This syllabus #1249975v1 was reviewed and approved by the College Board in May, 2014.

AP® Computer Science A

Syllabus

Last updated May, 2014

Course Overview

This AP Computer Science A class uses the *TeenCoder: Java Programming* curriculum as the primary resource. It is taught as a one-year (two-semester) sequence and covers all required topics in the "Computer Science A" Course Description published by the College Board at https://apstudent.collegeboard.org/apcourse/ap-computer-science-a. Other introductory programming courses are not required; students merely need to have typical computer usage skills prior to starting this course. All programming concepts are taught from the ground up in a fun, step-by-step manner. The course includes uses a variety of multi-media content such as full-color text, animated / narrated instructional videos, and guided classroom discussions. Strong emphasis is placed on hands-on programming labs that demonstrate mastery of lesson concepts.

Text and Resources

[1] *TeenCoder: Java Programming*, CompuScholar, Inc. 2013, ISBN 9780988707023

http://www.compuscholar.com/teencoder/teencoder_java.php

Teaching Strategies

The course material is designed to appeal to a variety of students, from traditional learners who thrive on written text to audio-visual students who enjoy a multi-media format. Built-in support is included for English Language Learner (ELL) students using the Texas English Language Proficiency Standards (ELPS). All content is delivered through an online system that allows students to work seamlessly both in the classroom and at home.

The course integrates the standard 5E instructional model: *engage* with familiar, real-world examples, *explore* with integrated multi-media lessons, *explain* with guided classroom discussions, *elaborate* with hands-on activities to apply concepts, and *evaluate* with automated lesson quizzes and chapter tests. Each lesson contains a combination of text, video, and guided classroom discussions. Students can seamlessly shift between instructional videos and lesson text to suit their learning styles. Fun, hands-on programming labs allow students to quickly see concrete results.

Labs and Programming Environment

Each of the 25 chapters in *TeenCoder: Java Programming* course contains one or more hands-on programming labs where students will design or implement programs to demonstrate understanding of the lesson topics. Combined, these labs easily exceed the 20 hour minimum lab requirement. Students will get the opportunity to work on individual and group projects and will experience all phases of a project lifecycle, including requirements, design, implementation, and testing. Students are exposed to the three new College Board labs (Magpie, Picture Lab, Elevens) in addition to the textbook labs.

The primary Java programming IDE is "Eclipse" (www.eclipse.org). This free, open-source software is widely used throughout academic and professional environments. It is important for students to become familiar with industry-standard tools and learn to take advantage of modern editing and debugging features. Student classroom computers will be pre-installed with the Eclipse software, and the course contains step-by-step installation instructions for students wishing to work on home computers as well.

Cross-Reference Tables for AP Exam Topics

For a detailed, point-by-point cross reference of every required AP exam topic and Java subset feature to specific lessons in the primary resource [1] *TeenCoder: Java Programming*, please see the CompuScholar "AP Exam Cross-Reference" document:

http://www.compuscholar.com/samples/TeenCoder/AP_Exam_Cross_Reference.pdf

The remainder of this syllabus contains a Course Planner showing the week-by-week progression through the primary textbook and labs.

Course Planner

All readings, unless otherwise noted, are from [1] *TeenCoder: Java Programming*. Each chapter contains multiple lesson quizzes and a chapter test in addition to the listed Lab assignments.

A school year consists of approximately 40 calendar weeks from the last week in August through the last week in May. From that calendar, one week is subtracted for Thanksgiving, two weeks for Christmas, and one week for Spring Break. That leaves approximately 36 calendar weeks or 180 days of school.

The course plan covers 30 school weeks of exam-prep material from late August through mid-April, leaving time prior to the exam for review, practice, and make-up work.

Week	Reading and Objectives	Labs
1	Chapter One: Understanding Computer Programming • A Survey of Computer Hardware • Introduction to Computer Software • Common Programming Languages • Computer Ethics and Security (ethics, copyrights, intellectual property, piracy, software license agreements, firewalls, anti-virus programs, passwords)	**Establish Development Environment** - Install JDK, create working directory, practice submitting projects through the online interface. Class discussion and review of a sample EULA terms and conditions.
2	Chapter Two: Getting Started with Java • The Java Platform • Writing Your First Program • Building and Running from the Command Line • Java Classes and Packages	**Show Time!** – The student's first Java program will print the current time to the console. The student will compile and run the program from the command line (without an IDE).

