

# AP<sup>®</sup> Computer Science A Course Syllabus

## Course Overview

The purpose of this course is to give students an understanding of computer science in general and a practical knowledge of object-oriented programming using Java in particular. Upon successful completion of this course, students will be able to recognize and assess the role that computers play in society. They will be able to identify the major parts of a computer system in terms of hardware and software components. Students will be able to solve a variety of problems in different areas by designing and implementing computer-based solutions. They will be proficient in the evaluation and use of algorithms and data structures in those solutions. Students will be able to create, document, and debug increasingly sophisticated code in an object-oriented environment using the Java programming language, in particular by using standard Java library classes from the AP<sup>®</sup> Java subset. Finally, they will be able to critically discuss significant programs, including the AP<sup>®</sup> *Computer Science Case Study*.

My goal for the AP<sup>®</sup> Computer Science course is to give my students the skills and confidence necessary to solve a wide variety of problems and to communicate effectively about computer science issues. Our students have a great enthusiasm for and interest in technology and by offering this course I hope to give them the tools needed for future studies and possibly careers. This course will complement our AP<sup>®</sup> course offerings in mathematics and science and give students another avenue to pursue the rigorous study of a challenging and hopefully rewarding field.

I also place special emphasis on two topics that I feel are important in an all-girls school environment. First, I require students to research and present information about a woman who has contributed in a significant way to the development of computer science and/or technology. Second, I require students to choose a real-world employment opportunity in the computer science or technology fields and to research the preparation necessary to be qualified for such a position, especially the educational background required.

## Computer Facilities

The course is taught in a combined classroom/computer lab with Apple iMacs which features an interactive Epson projector connected to one of the computers for instruction and demonstrations. All computers have the *BlueJ* IDE installed. In addition to classroom time in our “Mac lab”, we make our computer facilities available to our students as much as possible in order to meet the needs of our students, particularly the small number who do not have computers and/or internet access at home. The Mac lab is available for student use outside of class before school, at lunch, during study hall and activity periods, and after school at various times/days throughout the week, typically 3-4 hours per week. Students can also work on assignments in two other computer labs on campus which primarily have PCs with some additional iMacs every day between 7:00 am and 6:00 pm. Finally, all students are given instructions on how to download and install the *BlueJ* IDE on their home computer so that they can work on assignments at home if they wish to do so.

## Required Texts

College Board. *AP<sup>®</sup> GridWorld Case Study*. New York: College Entrance Examination Board, 2006.

Litvin, Maria and Litvin, Gary. *Java Methods A & AB: Object-Oriented Programming and Data Structures*. Andover, Massachusetts: Skylight Publishing, 2006.

## Other Resources

College Board. *AP<sup>®</sup> Computer Science A Course Description*. New York: College Entrance Examination Board, 2010.

College Board. *2004 AP<sup>®</sup> Computer Science A Released Exam*. New York: College Entrance Examination Board, 2004.

Litvin, Maria and Litvin, Gary. *Be Prepared for the AP<sup>®</sup> Computer Science Exam in Java, Fourth Edition*. Andover, Massachusetts: Skylight Publishing, 2009.

Litvin, Maria and Litvin, Gary. *250 Multiple-Choice Computer Science Questions in Java*. Andover, Massachusetts: Skylight Publishing, 2008.

Web resources accompanying *Java Methods A & AB: Object-Oriented Programming and Data Structures*: [www.skylit.com/javamethods](http://www.skylit.com/javamethods)

Web resources for the *BlueJ* IDE: [www.bluej.org](http://www.bluej.org)

Online textbook – *Introduction to Programming Using Java, Sixth Edition* by David J. Eck:  
<http://math.hws.edu/javanotes/>

## Course Outline

**Note:** Each week has one to three 90-minute class periods depending on the school calendar.

Resources include references to *Java Methods A & AB: Object-Oriented Programming and Data Structures (JM)*; *Be Prepared for the AP<sup>®</sup> Computer Science Exam in Java (BP)*; and the *AP<sup>®</sup> GridWorld Case Study (GW)*.

