AP Computer Science (A) Course Syllabus 2015-2016

Curricular Requirements		Page(s)
CR1	The course teaches students to design and implement computer-based solutions to problems.	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
CR2a	The course teaches students to use and implement commonly used algorithms.	12, 13, 16, 17, 18, 19, 20
CR2b	The course teaches students to use commonly used data structures	12, 13, 14, 15, 16, 17, 19, 20
CR3	The course teaches students to select appropriate algorithms and data structures to solve problems.	17, 19, 20
CR4	The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.	10, 11, 14, 15, 17, 19, 20
CR5	The course teaches students of the standard Java library from the AP Java subset in Appendix A of the AP Computer Science A Course Description.	5, 7, 8, 9, 13, 14, 20
CR6	The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.	1
CR7	The course teaches students to recognize the ethical and social implications of computer use.	2, 3, 9, 16, 20

Texts

Schram, Leon. *Exposure Java 2014.* Royse City, TX: Leon Schram, 2014
http://www.schram.org

College Board. AP Computer Science Course Description 2014. New York: College Entrance Examination Board, 2014.
http://apcentral.collegeboard.com

Schram, Leon*. Multiple-Choice & Free-Response Questions in Preparation for the AP Computer Science Examination* 9th Ed. New York: D&S Marketing, 2015
http://www.dsmarketing.com

Syllabus at a Glance

Semester 1	Semester 2
I - Introduction to Computer Science	XI - One-Dimensional Static Arrays
II - Introduction to Programming in Java	XII - One-Dimensional Dynamic Arrays with ArrayList
III - Java Primitive Data Types and String	XIII - Interfaces, Concrete Classes and Abstract Classes
IV - Using Methods and Parameters	XIV - Focus on OOP, Polymorphism
V - Control Structures and Boolean Logic	XV - Two-Dimensional Arrays with the Picture AP Lab
VI - Using Class Methods and Object Methods	XVI - Focus on OOP, Object Oriented Design
VII - Creating Class Methods	XVII - Recursion
VIII - The String Class and the AP Magpie Chatbot Lab	XVIII- Algorithms and Informal Algorithmic Analysis
IX - Focus on OOP, Encapsulation	XIX - Preparation for the AP Computer Science Exam
X - Focus on OOP, Inheritance & Composition	XX - Post AP Exam I, Sequential Files
	XXI - Post AP Exam II, Advanced Graphics Project

Students complete **24 Lab Assignments** in the course prior to the AP Exam. Students know that they will not have sufficient time to complete any major lab assignments during scheduled school lab time. About 8 lab assignments are minor and can be finished in 1-Hour. The remaining 16 major lab assignments take a minimum of 2-Hours and half the major assignments will require 3-4 hours for completion. Students are also required to participate with three Saturday sessions, which are about 50% hands-on labs. A conservative estimate is that students devote a minimum of **20 hours** in hands-on labs during scheduled class-time and roughly **30 hours** in hands-on labs outside school hours for a total minimum of **50 hours** hands-on lab time.

Unit I	**Introduction to Computer Science - 1 Week**
General Objectives	Students get a brief introduction about computer science by learning about the history of Computer Science, storing and computing information in base2 and base16, computer hardware and software, program languages, computer operating systems and networks.
Topics, Sub-Topics, Details to be Covered	• Learn how to learn computer science • A brief history of computing devices • Talk about *Pirates of Silicon Valley* and ethical issues leading into today's *Identity Theft* • Computers and number systems o Converting between **Base-2**, **Base-10** and **Base-16** o Use of **ASCII** and **Unicode** to store characters • Primary memory and secondary memory devices • Computer processors • Computer hardware and peripheral devices • What is programming? • A brief history of Program languages • Computer operating systems like Windows, Unix, Mac OS • Single user systems • Networks o Peer-to-peer networks o LANs, WANS, Intranet, Internet o Wired networks o Wireless networks
Resources, Evaluations	**Exposure Java, Chapter 1, Introduction to Computer Science** 2 Reading Quizzes and 1 Multiple-Choice Chapter Test
Class Tasks, Lab Assignments	Students perform calculations at the desk in different number systems. They also come forward in groups of 8, representing a byte in the computer. Students use a card showing 0 and 1 and then represent different decimal numbers in binary. The first unit has no program lab assignments.
Teaching Strategies, Best Practices	This first unit is very depended on the knowledge of incoming students. Most students should have received training in computer history and computer applications prior to enrolling in AP Computer Science. At the same time, the manner in which a computer stores information, processes information, the concept of programming and networks is usually not taught. The *Pirates of Silicon Valley* is an interesting story and a good introduction discussing the abuse of copyright and the start of ethical concerns like *Software Piracy*, *Patent Abuse* long before today's concern with *Identity Theft* and the implications of using Social Media. It is important that the pace of the course is established in the first two weeks. There are many topics to be covered. Every class starts with a ***DYRT*** quiz. This means ***D****id* ***Y****ou* ***R****ead* ***T****his?* The quizzes are presented as multiple-choice, timed Power Point slides. The quiz starts one minute after class starts. This type of quiz sets the "tone" of the class immediately and avoids wasted time at the start of class. A fun exercise with students is for the teacher to be a "robot" who follows a precise program of instructions to pick up chalk or marker and draw a circle on the board. It is even more fun with putting peanut butter on a slice of bread. Another exercise is to line up eight students, representing a byte. Each student is a bit with a place hold value. Students are "on" facing the class and "off" with their back turned. Use the student "byte" to represent base-10 numbers in binary memory.
Curriculum	

