## Course Overview

AP® Computer Science A emphasizes a study in object-oriented programming methodologies with a focus on problem solving and algorithm development. Data structures, design, and abstraction are also covered, but not in the depth that would be appropriate for an AB course. Also, an examination of a large case study program is undertaken, and students are expected to be able to understand and modify code that they have not written. Hands-on laboratory work is used to solidify each concept, and end-of-unit labs and tests are used to assess the progress of each student.

Once the AP® Computer Science A Exam has been completed, students have the opportunity to apply their knowledge in a new setting, using LEGO® MindStorms™ robots that they design, build, and program. Using the LEJOS operating system and the TextPad IDE, students are given the opportunity to use Java to program robots that can interact with their settings.

## Computer Facilities/Lab Component

A dedicated lab of 21 computers is directly attached to our classroom, which provides the perfect environment for teaching the AP® Computer Science A curriculum. Once a concept has been discussed in the classroom setting, only seconds pass before we are experimenting with and implementing the concepts that were covered. The instructor has administrative rights to install software, as well as to access the shared drives of the students. Students have access to the lab before school, during their lunch, and after school.

## Course Resources

Bergin, Joseph, et al. *Karel J. Robot: A Gentle Introduction to the Art of Object-Oriented Programming Using Java*. Copyright Joseph Bergin.
**http://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html**

Bloss, Adrienne and N. Jane Ingram. *Lab Manual to Accompany Java Software Solutions*. New York, New York: Pearson Education, Inc, 2003.

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

Lewis, John, William Loftus, and Cara Cocking. *Java Software Solutions for AP Computer Science A, 2nd Edition*. New York, New York: Pearson Education, Inc, 2007.

Lewis, John, William Loftus, and Cara Cocking. *Instructor's Resource Manual to Accompany Java Software Solutions*. New York, New York: Pearson Education, Inc, 2004.

**Course Outline**

| Unit | Title, Topics, and Student Objectives | Resources, Assessments, and Strategies |
|---|---|---|
| 1 | Introduction to the principal concepts in computer science using Karel J. Robot<br><br>**Topics:**<br>• Objects<br>• Classes<br>• Looping<br>• Conditionals<br><br>**Objectives:**<br>• Write and use simple classes with Karel J. Robot<br>• Learn the basics of conditionals and looping in a Java environment<br>• Reinforce the introductory concepts of the Java programming language<br>• Reinforce the steps involved in program compilation/execution | **Resource:**<br>Karel J. Robot<br><br>**Sample Assessments:**<br>• Program specific tasks for Karel<br>• Create an enhanced Robot class to expand the number of commands that Karel understands<br>• Use loops to clear a field of beepers<br>• Use loops and conditionals to redistribute a field of beepers<br>• Use loops and conditionals to run a hurdle race of varying lengths and hurdle sizes |
| 2 | Introduction to the computer, ready-made programming environments, and Java basics<br><br>Students also walk through Part One of the GridWorld Case Study.<br><br>**Topics:**<br>• Computer hardware and software<br>• Computer networks<br>• Representing numbers in different bases<br>• Java program features (comments, identifiers, reserved words)<br>• Ethical and social implications of computer use<br><br>**Objectives:**<br>• Describe the relationship | **Resources:**<br>• AP GridWorld Case Study<br>• Lewis, Loftus, Cocking: Chapter One<br><br>**Sample Assessments:**<br>• Student presentation (one topic per student) on hardware components, computer architecture, or computer networking<br>• Self-made “History of Programming” quiz<br><br>**Strategies:**<br>• Students will have a mock debate in which students argue/defend both sides of a current topic in computer ethics (e.g. pirating music, making copies of programs, etc.) |