### Fall Semester

Unit (Weeks)	Title, Topics and Student Objectives	Resources, Assessments and Strategies
1 (1)	<p><b>Introduction to Computer Science and Java</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Computer systems</li> <li>• Java basics</li> <li>• Using the compiler</li> <li>• Input and output</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Understand basic computer vocabulary</li> <li>• Understand the components of a computer system – hardware vs. software</li> <li>• Understand computer ethics – privacy and legal issues, intellectual property rights, impact on society</li> <li>• Understand basic programming and Java vocabulary</li> <li>• Edit, compile and run simple programs in Java</li> <li>• Understand different types of errors – compile-time, run-time, and logic</li> <li>• Understand different methods of input</li> <li>• Use <code>System.out</code> for output using <code>print</code> and <code>println</code></li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch.1: 1.1-1.7; ch.2: 2.1-2.7</li> <li>• BP ch.1: 1.1-1.5; ch.3: 3.5-3.6; ch.4: 4.1</li> <li>• College Board AP<sup>®</sup> <i>Computer Science A Course Description</i></li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 1.1-1.6, 2.1-2.7: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Lab - “Hello, World!”: traditional first assignment to introduce students to Java and the <i>BlueJ</i> IDE</li> <li>• Lab - Greetings: additional practice with input and output</li> </ul> <p><b>Strategies:</b></p> <ul style="list-style-type: none"> <li>• Role-playing activity with written commands – introduces concept of computer programming and need for unambiguous instructions</li> <li>• Guest lectures from school’s Network Administrator on network security and from Director of Academic Technology on Acceptable Use Policy and related issues</li> <li>• Students practice various I/O methods with short, diverse program assignments</li> <li>• Students look at the inside of an actual desktop computer and ID major parts</li> </ul>
2 (2-3)	<p><b>Defining Variables, Arithmetic Expressions</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Understand variables and their use</li> <li>• Using comments in programs</li> <li>• Arithmetic expressions in Java</li> <li>• Representing numbers in different bases</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Understand vocabulary related to variables</li> <li>• Declare and initialize variables and constants in Java</li> <li>• Understand mathematical expressions and their precedence in Java</li> <li>• Be able to change bases of numbers – binary, octal,</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch. 5: 5.1-5.7; ch.6: 6.1-6.11</li> <li>• BP ch.2: 2.1</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 5.1-5.7, 6.1-6.11: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Lab: Correcting Syntax Errors – reading and correcting code</li> <li>• Lab: Pie Chart – insert appropriate variable declarations and arithmetic expressions</li> <li>• Lab: Quadratic Formula – use variables and arithmetic expressions to implement formula</li> </ul>

	<p>decimal, and hexadecimal</p> <ul style="list-style-type: none"> <li>• Use casting to make data more accurate</li> <li>• Understand the limits of finite number representation</li> <li>• Use the assignment operator correctly</li> </ul>	<p><b>Strategies:</b></p> <ul style="list-style-type: none"> <li>• Draw comparisons with topics from Algebra II course</li> <li>• Identify and discuss similar concepts using the students' graphing calculators (used in all math and science classes)</li> </ul>
3 (4-5)	<p><b>Introduction to Classes and OOP</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Creating and using classes</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Understand vocabulary related to classes</li> <li>• Understand the difference between public and private access in a class</li> <li>• Use and understand the <code>DecimalFormat</code> class and the <code>Random</code> class</li> <li>• Write classes from scratch using appropriate data representation</li> <li>• Understand how to declare a method and parameters in that method</li> <li>• Understand the use of preconditions, postconditions and assertions in methods</li> <li>• Understand the difference between OOP and top-down development</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch.3: 3.1-3.7</li> <li>• BP ch.3: 3.1-3.3</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 3.1-3.7: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Lab: First Steps – fill in class definitions</li> <li>• Lab: RPG ability scores – use <code>Random</code> class</li> </ul> <p><b>Strategies:</b></p> <ul style="list-style-type: none"> <li>• “Class of the class” competition – students compete in small groups to correctly represent a class given a description</li> </ul>
4 (6-10)	<p><b>Conditionals and Looping</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• <code>if</code>, <code>if-else</code>, <code>while</code>, <code>for</code></li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Understand vocabulary related to conditionals and looping</li> <li>• Construct conditional statements and loops</li> <li>• Understand and be able to identify and correct various errors associated with loops</li> <li>• Use logical operators effectively</li> <li>• Construct truth tables</li> <li>• Be able to calculate statement execution counts</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch.7: 7.1-7.13; ch. 8: 8.1-8.7</li> <li>• BP ch.2: 2.2</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 7.1-7.13, 8.1-8.7: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• Lab: Rolling Dice – program design, extending classes</li> <li>• Lab: Perfect Numbers – iterative problem solving</li> </ul> <p><b>Strategies:</b></p> <ul style="list-style-type: none"> <li>• Revisit role-playing with written instructions to demonstrate basic looping concepts</li> <li>• Use flowcharts to analyze flow of programs</li> </ul>
5 (11-12)	<p><b>The String Class</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• <code>String</code> class</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Instantiate <code>String</code> objects</li> <li>• Understand that strings are immutable</li> <li>• Use appropriate <code>String</code> methods to solve problems</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch.10: 10.1-10.10</li> <li>• BP ch.2: 2.3</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 10.1-10.10: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• Lab: Lipograms – create class to handle strings</li> </ul>
6 (13-16)	<p><b>Arrays and ArrayList</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Using <code>ArrayList</code> class</li> <li>• Declaring and initializing arrays</li> <li>• Manipulating array with loops</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch.12: 12.1-12.12</li> <li>• BP ch.2: 2.5, 2.6</li> </ul> <p><b>Assessments:</b></p>