Requirements	CR #7

Unit II	**Introduction to Programming in Java - 1 Week**
General Objectives	Students learn how to download, install and use the software tools to write, compile and execute Java programs. Students also learn about the ethical and legal implications of using computers.
Topics, Sub-Topics, Details to be Covered	• Platform dependent and platform independent languages • The need for computer translators o Translate one-line and then execute one line with an **interpreter** o Translate the entire program into machine code and then execute with a **compiler** o How Java creates **Bytecode** and uses both a **compiler** and **interpreter** • Java **application** programs and **applet** programs • Student download and install Java software o Download and installing the **JDK** and the **JRE** o Download and install a free **IDE**, like **jGrasp** • Responsible use of computer software, hardware o Hardware care o Protect against surges and power outs o Back up data o Protect against viruses and identity theft • Students become aware of social, ethical and legal implications of using computers o Protecting privacy concerns o Intellectual property, copyright issues o Software piracy problems o Social network issues o Explain how the download was Java software is legal. o Difference between free software, evaluation software, and purchased software • Setting up the Java programming workspace o Check if necessary Java software (JDK & IDE) is properly installed o Explain the IDE components o Students practice loading, compiling and executing existing programs o Java input/output issues • Java creates text program output with **print** and **println** • Examples of common Java compile errors
Resources, Evaluations	**Exposure Java, Chapter 2, Introduction to Programming in Java** 2 Reading Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test
Class Tasks, Lab Assignments	Students download Java software from the Internet. They learn how to recognize when software is free, when it is a trial version and when payment is required. Issues of copyright, software piracy are discussed. Students learn to install the JDK, JRE and an IDE like jGrasp. Students do their first Lab Assignment, which requires copying, altering and creating different program output showing knowledge of using the IDE and the Java's **print** and **println** methods.
Teaching Strategies, Best Practices	The main purpose of this unit is to teach students how to use the Java JDK and an IDE to write programs. The intent at this stage is to learn the mechanics of working with the software and understanding the compiling and executing process. In particular, students will learn the difference between compiler and interpreter translators. Once this is clear, students then learn how Java uses both translators. The responsible use of the computer is taught throughout the year as good teachable moments arise. Unit 2 lends itself quite well for a discussion after the Unit 1 introduction. Students are shown how to download the Java software and jGrasp. Both downloads are free and it is shown on the web site that the software is free. From this stepping stone of what is free, what is shared and what must be paid motivates a discussion into general issues of the ethical use of the computer.

Curriculum Requirements	CR#1, CR#7
Unit III	**Java Primitive Data Types and String - 1 Week**
General Objectives	Students learn the Java primitive data types and operations on **int**, **double**, **char**, **boolean** and also learn to use **String** in a manner that is similar to the primitive data types.
Topics, Sub-Topics, Details to be Covered	• Declaring and operating with numerical simple/primitive data types • Integer types (**int**, **byte, short, long**) • Integer binary operations (+, -, *, /, %) • Real number types (**double, float**) • Real number binary operations (+, -, *, /, %) • Arithmetic shortcut notations (++, --, +=, -=, *=, /=, %=) • Limitations of finite representations o Memory overflow resulting in imprecision o Mathematical accuracy and computer accuracy o Round-off errors and real number representations • Character type (**char**) • Boolean type (**boolean**) • String type (**String**) • String concatenation with (+) • Type casting with (**int)** and other data types • Declaring constants with **final** • Single line documentation with // • Multiple line documentation with **/* */** • Mathematical precedence in programs • Escape sequences (**\n,** \\, **\")** • The AP Java subset importance • The testing rationale and the subset need • The importance for learning non-tested topics
Resources, Evaluations	**Exposure Java, Chapter 3, Java Primitive Data Types and String** 2 Reading Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	The introduction of new programming features in Java are presented in "Lab Lectures" that have students examine program code on their own computers and alter code to observe execution changes. Students can also quickly perform brief, non-graded lab-experiments to gain better hands-on experience. **Inches to Feet & Miles and Seconds to Minutes and Hours Lab Assignment** Students complete a lab assignment that involves knowledge of Java primitive data types and binary operations by dividing inches into feet & miles and also seconds into minutes & hours.
Teaching Strategies, Best Practices	Unit 3 is where students start to learn the concept of writing a program. At this stage all program writing is done in the **main** method. Students are provided with a program template. The **String** class is included in this chapter. It is included without teaching methods or instantiation. Students need to appreciate that the mathematical reality they have learned does not always apply to a computer. There are round-off errors, imprecise representations of floating point numbers and they need to understand what happens when a number is stored that is too large for its memory location. The course material is not taught in a *teacher lectures and students listen style*. The presentation is a *lab lecture*. Students follow the *discussion/demonstration* with their computers, do certain experiments as appropriate and try out program concerns and confusion and they have questions.