| | | |
|---|---|---|
| | between hardware and software<br>• Define various types of software and how they are used<br>• Identify basic computer hardware and explain what it does<br>• Explain how hardware components execute programs and manage data<br>• Describe how computers are connected together into networks to share information<br>• Understand computer ethics such as acceptable use policies, copyright, intellectual property, freeware and shareware | |
| 3 | Objects, primitive data, variables, and expressions<br><br>**Topics:**<br>• Program development<br>• Reinforcement of objects and classes<br>• Primitive data types<br>• Strings and escape sequences<br>• Variables and assignment<br>• Expressions<br>• Data conversion and data types<br>• Simple input/output<br>• Random number generation<br>• Libraries and packages<br>• Formatting output<br><br>**Objectives:**<br>• Discuss basic program development steps<br>• Understand terminology: variables, constants, reserved words, literals<br>• Define the difference between primitive data and objects<br>• Declare and use variables<br>• Perform mathematical computations<br>• Create objects and use them<br>• Reinforce aforementioned | **Resources:**<br>• Lewis, Loftus, Cocking: Chapter Two and Three<br>• Lewis, Loftus, Cocking: Chapter 2 Internet AP GridWorld tie-in supplement<br><br>**Sample Assessments:**<br>• Chapter questions, multiple choice, true/false, short answer<br>• Sum/difference/product program: Write an application that reads floating point numbers and computes their sum, difference, and product<br>• Random phone number program: Write an application that creates and prints a random phone number in a specified format<br><br>**Strategies:**<br>• Students are led through their first “pure” Java program (Hello World) in order to explore the format of a properly written program<br>• Students need multiple examples and practice with types and type conversions, |

| | | |
|---|---|---|
| | chapter objectives through the use of the GridWorld case study | especially when they are used in mathematical expressions |
| 4 | Conditionals and Repetition<br><br>**Topics:**<br>• Flow of control<br>• The if statement<br>• Equality, relational, increment/decrement, and assignment operators<br>• Logical operators<br>• Short-circuiting<br>• The while statement<br>• The for statement<br>• Infinite loops<br>• Nested loops<br>• Iterators<br><br>**Objectives:**<br>• Define the flow of control through a program<br>• Reinforce the use of if statements<br>• Define expressions that let us make complex decisions<br>• Reinforce the use of while and for statements to repeat programmatic actions<br>• Reinforce aforementioned chapter objectives through the use of the GridWorld case study | **Resources:**<br>• Lewis, Loftus, Cocking: Chapter Three<br>• Bloss and Ingram: Lab Manual<br>• Lewis, Loftus, Cocking: Chapter 3 Internet AP GridWorld tie-in supplement<br><br>**Sample Assessments:**<br>• Chapter questions, multiple choice, true/false, short answer<br>• "Computing a Raise" lab<br>• "Charge Account" lab<br>• "Date Validation" lab<br>• "Rock, Paper, Scissors" Lab<br>• "Factorials" lab<br>• "Counting Vowels" lab<br><br>**Strategies:**<br>• Students need practice writing different types of loops and conditionals |
| 5 | Writing classes, enhancing classes, and inheritance<br><br>**Topics:**<br>• Creating and using classes<br>• Inheritance<br>• Abstract classes<br>• Interfaces<br>• Polymorphism<br><br>**Objectives:**<br>• Define classes that act as blueprints for new objects<br>• Explain encapsulation and Java modifiers | **Resources:**<br>• Lewis, Loftus, Cocking: Chapters Four, Five, and Seven<br>• Bloss and Ingram: Lab Manual<br>• Lewis, Loftus, Cocking: Chapters 4, 5, and 7 Internet AP GridWorld tie-in supplement<br><br>**Sample Assessments:**<br>• Chapter questions, multiple choice, true/false, short answer<br>• "Coin Class" lab<br>• "Bank Account" lab<br>• "Tracking Grades" lab"<br>• "Tracing References" lab |

