

Syllabus - Advanced Placement (AP) Computer Science A

[Instructor Name]

[School Name]

Email: [Instructor Email]

Web: [Instructor Web Site]

[school information removed]

Overview and Objectives

The basic skills covered in this course give students a strong background in the fundamental concepts of structured and object-oriented programming. Students are engaged in formal, in-depth study of topics including abstraction, design, data structures, and algorithms. The material covered is equivalent to that of a first-semester introductory computer science course found in most university programs.

In the second semester following the completion of the primary text and case study, students will participate in a team-based software development project with the support and guidance of a prominent [county] web development company. Students will put the skills that they have learned in class to use in a production environment, dealing with real-world issues and seeing their work “go live” as part of a finished product. In addition to earning valuable working experience, students also go forth with an enhanced resume and networking opportunities.

Finally, this course prepares students for the Advanced Placement (AP) Computer Science A exam, offered at the end of the second semester of each academic year.

Personal Philosophy

Computer Science encompasses not only programming but calls upon students to think about the “big picture” in solving problems using software. I hope to impart on my students the ability to make use of their computing skills to address challenges in many diverse disciplines from the physical sciences to music, as well as discover new avenues of communication and teamwork.

My classes are not structured in a lecture format in the traditional academic sense. Students are engaged in discussion at every step, and are asked to describe the concepts learned in their own terms. I actively encourage students to come up with creative solutions to programming problems that may not be present in the text, and am interested in discovering answers for the challenges put forth by the many ways that different people learn.

Prerequisites

This class is an Advanced Placement (AP) level course offering that is currently the most advanced of its type offered at [School Name]. Students that wish to take this course will need to have earned a grade of “B” or higher in the introductory class offered by the Computer Science department, or demonstrate equivalent proficiency with the required computing concepts.

Students are assumed to be familiar with the basic information storage and retrieval aspects of modern operating systems (mainly Microsoft Windows and Apple Mac OS X), and are comfortable using email, web-based resources, and downloading/installing free open-source software.

Use of the school computer labs is dependent on the acceptance of the terms laid out in the [School Name] Computer and Network Acceptable Use Policy.

Course Materials

The primary textbook for this course is:

Brown, Beth. *A Guide To Programming In JAVA, Second Edition*. New Jersey: Lawrenceville Press, 2007.

Supplementary reference materials include:

Liang, Daniel Y. *Introduction to Java Programming, Comprehensive Version, 7th Edition*. New Jersey: Prentice Hall, 2008.

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

Textbook Data Files: www.lpdatafiles.com – A collection of Slides, Code Examples, and Self-Tests to reinforce comprehension of materials in the main textbook.

JCreator Integrated Development Environment: www.jcreator.com

BlueJ Integrated Development Environment: www.bluej.org

Course Outline

Unit	Title/Topics/Objectives	Resources/Assessments/Strategies
1	<p>Review of Computing</p> <p>Topics:</p> <ul style="list-style-type: none"> • Hardware • Software • Networks • Ethical and Social Implications of Computer Use <p>Objectives:</p> <ul style="list-style-type: none"> • Review and understand key concepts and terminology of computing. • Review and understand school policies regarding proper use of campus computers and networks. • Read and discuss current-day issues in computing and technology, and 	<p>Resources:</p> <p>Textbook – Chapters 1 and 2</p> <p>On-line reading (see below)</p> <p>Assessments:</p> <p>Quiz 1 – Terms and definitions of common hardware, software, and network systems in use today.</p> <p>Quiz 2 – Opinion writing on a major ethical and/or social issue in technology.</p> <p>Strategies:</p> <p>Active in-class discussions using language students can already understand without excessive technical jargon.</p>