Curriculum Requirements	CR#1

Unit IV	**Using Methods and Parameters - 1.5 Weeks**
General Objectives	Students learn how to recognize the difference between primitive data types, classes and objects. They also learn how to use methods and how to use parameters. The chapter concentrates on the methods of the **Math** class and the methods of the **Graphics** and **Color** classes.
Topics, Sub-Topics, Details to be Covered	• A brief history of program design • OOP, a gentle first exposure of **encapsulation**, **inheritance**, **composition** & **polymorphism** • Procedural abstraction • Data abstraction • Java's standard **Math** class, which is loaded automatically and does not require **import** o Methods **abs**, **pow**, **sqrt**, **random** o Methods **floor**, **ceil**, **round**, **max**, **min** o Field attributes **PI**, **E** • Access of methods in a user-defined class is identical to standard Java library methods • Accessing standard Java classes with packages using **import** • Methods with multiple parameters • Differences between application program and applet programs • Compiling and executing applet programs • Using the **Graphics** and **Color** classes of the **java.awt** package, which requires **import** o Methods **drawLine**, **drawRect**, **drawOval** o Potential parameter confusion with multiple parameters o Altering graphics color with method **setColor**
Resources, Evaluations	**Exposure Java, Chapter 4, Using Methods and Parameters** 2 Reading Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students are doing desk exercises on paper learning interest computations that must be translated into Java program statements for a later lab assignment. **Mortgage Payment & Amortization Table Lab Assignment** Students complete a lab assignment that uses **Math** class methods and binary operations to compute mortgage payments and amortization schedules. **Multiple Graphics Objects Lab Assignment** Students complete a graphics lab assignment where they need to use coordinate geometry with calling methods and parameter passing to create a cube, sphere, star and other graphics objects on the monitor.
Teaching Strategies, Best Practices	Unit 4 introduces methods and parameters. This is hardly a complete unit on OOP. Students do learn some fundamental OOP terminology and realize the importance of using code that has already been written. In this case students use methods of several provided classes. The Mortgage Payment assignment is a good lab assignment to write a program that involves binary operations and Math class methods. It is also an assignment in the context of teaching students about monthly payments for house mortgages, amortization schedule and comprehending interest better. Ask students the expected monthly payment of their choice. They are very surprised when they see how much higher the cost is than they expected. Graphics is intentionally introduced early. The use of graphics is optional in an AP course, and as such it is not tested. The reality is that students like graphics programs and are far more motivated to work on lab assignments with graphics output. There are other benefits. Calling graphics methods gives students

	excellent practice with methods that use multiple parameters.
Curriculum Requirements	CR#1, CR#5

Unit V	**Control Structures and Boolean Logic - 1.5 Weeks**
General Objectives	Students learn to direct program flow with iterative and selection control structures. They also learn to handle both simple conditions and compound conditions following laws of Boolean logic.
Topics, Sub-Topics, Details to be Covered	• Types of control structures o Simple sequence o Selection o Iteration • Relational operators (==, !=, <, <=, >, >=) • Selection constructs o One-way selection with **if** o Two-way selection with **if...else** o Multiple-way selection with **switch...case...break** • Iteration constructs o Fixed iteration with **for** o Pre-condition iteration with **while** o Post-condition iteration with **do..while** • Performing a variable trace • Boolean operations o Truth tables o Compound conditions with (!, &&, ||) o Program input protection o DeMorgan's Law o Short-circuiting • Output exercises • Using control structures and graphics together
Resources, Evaluations	**Exposure Java, Chapter 5, Control Structures and Boolean Logic** **D&S Marketing APCS Prep, Chapter 1, Control Structures** **D&S Marketing APCS Prep, Chapter 3, Boolean Logic** 2 Reading Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	In Unit V for the first time students perform *output exercises* in class, on paper. It gets them in the habit to think like a computer and create a variable trace to follow the program execution and the values of the variables. This is a skill that they need to debug their programs and answer multiple-choice questions. **Turning Many Graphics Straight Lines Into Curves Lab Assignment** Students complete a single, challenging lab assignment. The assignment creates a graphics picture that displays many curves, yet all the lines are straight. The illusion is achieved with many straight lines that are displayed in many loops and follow compound conditions of Boolean logic to achieve the correct result. For extra credit the pattern repeats itself; it becomes smaller, but shows the same appearance.
Teaching Strategies, Best Practices	In this unit all control structures start with single conditions. This is followed by several sections that explain Boolean logic and compound conditions. Additional control structures follow that now show how to use compound conditions. Special emphasis is placed on showing the concept of *short circuiting.* The lab assignment continues the graphics theme started in unit 4. Students use control structures to draw hundreds of straight lines. The end result is an interesting graphics design. This unit also introduces students to variable tracing. Worked out exercises are provided for students to

	teach them to "play computer" and learn how to determine program output without using a computer.
Curriculum Requirements	CR#1,