Week	Reading and Objectives	Labs
3	Chapter Three: The Eclipse IDE • Introducing Eclipse • Eclipse Java IDE Walk-through • Creating an Eclipse Project • Help and Reference Documentation	**Install Eclipse IDE** – If not already installed, the student will add the Eclipse IDE to their home or school computer. **Eclipse Show Time Project** – The student will recreate the same Show Time project using the Eclipse IDE to write, build, and run the program.
4	Chapter Four: Data Types and Variables • Primitive Data Types • Variables • Printing Data	**Experiment with Data Types** – The student will demonstrate declaring, initializing, and printing variables of different data types.
5	Chapter Five: Working With Strings • Reference Data Types • Comparing Strings • Common String Operations • Formatting and Building Strings • Converting Between Strings and Numbers	**String Theory** – The student will create multiple strings and perform a variety of operations on them, including comparison, substrings, formatting, parsing, and case conversion.
6	Chapter Six: User Input • Using Command-Line Parameters • Interactive User Input • Validating User Input	**Conversation Piece** – The student will create a program using a command-line Scanner to obtain a variety of user input, and then format that input into an output story.

Week	Reading and Objectives	Labs
7	Chapter Seven: Basic Flow Control • Logical Expressions and Relational Operators • Using the "if" Statement • The "switch" Statement • For Loops • While Loops	**Fun Factorials** – The student will demonstrate use of a for() loop, while() loop, and do-while() loop to calculate factorials of an input number. Boundary conditions involving maximum integer sizes are explored and tested.
8	Chapter Eight: Writing Methods • Writing and Calling Methods • Method Parameters and Return Values • Calling Methods	**Checkerboard** – The student will write a program that includes a new function to print a checkerboard pattern to the screen given input row and column size parameters.
9	Chapter Nine: Debugging and Exceptions • Logic Errors, Runtime Errors and Exceptions • Catching Exceptions • Finding Runtime Errors • The Eclipse Debugger	**Bug Hunt** – The student is presented with a program that contains a number of bugs. The student will use the Eclipse debugger and troubleshooting skills to identify and resolve each issue.
10	Chapter Ten: Introduction to OOP • Object-Oriented Concepts • Functional Decomposition • Composite Classes • Copying Objects	**Designing a Composite Class** – In this lab the student will design a composite class based around a Computer object. The output is a diagram instead of a program.

Week	Reading and Objectives	Labs
11-12	Chapter Eleven: Objects in Java • Defining a Class • Public, Private, and Protected Members • Constructors • Object Interfaces • Static Members	**Let's Go Racing!** – The student will create a RaceCar object and an IRacer object. Multiple RaceCar instances will be added to a provided RaceTrack object that knows how to run races through the IRacer interface.
13	Chapter Twelve: Graphical Java Programs • Java Swing • Creating a Simple Window • Event-Driven Programming • Layout Managers	**Phone Dialer** – The student's first Java Swing program will show a simple phone keypad and allow users to enter a phone number for display.
14	Chapter Thirteen: Swing Input Controls • Text and Numeric Input • List Input • Option Input	**Pizza Place** – The student will create a pizza ordering screen to demonstrate proper use of many common UI widgets (check boxes, radio buttons, list boxes, etc).
15	Chapter Fourteen: Arrays and Collections • Arrays (1D and 2D) • Java Lists and ArrayLists • Iterators	**Baseball Stats** – The student will use 1D arrays of integers and ArrayLists containing Player objects to insert, track and calculate baseball player batting statistics.