| | | |
|---|---|---|
| | • Explore the details of method declaration, invocation, parameter passing, and overloading<br>• Learn to divide complex methods into simpler supporting methods<br>• Describe relationships between objects<br>• Define reference aliases<br>• Define formal interfaces and their class implementations<br>• Derive new classes from existing classes through the use of inheritance<br>• Add and modify methods in child classes<br>• Discuss how to use class hierarchies<br>• Define polymorphism and methods to achieve it<br>• Reinforce aforementioned chapter objectives through the use of the GridWorld case study | • "Counting Transactions" lab<br>• "Exploring Inheritance" lab<br>• "Test Questions" lab<br><br>**Strategies:**<br>• Give students classes to complete, in which they are given a description and they must choose appropriate representation for that class<br>• Draw pictures of the inheritance hierarchy |
| 6 | Advanced programming structures and algorithms (arrays/ArrayLists, and sorting/searching algorithms)<br><br>**Topics:**<br>• Declaring and initializing arrays<br>• Manipulating arrays with loops<br>• Creating parallel arrays<br>• Using the ArrayList class<br>• Bubble, Selection, and Insertion sorts<br>• Sequential and Binary searches<br><br>**Objectives:**<br>• Understand terminology: array, element, index, logical size, physical size, parallel arrays<br>• Declare one-dimensional arrays in Java<br>• Use initializer lists when declaring arrays<br>• Manipulate arrays using loops | **Resources:**<br>• Lewis, Loftus, Cocking: Chapter Six<br>• Bloss and Ingram: Lab Manual<br>• Lewis, Loftus, Cocking: Chapter 6 Internet AP GridWorld tie-in supplement<br><br>**Sample Assessments:**<br>• Chapter questions, multiple choice, true/false, short answer<br>• "Tracking Sales" lab<br>• "Grading Quizzes" lab<br>• "Reversing an Array" lab<br>• "Searching/Sorting Within an Integer List" lab<br>• "ArrayList Shopping Cart" lab<br><br>**Strategies:**<br>• Students need to be reminded that array indices start at zero<br>• Students need materials to |

| | | |
|---|---|---|
| | and array indices<br>• Use the physical and logical size of an array to guarantee they do not go beyond the bounds of the array<br>• Understand how parallel arrays can be useful when processing certain types of data<br>• Work with arrays of primitives and well as objects<br>• Understand when to choose an array to represent data instead of an ArrayList<br>• Use the ArrayList methods<br>• Write a method for searching an array<br>• Perform insertions and deletions at given array positions<br>• Trace through sorting and searching algorithms to understand time constraints of each<br>• Understand the algorithms behind bubble, selection, and insertion sorts and sequential and binary searches<br>• Understand the time efficiency of each sort and search and when it is desirable to use each one<br>• Reinforce aforementioned chapter objectives through the use of the GridWorld case study | reinforce the use of looping through elements of an array<br>• Stress the difference between add and set<br>• Draw pictures of the ArrayList after ArrayList methods have been used<br>• Use one of several Internet sites that show the runtime and efficiency of each of the strategies they have learned |
| 7 | Recursion and Merge Sort<br><br>**Topics:**<br>• Recursion<br>• Merge Sort<br><br>**Objectives:**<br>• Create a recursive method to solve a problem<br>• Understand the difference between recursive and iterative solutions to a problem<br>• Understand and use the Merge | **Resources:**<br>• Lewis, Loftus, Cocking: Chapter Eight<br>• Bloss and Ingram: Lab Manual<br>• Lewis, Loftus, Cocking: Chapter 8 Internet AP GridWorld tie-in supplement<br><br>**Sample Assessments:**<br>• Chapter questions, multiple choice, true/false, short answer<br>• “Palindrome Checker” lab<br>• “String Backwards” lab |