Unit VI	**Using Class Methods Object Methods - 1.5 Weeks**
General Objectives	Students learn the difference between Classes and Objects, as well as between Class Methods and Object Methods. They learn that calling object methods requires the instantiation of an object with the **new** operator. Students also learn the importance of randomness in their daily lives and how randomness is generated in a Java program.
Topics, Sub-Topics, Details to be Covered	• Classes and objects • Using **new** to instantiate a new object • Calling object methods vs calling class methods • Using overloaded constructors by recognizing the *method heading signature* • Generating random values o Using the **Random** class o Methods **nextInt**, **nextDouble**, **setSeed** o Using **Math.random** o Generating any desired random range • Using the **Graphics** class o The hidden **new** operator of the **Graphics** object o Additional **Graphics** methods **setColor**, **drawPolygon**, **fillPolygon** • Using the **Polygon** class o Method **addPoint** • Constructing custom **Color** objects • Anonymous objects and concrete objects • Displaying random graphics objects with random colors • Wrapper classes o **Integer**, **Double**, **Boolean** o **Integer.MAX_VALUE** o **Integer.MIN_VALUE** o Using **intValue**
Resources, Evaluations	**Exposure Java, Chapter 6, Using Class Methods and Object Methods** 2 Reading Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students do several class exercises creating Java statements that generate a specified random range. Conversely they do exercises with program statements and determine the random range. Students complete a sequence of timed output exercises. **Random Objects With Random Colors Lab Assignment** Students complete a lab assignment with 8 graphics cells. Each cell feature a group of randomly colored objects, such as lines, circles, squares, etc. All randomness generated is done with **Math.random**.
Teaching Strategies, Best Practices	Unit six teaches students to use many classes that use object methods. It is now necessary to define an object with the **new** operator before any of the object methods can be called. This unit teaches that methods of the **Graphics** class are called with a **Graphics** object that appears to exist without the use of the **new** operator. This is an illusion, because the **Graphics** object is constructed in the background and then passed to the **Graphics** object parameter of the **paint** method. Wrapper classes are introduced and will be important for future topics where data structures are only

	capable of storing objects.
Curriculum Requirements	CR#1, CR#5

Unit VII	**Creating Class Methods 1.5 Weeks**
General Objectives	Students learn how to create their own methods and break up large segments of code into methods for a common purpose.
Topics, Sub-Topics, Details to be Covered	• The **Math** class revisited, looking at additional **Math** methods • Modular Programming and user-created methods • User-created parameter methods • Using **static** to declare *static methods* or *class methods* • Actual and formal parameters • Parameter passing rules • **void** methods with one and more parameters • **return** methods with one and more parameters • Using **return** to exit a method with returning a value • Introduction to program design • Creating methods with other methods • Making a utility library class • A pre-OOP program design investigation with the *Payroll* class case study
Resources, Evaluations	**Exposure Java, Chapter 7, Creating Class Methods** **D&S Marketing APCS Prep, Chapter 2, Methods and Parameters** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students are introduced to doing Free-Response quizzes in a time limit for the first time. The questions and quizzes in the first semester are not yet at the AP Exam, free-response level. Students participate in an output class exercise. **The Geometric Functions Lab Assignment** Students complete a Geometry program which computes the circumferences, areas and volumes of many geometric objects. Students are expected to create a **Geometry** class and write all the class methods. **Open-Ended Two-Person Graphics Picture Lab Assignment** Students complete a second program with a partner. They must design a program of multiple classes that creates a graphics picture, such as a house in a yard. Methods must be written for separate functionalities.
Teaching Strategies, Best Practices	The chapter starts with the idea of modular programming and the syntax required to create your own classes and methods. Students learn only to create classes with static or class methods. This allows an easier introduction than creating classes with constructors and object methods. The Payroll case study presents a very, very poorly designed program where every confusing program statement is shoved into the main method. The case study then demonstrates step-by-step how to improve the poorly designed program. This case study is not yet a good example of Object Oriented Programming. It is a good start to teach important programming design concepts. As additional concepts are taught so will additional features of program design get introduced. This unit introduces the new "write-a-method" Free-Response quiz style of evaluation. This style of quiz

	will steadily increase in frequency and help prepare students for the free response section of the AP Computer Science Examination.
Curriculum Requirements	CR#1, CR#5