	<p>evaluate the responsibilities of people working in computing to themselves, their peers, and the network-connected world around them.</p> <ul style="list-style-type: none"> • 	<p>Students are encouraged to check the course web page for links to a variety of current articles on computing and technology.</p>
2	<p>Java, IDE's, and Web Programming</p> <p>Topics:</p> <ul style="list-style-type: none"> • World Wide Web • HTML and CSS • JavaScript and Applets • Integrated Development Environments <p>Objectives:</p> <ul style="list-style-type: none"> • Understand the development and overall structure of the Java language as it is used in web development. • Overview of the JCreator and BlueJ IDEs. 	<p>Resources: Textbook – Chapters 3 and 4 Free Software Download (IDE)</p> <p>Assessments: Quiz 2 – Terminology and use of web-based tools. Lab 1 – Integrate a simple applet into a web page template.</p> <p>Strategies: Students who are not yet proficient with an individual concept after the in-text review assignments should complete the corresponding end-of-chapter exercises.</p> <p>Technology terms are abundant and can be confusing; this class only defines and uses those terms that are directly related to the problems being addressed.</p>
3	<p>Variables and Constants</p> <p>Topics:</p> <ul style="list-style-type: none"> • Abstract data types • Numeric expressions and operators • Debugging • Strings • User Input • Code Conventions <p>Objectives:</p> <ul style="list-style-type: none"> • Understand the declaration and instantiation of different abstract data types. • Accept and handle user input through the <i>Scanner</i> class. • Manipulate data through the use of numeric expressions, type casting, and operators. • Become familiar with strategies for 	<p>Resources: Textbook – Chapter 4 Textbook – Chapter 6 (Strings)</p> <p>Assessments: Quiz 3 – Terminology and use of data types, expressions, operators, and built-in language tools. Lab 2 – A variety of small programs to take in and manipulate user input. Lab 3 – A variety of small programs to calculate math and physics problems requiring user input.</p> <p>Strategies: Students are assigned lots of small problems that deal with all of the different types, and display them in many different ways.</p>

	<p>debugging syntax, logic, and run-time errors.</p> <ul style="list-style-type: none"> Establish a consistent set of coding conventions for code readability, maintainability, and re-use. 	
4	<p>Conditional Control Structures</p> <p>Topics:</p> <ul style="list-style-type: none"> <i>if</i> <i>if-else</i> <i>switch</i> java.lang.Math Compound boolean expressions <p>Objectives:</p> <ul style="list-style-type: none"> Understand the structure and use of the different conditional control structures in problem-solving. Explore the functionality of the Math class for random number generation and formatting of numeric output. Combine boolean expressions with conditionals in order to simplify complex branching problems. 	<p>Resources: Textbook – Chapter 5</p> <p>Assessments: Quiz 4 – Terminology and use of conditional statements. Lab 4 – Rock/Paper/Scissors Game written in a structured programming style requiring the use of different conditional statements, compound boolean expressions, and a random number generator.</p> <p>Strategies: Students have extensive practice in-class writing nested conditional statements to solve logic problems, and come up with original problems that others have to solve.</p> <p>Emphasis on solving problems rather than inputting code.</p>
5	<p>Loops</p> <p>Topics:</p> <ul style="list-style-type: none"> <i>while</i> <i>do-while</i> <i>for</i> <p>Objectives:</p> <ul style="list-style-type: none"> Understand the structure and basic use of loop structures in problem-solving (to be expanded on in a later unit covering Data Structures). Use of accumulator and counter variables. Decision-making while coding: what is the best kind of loop to solve specific types of problems? 	<p>Resources: Textbook – Chapter 6</p> <p>Assessments: Quiz 5 – Terminology and use of loop structures. Lab 5 – Complex word guessing game written in a structured manner requiring a flowchart, pseudocode, and implementation of the String class and loop structures. Midterm 1 – A comprehensive test of all concepts covered through unit 5.</p> <p>Strategies: Students diagram and data-trace the progress of loops to ensure that there is a conceptual understanding of how each</p>