Week	Reading and Objectives	Labs
16-17	Chapter Fifteen: Inheritance and Polymorphism • Learn about the "Jail Break!" game. • Base Classes and Derived Classes • Using References to Base and Derived Classes • Overriding Base Methods • The "Object" Base Class • Using Base Features from Derived Classes	**Game Pieces** – The student will create three derived classes (Deputy, Henchman, Kingpin) from an abstract base, in preparation for using these classes in the mid-term project. The classes are tested to ensure they meet the requirements using a provided test class.
18-19	Chapter Sixteen: MidTerm Project For the mid-term project the student will complete a game called "Jail Break" that is based on an old Viking board game. The student will create the abstract hierarchy of pieces (AbstractGamePiece, Deputy, Henchman, Kingpin) and write other logic to complete the game. The project consists of 6 guided lab steps that involve creating new classes, modifying existing code, and integrating with provided starter objects. Each guided step contains a checkpoint for testing to ensure code meets the requirements at each step. Key concepts demonstrated include: • Encapsulation • Inheritance • Polymorphism • Modeling real-world activities • Integrating new and existing classes	**Building the Activity Starter** – Ensure the student can find and build the starter project. **Completing JailBreak.reset()** – Write logic to initialize the game board with pieces in the starting position. **Selecting Game Pieces** – Write game logic to allow selection and de-selection of game pieces. **Moving Game Pieces** – Write game logic (including virtual method overrides) to control game piece movement. **Capturing Game Pieces** – Write game logic to control game piece capturing. **Ending the Game** – Complete the end-of-game logic.

Week	Reading and Objectives	Labs
20	Chapter Seventeen: Math Functions in Java • Java Math Functions • The Binary Number System • Creating a MathFactory demonstration	**MathFactory Activity** – The student will expand the MathFactory lab to include decimal-to-binary conversion.
21	Chapter Eighteen: File Access • Data Streams • Reading and Writing Text Data • Reading and Writing Binary Data	**Address CSV** – The student will write a program to convert a list of Address structures to a CSV file on disk, and then read that file back in again and re-populate the address list.
22	Chapter Nineteen: Sorting, Searching and Recursion • Recursion • Sorting Algorithms (Bubble, Selection, Insertion, Merge) • Searching Algorithms (Sequential, Binary)	**Recursive Binary Search** – The student will write a binary search function to locate a number in a pre-sorted array.
23	Chapter Twenty: Program Efficiency • Common Algorithms • Algorithm Performance (Big-O) • Measuring Sorting Efficiency	**Comparison of Sorting Algorithms** – The student will implement timing and data-generation algorithms and measure the performance of 4 different sort routines with various numbers of elements.
24	Chapter Twenty-One: Vector and Bitmap Images • Screen Coordinates • Drawing Shapes • Drawing Images	**Sky Art** – The student will use recursion, vector graphics, and image graphics to generate a randomized cloudy sky scene.

Week	Reading and Objectives	Labs
25-26	Chapter Twenty-Two: College Board Supplemental Labs This chapter gives students a chance to complete many activities in the College Board AP supplemental labs, including: • Magpie • Picture Lab • Elevens The older GridWorld case study is available for students who wish to pursue extra exercises on their own.	**Magpie Chatbot** – Guided lab involving string parsing and manipulation. **Picture Lab** - Guided lab using 2D arrays in the context of image processing. **Elevens** - Guided lab through object-oriented design concepts with a simple card game.
27	Chapter Twenty-Three: Computer Networking • Basic Networking • Network Topology • Network Addressing	**Animal Palace** – Students will use online tools to find images and store in a shared directory and class web page.
28	Chapter Twenty-Four: Software Engineering Principles • Design Processes and Teamwork • Java Doc • Testing Your Code	**Creating JavaDoc HTML** – The student will add JavaDoc comments to an earlier lab project and generate HTML output using the javadoc tool.

Week	Reading and Objectives	Labs
29-30	Chapter Twenty-Five: Team Project The final project can be completed before or after the AP exam and the timeline scaled to fit available time. Student-driven labs will cover each phase of the software lifecycle. • Project Requirements • Project Design • Project Implementation • Project Testing	**Team Project Requirements** – Student teams will define their final project requirements. **Project Design** – Student teams will design their final projects. **Team Project Implementation** – Student teams will code their final project. **Team Project Testing** – Student teams will test their final project.
31-32	AP EXAM – PRACTICE TESTS, REVIEW, MAKE-UP WORK	Flexible time used to review and practice for the AP exam.
33	AP EXAM – EARLY MAY	
34+	After the exam, the class will work on completion / extension of Team Project or other fun teacher-driven activities.	Hands-on programming activities determined by class, teacher, and available time.