	<ul style="list-style-type: none"> <li>• Creating parallel arrays</li> </ul> <b>Objectives:</b> <ul style="list-style-type: none"> <li>• Use the <code>ArrayList</code> methods</li> <li>• Understand vocabulary related to arrays</li> <li>• Declare one-dimensional arrays in Java</li> <li>• Use initialize lists when declaring arrays</li> <li>• Manipulate arrays using loops and array indices</li> <li>• Use the physical and logical sizes of an array together to ensure that they do not exceed array bounds by identifying the boundary cases and using text data to verify results</li> <li>• Understand the appropriate use of parallel arrays for the processing of certain types of data</li> <li>• Understand the difference between primitive data types and objects and work with arrays of each type</li> <li>• Understand when to use an array or an <code>ArrayList</code> to represent data</li> </ul>	<ul style="list-style-type: none"> <li>• JM exercises 12.1-12.12: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• Lab: Fortune Teller – insert appropriate arrays in program</li> <li>• Lab: Document Index – read text file and generate index</li> <li>• Lab: Chomp – implement multiple classes and arrays</li> </ul> <b>Strategies:</b> <ul style="list-style-type: none"> <li>• Students are given the opportunity to find artistic methods of representing arrays and their composition after different forms of manipulation</li> <li>• Students use the Chomp program as a model for their own projects to be completed in the spring semester</li> </ul>
7 (17-19)	<b>Searching and Sorting Arrays</b> <b>Topics:</b> <ul style="list-style-type: none"> <li>• Bubble, Selection and Insertion sorts</li> <li>• Sequential and Binary searches</li> </ul> <b>Objectives:</b> <ul style="list-style-type: none"> <li>• Write a method for searching an array</li> <li>• Perform specified insertions and deletions in arrays</li> <li>• Trace through and understand the time constraints of searching and sorting algorithms</li> <li>• Understand the algorithmic basis of each type of search and sort technique</li> <li>• Understand when to use each search and sort technique and the time efficiencies of each</li> <li>• Identify reusable components from existing code using classes and class libraries</li> <li>• Understand the criteria for selecting the most appropriate sort or search</li> </ul>	<b>Resources:</b> <ul style="list-style-type: none"> <li>• JM ch.13: 13.1-13.6, 13.11</li> <li>• BP ch.5: 5.1-5.3</li> </ul> <b>Assessments:</b> <ul style="list-style-type: none"> <li>• JM exercises 13.1-13.6: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• Lab: Keeping Things in Order – extend class <code>ArrayList&lt;String&gt;</code> in order to sort list of words</li> </ul> <b>Strategies:</b> <ul style="list-style-type: none"> <li>• Role-playing redux – students stand in a line and act out searching and sorting methods</li> </ul>
(20)	<b>Fall Final Exam</b>	<b>Fall Final Exam</b> <ul style="list-style-type: none"> <li>• 90-minute cumulative exam with multiple choice and free response questions</li> </ul>

## Spring Semester

Unit (Weeks)	Title, Topics and Student Objectives	Resources, Assessments and Strategies
8 (21-23)	<b>GridWorld (Parts 1-3)</b> <b>Topics:</b> <ul style="list-style-type: none"> <li>• Experimenting with a large program</li> <li>• Using classes</li> <li>• Modifying classes</li> </ul> <b>Objectives:</b> <ul style="list-style-type: none"> <li>• Run the case study and analyze the output</li> <li>• Experiment with the GridWorld environment</li> <li>• Understand the <code>Bug</code> class, the <code>Runner</code> class, and the <code>Grid</code> interface</li> </ul>	<b>Resources:</b> <ul style="list-style-type: none"> <li>• GW Parts 1-3</li> <li>• BP ch.6: 6.1-6.4</li> </ul> <b>Assessments:</b> <ul style="list-style-type: none"> <li>• GW student exercises</li> </ul> <b>Strategies:</b> <ul style="list-style-type: none"> <li>• Students are initially given time to play with the program with minimal introduction. This encourages</li> </ul>