| | | |
|---|---|---|
| | Sort<br>• Understand how to calculate the informal runtime of merge sort and compare its running time to the other sorts already learned<br>• Reinforce aforementioned chapter objectives through the use of the GridWorld case study | • "Base Conversion" lab<br>• Self-made recursion worksheet<br><br>**Strategies:**<br>• Recreate the "Towers of Hanoi" using Fisher-Price ring sets |
| 8 | AP GridWorld Case Study (continued)<br><br>**Topics:**<br>• Experimenting with a large program<br>• Using classes<br>• Modifying classes<br>• Inheritance<br><br>**Objectives:**<br>• Run the case study and analyze output<br>• Understand how the development of a large program came about by reading the chapters of the case study<br>• Observe and experiment with the GridWorld case study<br>• Understand the Bug class, Runner class, Grid Interface<br>• Extend the Bug class by creating a specialized bug to meet new requirements<br>• Use inheritance to extend the Critter class by making new types of critters | **Resources:**<br>• AP GridWorld Case Study<br><br>**Sample Assessments:**<br>• Exercises from within the case study<br><br>**Strategies:**<br>• Read the manual for the case study thoroughly<br>• Be familiar with all the classes and interfaces discussed<br>• Allow the students to be creative after working through the exercises and analysis<br>• Create different kinds of Critters |
| 9 | Getting Ready for the AP Exam<br><br>**Topics:**<br>• Review AP Computer Science A topics<br><br>**Objectives:**<br>• Prepare for the AP Computer Science A Exam by reviewing material and taking practice exams | **Resources:**<br>• Previous free-response questions from AP Central<br><br>**Sample Assessments:**<br>• Practice Exams |

| | | |
|---|---|---|
| 10 | Robotics Using Java<br><br>**Topics:**<br>• Movement<br>• Navigation<br>• Light sensors<br>• Touch Sensors<br>• Behavior Arbitration<br>• RCX Communication<br><br>**Objectives:**<br>• Transfer Java programming concepts and techniques learned throughout the course of the year to a new setting and environment<br>• Design, build, and test Java programs that solve a number of problems<br>• Use the Lejos API to expand their knowledge of the Java programming language | **Resources:**<br>• LEGO® MindStorms™ kits<br>• Various Internet sites (most importantly, www.lejos.org)<br>• Various self-made handouts and presentations<br><br>**Sample Assessments:**<br>• "Line Follower" task<br>• "Boundary" task<br>• "Hit-backup-turn" task<br>• "Green-Red Liner" task<br>• "Sumobot" task<br><br>**Strategies:**<br>• Students are encouraged to have fun and be creative<br>• Tasks are not given hard delivery dates, but a Task Signoff sheet must be completed by the end of the marking period |

**Correlation to AP Topic Outline**

This section shows correlation between the "Computer Science A" column of the Topic Outline in the *AP Computer Science Course Description* and each unit of this syllabus.

| AP Computer Science A Topics | Unit Where Covered |
|---|---|
| I. Object-Oriented Program Design<br>The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved. | |
| A. Program design | |
| 1. Read and understand a problem description, purpose, and goals. | All units |
| 2. Apply data abstraction and encapsulation. | Units 3 and 5 |
| 3. Read and understand class specifications and relationships among the classes (‘‘is-a,’’ ‘‘has-a’’ relationships). | Unit 5 |
| 4. Understand and implement a given class hierarchy. | Unit 5 |

