented Design

Purpose:
Well written code should be the goal for every computer programmer. As programs get larger and more complex, the design of the classes, methods, and interfaces becomes critical. In this unit, the student will learn about the life-cycle of a software project. That is, the student will learn how to transfer their original idea into by analyzing the problem, designing, implementing, testing and then deploying it. This is a critical unit in that it now takes all of the "tools" of programming that they have learned and applies it to a large-scale original application.

Key Terms:

• Software life cycle
• Analysis
• Design
• Implementation
• Testing
• Deployment
• Responsibility of classes
• Relationship between classes
• UML diagram

Apps:
App 12.1 There's an App for That!

Unit 13: Recursion

Purpose:
Recursion is a technique that can be used in certain situations where it can solve a rather complex computational problem by repeatedly calling a simple one. The term "recursion" is used when a sequence is called repeatedly but not in the form of a loop. The goal of this unit is to teach the student how recursion works.

Key Terms:

• Recursion
• Terminating a recursive call

Apps:
App 13.1 Factorial Class
App 13.2 Towers of Hanoi Class

Unit 14: Sorting and Searching

Purpose:

One of the most common tasks in data processing is sorting. For example, a collection of athletes on a track team may be wanted to be sorted out in alphabetical order or sorted by their times. In this unit, we will study several sorting methods, learn how to implement them, and compare their performance.

Key Terms:

• Sorting algorithms
• Swapping
• Selection sort
• Insertion sort
• Merge sort
• Quicksort
• Searching
• Binary Search

Apps:

App 14.1 Selection Sort Algorithm
App 14.2 Modified Selection Sort Algorithm
App 14.3 Insertion Sort Algorithm
App 14.4 Merge Sort Algorithm
App 14.5 In Search Of
App 14.6 Binary Search Algorithm