This is my "AP Computer Science A" syllabus, 2008-2014.
I went through the AP Audit process in 2008-2009.

AP Computer Science A

Mr. Leyzberg

Overview:

This course will develop students' abilities to think logically and solve many-layered problems. Students will learn to program computers and what programs can and cannot do. Students will learn to divide seemingly-impossible problems up into smaller doable problems. Students will tackle social and ethical issues raised by computing such as intellectual property rights and privacy via in-class discussion and essays.

Textbooks:

The Pattern On The Stone: The Simple Ideas That Make Computers Work
W. Daniel Hillis, 1999, Basic Books

Head First Java (2nd Ed.)
Kathy Sierra and Bert Bates, 2005, O'Reilly Media

GridWorld Case Study Manual
https://apstudent.collegeboard.org/apcourse/ap-computer-science-a/about-the-exam/gridworld-case-study

Progression:

Summer Reading

We will read The Pattern On The Stone and follow reading response worksheets to prompt discussion in class and provide background for our study. Reading responses will be graded.

Introduction and Social Issues (2 weeks)

Introductory material: What is code? What is a compiler? What is Java? Followed by discussions of malicious programming and potentials for abuses of technology, as prompted by our summer reading. We will also borrow from current technology news in this first week to bring specific examples into our dialogue about privacy (when we can expect it and how we can respect others' expectations of privacy) and intellectual property (what rights do we have when we generate content and how do we know what rights other content-generators have). Readings and essays will be assigned.

Hardware vs. Software (2 weeks)

Students will learn to distinguish the major components of a computer and their functionality. We will learn about the history of computing, specifically how the current dominant architecture came to be in its current form. We will discuss how data is stored on hardware and which functionality of a computer is dependent entirely on software. We will end with the common thread between hardware and software: logic.

Programming: Getting Started (8-10 weeks)

We will dive into procedural programming. With daily labs, we will begin to create programs and understand how they function. Starting with logic, and variables and an understanding of primitive data structures (arrays and ArrayLists). In this stage, we will use an online Java exercise program called CodingBat, which doesn't need to be installed and from which our program's results will be saved for access from home and school. Students can expect constant but brief homework assignments in this part of the course. In the beginning, we will focus on mathematical applications of programming: we will implement a unit converter (i.e. ounces to gallons, Fahrenheit to Celsius), an information encryption scheme (i.e. transposition and substitution cipher), and a grading program (i.e. calculate the average grade, dropping the two lowest values, grading on a curve, etc.).

What Are Objects & Inheritance? (1.5 weeks)

Having conquered the basics, we are ready to build bigger and more sophisticated

programs. To do that we will need to discuss an organizational scheme called object-oriented programming. In this stage, we will learn about what to expect when we use this scheme. Soon after, all of our programs will implement objects and inheritance.

Programming: Objects (10-12 weeks)

Now we will create big programs, some with hundreds of lines or more. After an initial exploration using a compiler called Greenfoot, we will use the AP GridWorld Case study to implement a few major projects. Here students will refer to the AP Computer Science Quick Reference Guide as we learn about the AP Java subset's methods, classes, and interfaces. As we progress through these labs we will discuss how to design classes to maximize their usefulness and to ensure a safe operating environment for future code. We will begin to collaborate on programming assignments to reinforce the discussions of modularity and good design decisions. At this stage we will work on natural-science applications of programming: we will model the animal kingdom (i.e. a Zebra is a Mammal is a Vertebrae), we will model a few simple machines (which will share Axles and Levers among a number of Machines).

Analyzing Programs (2 weeks)

Having created a few impressive programs, we now turn to a discussion of efficiency. What makes one program more efficient than another? We'll learn about something called "Big-O" notation. Should we sacrifice running-time for code legibility? We will learn about how running-time is measured and what we can do to improve our code.

Programming: Searching, Sorting, And Other Algorithms (2.5 weeks)

Then we can use our knowledge of efficiency to compare algorithms that solve the same problem in better, more efficient ways. We will look at the problems of searching for an element in a list and, also, sorting a list. Several solutions exist for these problems, but they behave differently depending on how the list starts out. We will learn about insertion, selection, and merge sorting as well as sequential and binary search. We will then implement these algorithms, before moving on to our final projects.

Programming: Capstone Projects (remaining time, 4+ weeks)

Before the AP exam we will cement our knowledge by integrating our knowledge in a very large, potentially collaborative, final project. We will design games and simulations using the GridWorld case study. We will make new "critters" in each project: some enemies, some friends. The design decisions will be yours to make, but a defense of the class hierarchy will be required. At this stage, students will use Javadoc to describe what each of your methods does (i.e. pre-conditions, post-conditions, argument descriptions). This is the most creative and least structured of our lab time; students can expect to be in constant communication with their teacher throughout their projects' design and execution.