		different structure functions.
6	<p>Methods</p> <p>Topics:</p> <ul style="list-style-type: none"> • Methods • Parameters • Overloading • Documentation <p>Objectives:</p> <ul style="list-style-type: none"> • Understand the fundamental importance of writing methods as the primary problem-solving feature of an object-oriented programming language. • Implement efficient functions to work with data while passing parameters, and overloading similar functions. • Establish a set of core code commenting guidelines for complex functions. 	<p>Resources: Textbook – Chapter 7</p> <p>Assessments: Quiz 6 – Terminology and use of methods, parameters, method overloading, and code-commenting. Lab 6 – Write several small programs that use methods, pass parameters, and overload methods, to solve a variety of mathematical problems. Include code comments that document what each significant method accomplishes.</p> <p>Strategies: Numerous problems are assigned that ask students to create a series of methods to solve. The first group assignments show up here, introducing the need for Object-Oriented concepts presented in the following chapter.</p>
7	<p>Classes and Object-Oriented Design</p> <p>Topics:</p> <ul style="list-style-type: none"> • What is an object? • What is a class? • Encapsulation/Information Hiding • Constructors • Instance versus Class Members • Defining “Object-Oriented” • Inheritance • Polymorphism • Interfaces (Comparable) <p>Objectives:</p> <ul style="list-style-type: none"> • Understand the concepts that allow the building of object-oriented programs. • Evaluate the advantages/disadvantages of implementing solutions in an object-oriented manner. • Design and write classes. • Organize a software development team 	<p>Resources: Textbook – Chapters 8 and 9</p> <p>Assessments: Quiz 7 – Terminology and use of object-oriented design principles. Lab 7 - Re-implement the structured version of the Rock/Paper/Scissors game coded in unit 5 in an object-oriented fashion using methods, passing parameters, and overloading methods where appropriate. Lab 8 - Team Coding Project: In teams dividing labor up by designers, coders, and testers, students will implement, test, and deliver an object-oriented software system that allows a person to play the game “Blackjack” against a computer dealer. This lab involves both planning (algorithms, pseudocode) as well as coding.</p> <p>Semester 1 Final Exam</p>

	<p>(leader/designer, coders, testers) to code a software system to play a variety of simple games.</p>	<p>Strategies: Discussion-oriented class meetings focused on having students describe the relationships of objects and classes in their own terms.</p> <p>Team coding projects explore not only Object-Oriented approach to coding, but also the social dynamic of group programming.</p>
8	<p>Data Structures</p> <p>Topics:</p> <ul style="list-style-type: none"> • Array • ArrayList • Wrapper Classes • Stack • Queue • Linked-List <p>Objective:</p> <ul style="list-style-type: none"> • Implement and use four different data structures and their related utility code in solving programming problems. • Practice the use of the ArrayList class and Wrapper classes in working with object data. 	<p>Resources: Textbook – Chapters 10 and 14 <i>Liang</i> – Part 1</p> <p>Assessments: Quiz 8 – Terminology and use of different data structures. Lab 9 – Solve in-text exercises requiring the use of single, and two-dimensional arrays, objects using ArrayList, and wrapper classes to access the information in those objects. Lab 10 – Create and present test cases for three data structures: stack, queue, and linked-list.</p> <p>Strategies: Students diagram and data-trace the changes that occur while using each data structure to ensure that there is a conceptual understanding of how each one functions.</p> <p>There is an emphasis on being able to explain how a structure works rather than just entering code into the IDE.</p>
9	<p>Algorithms and Recursion</p> <p>Topics:</p> <ul style="list-style-type: none"> • Selection Sort • Insertion Sort • Mergesort • Binary Search 	<p>Resources: Textbook – Chapter 13 <i>Liang</i> – Part 5</p> <p>Assessments: Quiz 9 – Solve world problems involving each of the different sorting algorithms. Lab 11 - Using object-oriented methods</p>