Unit VIII	**The String Class and the AP Magpie Chatbot Lab - 2 Weeks**
General Objectives	Students learn many methods to manipulate **String** objects. They are also introduced to the first of the three AP Labs, the Magpie Chatbot Program. In this unit the Magpie lab is used for practice with **String** methods, compound selection structures and analyzing many different possible cases.
Topics, Sub-Topics, Details to be Covered	• Constructing **String** objects • Using common **String** methods o Using method **length** to measure the size of a string o Using method **substring** with one parameter to return a string segment o Using method **substring** with two parameters to return a string segment o Using **indexOf** with one parameter to find the first occurrence of a string o Using **indexOf** with two parameters to find multiple occurrence of a string o Using **equals** to check **String** object equality o Using **compareTo** with two **String** object to check equality and inequality. o Altering string with methods **trim**, **upperCase** and **lowerCase** • Introduction to the AP Magpie Lab Case Study o Explain ethical implications of using and altering the AP Labs o Identify Laurie White of Mercer University as the AP Magpie Lab Program Developer o Using the fundamental Magpie Chatbot for simple responses o Adding random responses to the Chatbot o Finding the shortcomings of the negative response in the Chatbot o Improving the negative Chatbot response with many cases and compound conditions
Resources, Evaluations	**Exposure Java, Chapter 8, The String Class** **College Board AP Magpie Chatbot Lab Materials** **D&S Marketing APCS Prep, Chapter 8, String Methods** 2 Reading Quizzes, 2 Free-Response Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students perform multiple classroom lab exercises with the AP Magpie Chatbot program. These are not yet lab assignments, but short lab exercises where students make changes in the existing program to observe the consequences or try to create improvement for undesirable executions. **The Palindrome and Almost-Palindrome Lab Assignment** Students complete a **String** manipulation lab assignment. In this assignment a string is entered and students must write method **isPalindrome**. Additionally, after determining that the string is not a palindrome other methods must determine if it is almost a palindrome. This is done by removing all characters, punctuation and altering the string to upper-case.
Teaching Strategies, Best Practices	The three AP Labs are used in this syllabus not as complete stand-alone units. Different segments of the labs are introduced at the point when prerequisite knowledge is adequate and at such a time that the AP Labs assist in teaching the main concepts of the unit. For instance, students have performed various assignments, but they have not yet needed to consider cases. The AP Magpie lab does a great job for that skill. One type of Chatbot response is based on finding the string **no** in an answer. Initially, that may seem simple. Students quickly learn that **no** is also found in words like k**no**w, **no**tice and **no**rth. Additionally, there are differences between **no** in the middle of a sentence, at the beginning of a sentence, at the end of a sentence or **no** strictly by itself. This creates a great opportunity for learning "cases."

	Later during the free-response section of the AP Computer Science Examination, students need to complete many methods. Most students loose a considerable amount of points, because their solutions are incomplete. Frequently, the solution is correct, but as it does not consider all the possible cases that can happen, points are lost.
Curriculum Requirements	CR#1, CR#5, CR#7

Unit IX	**Focus on OOP, Encapsulation - 2 Weeks**
General Objectives	Students are now learning Object Oriented Programming seriously in a series of units, titled *Focus On OOP*. The first unit concentrates on *Encapsulation* and it also introduces the **Card** class that will become a fundamental building class of the later **Deck** class of the **AP Elevens Lab**.
Topics, Sub-Topics, Details to be Covered	• Review of OOP Concepts o Encapsulating methods and attributes in the same class o Reusing existing classes with an "is-a" inheritance relationship o Reusing existing classes with a "has-a" composition relationship o Creating greater program efficiency and reliability with polymorphism • Introduction to the **Card** Class o General object method syntax o Default constructor methods o Overloaded constructor methods o Control class member access with **private** and **public** declarations o Creating "get" return object methods o Creating "set" modifying void object methods • Encapsulation with the Cube Case Study • Understanding the **scope** of a variable • Proper use of the **this** reference
Resources, Evaluations	**Exposure Java, Chapter 9, Focus on OOP, Encapsulation** **College Board AP Elevens Lab Materials** **D&S Marketing APCS Prep, Chapter 4, OOP, Encapsulation** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students participate in a "Do You Understand Methods and Parameters" output exercise. **The Rational Class, Part I Lab Assignment** Students work with the **Rational** class on two sequential lab assignments. The first lab assignment manipulates integers by reducing them and displaying them in original/rational format, decimal format and reduced/rational format. **The Rational Class, Part II Lab Assignment** The second lab assignment expands the **Rational** class by performing addition, subtraction, multiplication and division of the fractions stored by the class.
Teaching Strategies, Best Practices	With the official start of the ***Focus on OOP*** chapters students need to get serious about program writing. All OOP Program Design revolves around *program reliability*. The first OOP feature of *encapsulation* insure reliability by encapsulating all the data and the methods that access the data inside the same container. All members of the class container are declared to insure that only proper access is allowed. The **Card** class is used to teach encapsulation. It is not yet identified as another AP Lab. At this stage the class is so far removed from the **Elevens** card game, that it may not register properly. When students learn about arrays, the **Card** class is revisited as it is used to create an array of **Card** objects.

	The **Rational** class lab assignment is rather dull, certainly compared to graphics programs, but it a classic program that does a very good job using encapsulation.
Curriculum Requirements	CR#1, CR#4