| | |
|---|---|
| 5. Identify reusable components from existing code using classes and class libraries. | Unit 5 |
| B. Class design | |
| 1. Design and implement a class. | Unit 5 |
| 2. Design an interface | Unit 5 |
| 3. Choose appropriate data representation and algorithms. | Units 4 and 5 |
| 4. Apply functional decomposition. | Unit 5 |
| 5. Extend a given class using inheritance. | Unit 5 |
| II. Program Implementation<br>The overall goals of program implementation parallel those of program design. Classes that fill common needs should be built so that they can be reused easily in other programs. Object-oriented design is an important part of program implementation. | |
| A. Implementation techniques | |
| 1. Methodology | |
| a. Object-oriented development | Unit 3 |
| b. Top-down development | Unit 4 |
| c. Encapsulation and information hiding | Unit 5 |
| d. Procedural abstraction | Unit 5 |
| B. Programming constructs | |
| 1. Primitive types vs. objects | Unit 3 |
| 2. Declaration | |
| a. Constant declarations | Unit 3 |
| b. Variable declarations | Unit 3 |
| c. Class declarations | Unit 5 |
| d. Interface declarations | Unit 5 |
| e. Method declarations | Unit 5 |
| f. Parameter declarations | Unit 5 |
| 3. Console output (System.out.print/println) | Unit 3 |
| 4. Control | |
| a. Methods | Unit 5 |
| b. Sequential | Unit 4 |
| c. Conditional | Unit 4 |
| d. Iteration | Unit 4 |
| e. Recursion | Unit 7 |
| C. Java library classes (included in the A-level (AP Java Subset) | Units 3 and 5 |
| III. Program Analysis<br>The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms in order to understand their time and space requirements when applied to different data sets. | |
| A. Testing | |
| 1. Test classes and libraries in isolation. | Unit 5 |
| 2. Identify boundary cases and generate appropriate test data. | Unit 5 |

| | |
|---|---|
| 3. Perform integration testing. | Unit 5 |
| B. Debugging | |
| 1. Categorize errors: compile-time, run-time, logic. | Units 2 and 3 |
| 2. Identify and correct errors. | Units 2, 3, 6, and 7 |
| 3. Employ techniques such as using a debugger, adding extra output statements, or hand-tracing code. | Units 2, 3, 6, and 7 |
| C. Understand and modify existing code | Units 2, 3, 4, 5, 6, 7, and 8 |
| D. Extend existing code using inheritance | Unit 5 |
| E. Understand error handling | |
| 1. Understand runtime exceptions. | Unit 5 |
| F. Reason about programs | |
| 1. Pre- and post-conditions | Unit 5 |
| 2. Assertions | Unit 5 |
| G. Analysis of algorithms | |
| 1. Informal comparisons of running times | Units 6 and 7 |
| 2. Exact calculation of statement execution counts | Units 6 and 7 |
| H. Numerical representations and limits | |
| 1. Representations of numbers in different bases | Unit 2 |
| 2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error) | Units 3 and 4 |
| IV. Standard Data Structures<br>Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures. | |
| A. Simple data types (int, boolean, double) | Unit 3 |
| B. Classes | Units 3 and 5 |
| C. One-dimensional arrays | Unit 6 |
| V. Standard Algorithms<br>Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency. | |
| A. Operations on A-level data structures previously listed | |
| 1. Traversals | Unit 6 |
| 2. Insertions | Unit 6 |
| 3. Deletions | Unit 6 |
| B. Searching | |
| 1. Sequential | Unit 6 |
| 2. Binary | Unit 6 |
| C. Sorting | |
| 1. Selection | Unit 6 |
| 2. Insertion | Unit 6 |
| 3. Mergesort | Unit 7 |

| VI. Computing in Context<br>A working knowledge of the major hardware and software components of computer systems is necessary for the study of computer science, as is the awareness of the ethical and social implications of computing systems. These topics need not be covered in detail but should be considered throughout the course. | |
|---|---|
| A. Major hardware components | |
| 1. Primary and secondary memory | Unit 2 |
| 2. Processors | Unit 2 |
| 3. Peripherals | Unit 2 |
| B. System software | |
| 1. Language translators/compilers | Unit 2 |
| 2. Virtual machines | Unit 2 |
| 3. Operating systems | Unit 2 |
| C. Types of systems | |
| 1. Single-user systems | Unit 2 |
| 2. Networks | Unit 2 |
| D. Responsible use of computer systems | |
| 1. System reliability | Unit 2 |
| 2. Privacy | Unit 2 |
| 3. Legal issues and intellectual property | Unit 2 |
| 4. Social and ethical ramifications of computer use | Unit 2 |