	<ul style="list-style-type: none"> • Running Time <p>Objectives:</p> <ul style="list-style-type: none"> • Understand the conceptual framework of sorting algorithms as implemented in an object-oriented language. • Use learned concepts of data , conditionals, and loop structures to create and improve the performance of different sorting algorithms. • Evaluate the efficiency of recursive functions, expressed logarithmically. • 	<p>and data structures from unit 8, implement and test ALL of the recursive algorithms covered in this section using a variety of provided data sets.</p> <p>Strategies: Prior to coding, each student presents the design and a data-flow diagram of each structure.</p>
10	<p>Graphical User Interfaces (GUI)</p> <p>Topics:</p> <ul style="list-style-type: none"> • Swing Package • Events • Layout • GUI Elements <p>Objectives:</p> <ul style="list-style-type: none"> • Make use of existing classes in the current Java development platform to create graphical user interfaces that can: <ul style="list-style-type: none"> ◦ accept and manipulate user input. ◦ display text and graphics. ◦ show visual representations of objects in a software system. ◦ interact with existing web-based systems (ex. web browsers). ◦ 	<p>Resources: Textbook – Chapters 11 and 12 <i>Liang</i> – Part 3</p> <p>Assessments: Quiz 10 – Terminology and use of Java GUI classes and elements. Lab 12 – Create a graphical TicTacToe game that allows two human players to play versus one another. Place the completed applet on a website so that it is web-accessible. Midterm 2 – Tests knowledge and application of all concepts to this point.</p> <p>Strategies: In-text review assignments</p>
11	<p>GridWorld Case Study</p> <p>Topics:</p> <ul style="list-style-type: none"> • Current AP Computer Science case study for 2007-present. • AP Test Component (Parts 1-4 for A level exam) <p>Objective:</p> <ul style="list-style-type: none"> • Experiment with different object types and observe how programming changes affect the behavior of those 	<p>Resources: <i>AP Gridworld</i> Parts 1-4</p> <p>Assessments: Lab 13 - Answer all questions in the case-study text. Code all exercises in the case-study text. Submit completed case-study written questions and code. Final Exam – In lieu of a final examination, students participate in a long-month review for the AP Computer Science A Exam. Students who choose to not take the AP Exam are given a similar</p>

	<p>objects.</p> <ul style="list-style-type: none"> • Prepare for case study related questions on the AP Computer Science A Exam. 	<p>assessment for a final grade.</p> <p>Strategies: Students will need to refer back to their text, study guides, and other materials that they have created should any issues arise in working with the code and data presented in the case study.</p>
12	<p>Class Software Development Project</p> <p>Topics:</p> <ul style="list-style-type: none"> • Related programming languages • Team software development • Requirements • Design • Prototype (Beta) • Delivery (1.0) <p>Objective:</p> <ul style="list-style-type: none"> • Participate in a small-scale “software engineering” project from beginning to end including requirements, design, prototype, and delivery. • Participate in interviews and Q/A sessions with Chief Technology Officer. • 	<p>Assessments: Requirements Document Design Document Prototype 1.0 Build</p> <p>Strategies: Students will observe and interview professional software developers in the field. They will receive feedback on their ongoing project from these professionals as they attempt to complete it on a set timeline.</p>

Grading

This class incorporates a significant amount of written and coded work, most of which is completed in class. Regardless of the number of points available in total, each quiz, lab, and test is graded using the following rubric:

CONCEPT	DEFINITION	PERCENTAGE
Correctness	Programs should conform to the requirements as stated in the problem. They need to demonstrate the correct handling of special cases and error conditions. Testing data should include demonstrations of each of these exceptions to regular input.	40.00%
Design	Each class should consist of small, coherent, and independent methods. Global variables should be used on a very limited scale where needed.	40.00%

Style/Documentation	Programs should be easy to read and understand. Comments should be brief yet complete.	10.00%
Efficiency	Algorithms and program design should be selected with regard to efficiency of time and space. "Elegant" solutions are always preferred to "brute force" ones.	10.00%

Students earn a non-curved letter grade based on the total number of points earned on a straight percentile scale (90-100 = A, 80-89 = B, 70-79 = C, etc). The larger and more complex a quiz, lab, or test is, the more points it is worth. Late assignments are accepted for half-credit (or more) at the discretion of the instructor.

There are numerous opportunities for students to earn bonus points throughout the year in presenting creative solutions to problems and demonstrating effective leadership in team coding projects.

Overall, the grading in this course will reflect both a student's ability to turn assignments in on time as well as demonstrating proficiency with the material.

[additional school-related policy information removed]