Unit X	**Focus on OOP, Inheritance & Composition - 2 Weeks**
General Objectives	Students continue to learn the reliability goal of OOP. Classes that already exist, that have been tested thoroughly, are reliable. Such classes can be re-used with new programs to create new functionality with greater efficiency and greater accuracy. Students learn to re-use existing classes in an is-a inheritance relationship and a has-a composition relationship.
Topics, Sub-Topics, Details to be Covered	• Re-using existing classes with *has-a composition* using the **Tree** class o Creating a **Point** class to store graphics coordinates o Create a **Trunk** class, which has-a **Point** object member o Create a **Leaves** class, which has-a **Point** object member o Create a **Tree** class, which has-a **Trunk** object and has-a **Leaves** object • Re-using existing classes with is-a inheritance using the **Tree** superclass o Use inheritance with a **PineTree** class, which is-a **Tree**, so **PineTree extends Tree** o Use inheritance with a **XmasTree** class, which is-a **PineTree**, so **XmasTree extends PineTree** • Using **super** to pass information to higher class levels o Implied use of the **super** method with no-parameter superclass constructors o Using the **super** method to pass information to a superclass constructor o Using the **super** method to pass information to multiple superclass levels o Using **super** to execute a superclass method and then continue with the subclass method • Understanding the **Object** class o Original definitions of the **toString** and **equals** methods in the **Object** class o Re-defining the **toString** method o Using the **toString** method to display attribute **state** for debugging purposes o Re-defining the **equals** method
Resources, Evaluations	**Exposure Java, Chapter 10, Focus on OOP, Interaction with Inheritance & Composition** **D&S Marketing APCS Prep, Chapter 5, OOP, Inheritance** **D&S Marketing APCS Prep, Chapter 10, OOP, Composition** 2 Reading Quizzes, 2 Free-Response Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	**Open-Ended Two-Person Graphics Picture Lab Assignment** Students complete an open-ended graphics program with a partner. They must design a program of multiple classes that creates a graphics picture in such a manner that multiple relationships of inheritance and composition are shown. A simple example would be that a *JackOLantern is-a Pumpkin and it has-a face*. All classes are placed in separate files and a special "tester class" is used to make sure the program works correctly.
Teaching Strategies, Best	The *JackOLantern* example, done correctly would receive a low score, since it only demonstrates one example of composition and one example of inheritance. Students are expected to have multiple classes with two or more examples of composition and inheritance. This unit shows that *composition* is taught first. Sequence of topics is often debated and ultimately it is the teacher's choice. It can be effective both ways. The issue is to use *tested existing classes*. When large

Practices	programs are designed, it is often the creation of simple classes, such as the **Card** class that is done first. With the initial class there is yet no relationship and the initial class gets tested. Later the **Card** class can become part of an array that is kept in a **Deck** class. That is composition. A hospital class may start with a **Person** class that store and manage all personal information. This class can then be extended with a **Patient** class and then used inside a **Hospital** class. The sequence is not the true importance. The importance is that both *is-a* and *has-a* relationships are thoroughly taught and used in lab assignments.
Curriculum Requirements	CR#1, CR#4

Unit XI	**One-Dimensional Static Arrays - 2 Weeks**
General Objectives	Students will learn the difference between classes and data structures. They will use the one-dimensional static array to store elements of the same type. Students will continue to understand elements of program design by creating a **Deck** class that contains a one-dimensional static array of **Card** objects and the methods that manipulate the **cards** array.
Topics, Sub-Topics, Details to be Covered	• Introduction to data structures • One-dimensional array declaration with **new** and with an initializer list • 1D array access with the index. square bracket operators • Displaying array elements with a loop structure and the **length** field • Understanding runtime **ArrayIndexOutOfBoundsException** errors • Creating random sentences with multiple arrays • Displaying array elements with the **for..each** loop • Understanding the immediate, shallow value of a static array, which is a reference • Understanding the reference of the contiguous block where the deeper values are stored • Making *shallow* copies and *deep* copies of an array object • Consequences of *aliasing* when two objects reference the same deep contiguous block of values • Introduces the AP Elevens Lab o Repeat Explain ethical implications of using and altering the AP Labs o Identify Michael Glancy of University of California, Berkeley, o Robert Glen Martin of the School for the Talented and Gifted in Dallas o and Judith Hromcik of the School for the Talented and Gifted in Dallas o as the AP Elevens Program Developers o Review of the **Card** class o Creating a static array of **Card** objects o Creating a **Deck** class, which has-a **cards** array and accessing methods
Resources, Evaluations	**Exposure Java, Chapter 11, One-Dimensional Static Arrays** **College Board AP Elevens Lab Materials** **D&S Marketing APCS Prep, Chapter 6, Static One-Dimensional Arrays** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students participate in a classroom exercise where a sorted sequence of students is "shuffled" using random swapping and random inserting. Students participate in an output class exercise. **The Sieve of Eratosthenes Prime Number Lab Assignment** Students complete a program that creates prime numbers. This assignment uses an array of consecutive integers that use the "Sieve of Eratosthenes" algorithm to compute prime numbers.