	<ul style="list-style-type: none"> <li>• Extend the <code>Bug</code> class to meet new specified requirements thereby creating a new type of bug</li> <li>• Review the case study to understand the development process of a large program</li> </ul>	<p>them to analyze the program interactively and gives them reference points for our subsequent analysis of the underlying code.</p>
9 (24-27)	<p><b>More on Classes, Inheritance, and Interfaces</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Classes</li> <li>• Inheritance</li> <li>• Abstract classes</li> <li>• Interfaces</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Demonstrate inheritance by extending a class</li> <li>• Understand polymorphism and know when it is appropriate to override methods in a super class</li> <li>• Create and extend an abstract class</li> <li>• Create and extend a class given class specifications and description of relationship among the classes</li> <li>• Implement an interface</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch. 9: 9.1-9.13; ch. 11: 11.1-11.9</li> <li>• BP ch.4: 4.4-4.6</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 9.1-9.13, 11.1-11.9: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• Lab: Snack Bar – practice with classes and inheritance</li> </ul>
10 (28-30)	<p><b>GridWorld (Part 4)</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Inheritance</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Use inheritance to extend the <code>Critter</code> Class by making new types of <code>Critters</code></li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• GW Part 4</li> <li>• BP ch.6: 6.5, 6.6</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• GW exercises</li> </ul>
11 (31-33)	<p><b>Recursion and Merge Sort</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• Recursion</li> <li>• Merge Sort</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Use a recursive method to solve a problem</li> <li>• Understand the difference between recursive and iterative solutions to a problem</li> <li>• Understand and use the Merge Sort</li> <li>• Understand how to calculate the informal runtime of Merge Sort and compare with other sorts</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• JM ch.13: 13.7-13.11; ch. 22: 22.1-22.7</li> <li>• BP ch.5: 5.4, 5.5</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• JM exercises 13.7-13.10, 22.1-22.7: assigned as appropriate in each class session; checked and reviewed at following class session</li> <li>• Unit test - timed AP-style multiple choice and free response questions</li> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• Lab: Benchmarks – compare running times of sort methods</li> <li>• Lab: Tower of Hanoi – use recursion to solve puzzle</li> <li>• Lab: Game of Hex – simulate game using recursion</li> </ul>
(34-37)	<p><b>AP<sup>®</sup> Exam and Final Project</b></p> <p><b>Topics:</b></p> <ul style="list-style-type: none"> <li>• The AP<sup>®</sup> Exam</li> <li>• Student projects</li> </ul> <p><b>Objectives:</b></p> <ul style="list-style-type: none"> <li>• Understand the AP<sup>®</sup> Exam format and testing procedures</li> <li>• Finish review and preparation for AP<sup>®</sup> Exam</li> <li>• Propose, develop, code and document original programming projects</li> <li>• Effectively communicate project results</li> </ul>	<p><b>Resources:</b></p> <ul style="list-style-type: none"> <li>• College Board AP<sup>®</sup> <i>Computer Science A Course Description</i></li> <li>• College Board 2004 AP<sup>®</sup> <i>Computer Science A Released Exam</i></li> <li>• BP practice exams</li> </ul> <p><b>Assessments:</b></p> <ul style="list-style-type: none"> <li>• Previous AP<sup>®</sup> Exam questions</li> <li>• After AP<sup>®</sup> exam students complete major individual programming project and present results to class</li> </ul> <p><b>Strategies:</b></p> <ul style="list-style-type: none"> <li>• Students are given a complete simulated AP exam experience on a Saturday at school prior to the actual exam</li> <li>• Students do a peer review of some of the free response questions</li> </ul>

## **General Teaching Strategies**

The students in my AP<sup>®</sup> Computer Science course have a range of abilities and experience with computers. The course is designed to be as inclusive as possible. To this end, I carry out a number of activities to make sure that students have a good understanding of the nature, scope and demands of the AP<sup>®</sup> Computer Science course while at the same time building enthusiasm for the subject matter. The students are given a short assignment to carry out over the summer prior to the course. On the first full day of class, the students work in pairs or small groups and take turns carrying out written instructions that they create for each other. Throughout the course students are encouraged to work together on their assignments and periodically on activities that give them insight into the nature of computer science and computer programming. Students are exposed to AP<sup>®</sup> exam type questions early on and consistently throughout the course. Although the bulk of our examination of the case study is reserved for second semester, elements of the program are introduced at appropriate points throughout the course. Finally, the course makes use of a Google website created and moderated by the instructor to facilitate interaction and communication among the students and between the students and instructor in addition to providing a central location for posting class materials and relevant links.