	The Shuffle The Deck With A Static Array Lab Assignment Students complete a second lab assignment that continues with the AP Elevens Lab. They need to take the provided **Card** class and use it to make a one-dimensional, static array of **Card** objects in a **Deck** class. Students need to create a constructor to assign all the cards, two shuffle methods and re-define the **toString** method for the **Deck** class to display all the cards in the deck.
Teaching Strategies, Best Practices	Whenever possible use students physically in the classroom. For the shuffle exercise line them up in alphabetical order. Then have them decide they can be shuffled. Conclude by showing the "swap" and "insert" approach, if they did not decided that themselves. This chapter continues with the Elevens AP Lab. The actual card game is still not apparent, but students are learning the tools of the future card game over a series of chapters. In the process of slowly developing the AP Lab, it is used as a teaching tool for computer science concepts.
Curriculum Requirements	CR#1, CR#2a, CR#2b
Unit XII	**One-Dimensional Dynamic Arrays with ArrayList - 1.5 Weeks**
General Objectives	Students learn the difference between static and dynamic arrays. They will use the one-dimensional dynamic **ArrayList** class to store elements of the same type. Students will continue to understand elements of program design by creating a **Deck** class that contains a one-dimensional dynamic array of **Card** objects and the methods that manipulate the **cards** array.
Topics, Sub-Topics, Details to be Covered	• 1D array declaration with **ArrayList** • Using **ArrayList** methods o Placing new array elements to the end with the one-parameter **add** method o Placing new array elements to an index location with the overloaded **add** method o Determining the number of elements in an array with the **size** method o Understanding runtime **IndexOutOfBoundsException** error o Accessing array elements for reading with the **get** method o Accessing array elements for writing with the **set** method o Reduce the array size by deleting array elements with the **remove** method • Using the wrapper classes, like **Integer**, **Double** and **Boolean** to store primitive data values • Creating **ArrayList** objects with generics using **ArrayList<String> names = new ArrayList<String>();** • Displaying **ArrayList** objects with the **for..each** loop structure • Creating a **Deck** class with a dynamic array • Review the **Magpie** program for the lab assignment
Resources, Evaluations	**Exposure Java, Chapter 12, One-Dimensional Dynamic Arrays with ArrayList** **College Board AP Elevens Lab Materials** **College Board AP Magpie Chatbot Lab Materials** **D&S Marketing APCS Prep, Chapter 9, Dynamic Arrays** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students participate in a classroom exercise where a sorted sequence of students is "shuffled" using random swapping and random inserting. **Improving the Magpie Chatbot Program Lab Assignment** Students make two changes to the earlier version of the **Magpie Chatbot** program. First, they need to alter the family response from a single response to multiple random responses that are stored in a static one-dimensional array. Second they need to create additional general random responses and store them in a dynamic array. **The Shuffle The Deck With A Dynamic Array Lab Assignment** Students complete a second lab assignment that continues with the AP Elevens Lab. They need to take

	the provided **Card** class and use it to make a one-dimensional, dynamic array of **Card** objects in a **Deck** class. Students need to create a constructor to assign all the cards, two shuffle methods and re-define the **toString** method for the **Deck** class to display all the cards in the deck.
Teaching Strategies, Best Practices	A common teacher mistake is to think that "covering a topic" means understanding for their students. Repetition is beneficial in computer science as it is in sports. The same algorithmic logic is repeated in this unit's lab assignment as the last chapter, but there are key differences in how dynamic and static arrays handle the same process. This will increase their knowledge and comfort level. In this chapter students are first introduced to generics in a class. They have not yet learned how to create a generics class, which comes in the next chapter. In this chapter it is strictly a matter of using generics.
Curriculum Requirements	CR#1, CR#2a, CR#2b, CR#5

Unit XIII	**Interfaces, Concrete Classes and Abstract Classes - 1 Week**
General Objectives	Students will learn the difference between an interface, abstract class and concrete class and know their proper uses. Students will follow a High School program case study that uses interfaces, concrete classes and abstract classes to help appreciate the concepts.
Topics, Sub-Topics, Details to be Covered	• Explain the difference between abstract and concrete • The use for an abstract interface and the benefits of **information hiding** • Unique interface syntax requirements and the optional abstract • Implementing interfaces with concrete classes • How to use a field in an interface • Declaring an abstract class • Using an abstract class to implement the common methods • Using an abstract class as an adapter class by implementing all methods without functionality • Using constructors in an abstract class • How to create your own generics classes o Setting up a generic interface o Using a generic subinterface o Writing a generic concrete class o Understand where the use the generic **<E>** or **<T>** variable • The Java **Collection** classes o The **Collection** interface o The **List** and **Set** subinterfaces o The **ArrayList, LinkedList, HashSet** and **Treeset** concrete classes
Resources, Evaluations	**Exposure Java, Chapter 13, Interfaces, Concrete Classes and Abstract Classes** **D&S Marketing APCS Prep, Chapter 12, Interfaces and Abstract Classes** 2 Reading Quizzes, 2 Free-Response Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students are divided into teams and discuss how to put a man on the moon as President Kennedy stated back in 1961. The class exercise is expected to focus on the abstract nature of the assignment without any concern of the implementation of all the required tasks to accomplish the mission. **Open-Ended Two-Person Interface, Abstract Class & Concrete Class Lab Assignment** Students complete an open-ended lab assignment with a partner. For this assignment they need to take the **HighSchool** case study as a model and create their own program that includes practical examples of using an interface, an abstract class to implement common methods and concrete classes.
Teaching	This unit needs to start in the context of everyday life. The example used is the flight to the Moon. In the

Strategies, Best Practices	initial stage of the mission there was no concern with implementation details. This helps students understand the benefits of talking about ideas without concern how to implement the ideas, which is the essence of abstraction. This chapter puts emphasis on the proper use of an abstract class. Since it has the unique properties of implementing some abstract methods, but not all, an ideal use is as a bridge class. In the **HighSchool** case study there are 12 abstract methods in the interface. Four concrete classes are needed for each grade level. It is also found that 7 methods are identical for each grade level. The abstract class is ideal to implement these 7 methods and then allow the concrete classes to take care of the remaining 5. The details of the Collection classes, like **LinkedList, HashSet** and **TreeSet** are not covered. The Java **Collection** classes are presented to teach the **List** subinterface and the **ArrayList** concrete class in context. It also helps in explaining why the **add** method of **ArrayList** is a boolean return method.
Curriculum Requirements	CR#1, CR#2b, CR#4, CR#5

Unit XIV	**Focus on OOP, Polymorphism - 1.5 Weeks**
General Objectives	Students will learn that polymorphism requires an umbrella interface or an umbrella superclass. Students will learn how to take a method originally declared in the umbrella interface or umbrella class and re-define this method for implementing or extending classes, in such a way that each class is responsible for its own actions.
Topics, Sub-Topics, Details to be Covered	• Examples of overloaded operators and overloaded methods • Inefficient use of classes and methods with unnecessary selection structures • Understand the need for an umbrella interface or an umbrella superclass • Implementing polymorphism with an interface • Implementing polymorphism with an abstract class • Implementing polymorphism with an concrete superclass • When is an interface or a superclass better for using polymorphism? • Studying the **RailCar** graphics polymorphism case study o Designing the *umbrella* **RailCar** class with the polymorphic **drawCar** method o Design the **Locomotive** subclass and re-define method **drawCar** o Use **super** to call the superclass **drawCar** method, before adding additional statements o Design the **PassengerCar** subclass and re-define method **drawCar** o Design the **PassengerCar** subclass and re-define method **drawCar** o Design the **PassengerCar** subclass and re-define method **drawCar** o Display the entire train using polymorphism with a **drawTrain** method o Display the entire train using polymorphism with an **ArrayList** object
Resources, Evaluations	**Exposure Java, Chapter 14, Focus on OOP, Polymorphism** **D&S Marketing APCS Prep, Chapter 11, Polymorphism** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	**The Polymorphic Geometric Shapes Lab Assignment** Students write a program that displays a Square, a Triangle, an Octagon and a Circle graphically. Students are provided with an interface and its abstract methods. The lab assignment requires the writing of four concrete classes that implement the polymorphic **drawShape** method . **The Polymorphic Train Lab Assignment** Students are provided with the files for the **Railcar** class, **FreightCar** class, **PassengerCar** class, **Locomotive** class and **Caboose** class. The provided program is poorly designed. Students need to create a **Train** class that encapsulates objects of all the classes, provides proper class interaction and displays the entire train using polymorphism.

Teaching Strategies, Best Practices	This chapter starts with a **Language** interface. This is followed with concrete classes of four languages and methods of their greetings. It is a simple example of using polymorphism and in this case with an interface. Later in the train case study it starts with a **RailCar** and a **drawCar** method that draws the basic outline of every type car in the program. Subclasses of **RailCar** re-define method **drawCar**, but not from scratch. First **super.drawCar** is called to starts the drawing and then the subclass **drawCar** finishes by adding the unique features of the subclass. The two sets of programs are intentionally selected to show the difference in using polymorphism with an interface and a superclass. The same set of train classes shown in the chapter are used for the second lab assignment. Using these classes with proper program design is part of the lab assignment. Students are also expected to create an overloaded **addCar** method that inserts a new car at a specified location. Considerable changes must be made to every class to accomplish this requirement. This is quite a challenging program and will involve knowledge of encapsulation, inheritance, composition and polymorphism all in one assignment.
Curriculum Requirements	CR#1, CR#2b, CR#4

Unit XV	**Two-Dimensional Arrays with the Picture AP Lab - 1.5 Weeks**
General Objectives	Students will learn how to declare and use a static two-dimensional array sand understand that a two-dimensional array is a one-dimensional array of arrays. Students will also learn how to use the **ArrayList** class to create a two-dimensional array. Students will receive practice with two-dimensional arrays by manipulating picture images with the AP Picture Lab exercises.
Topics, Sub-Topics, Details to be Covered	• Declaring a two-dimensional static array • Understand row-major order of 2D arrays • Using an initializer list with 2D arrays • Using nested loops to access 2D array elements • Proper use of the **length** field with 2D arrays • Evidence that a 2D array is in fact a 1D array of 1D arrays • Using nested **for..each** loops to display a 2D array • Creating a 2D array with the **ArrayList class** • Using the AP Picture Lab for 2D array processing in the context of photo image manipulations o Repeat ethical and social implications of using the AP Labs and other software o Identify Barbara Ericson of the Georgia Institute of Technology as the AP Picture Lab Program Developer o Class Exercise A1: Introduction to digital pictures and RGB colors o Class Exercise A2: Picking with **Color Chooser** o Class Exercise A3: Exploring a picture with **PictureExplorer** tool o Class Exercise A5: Modifying a picture with methods that alter picture pixels o Class Exercise A6: Mirroring pictures by looping through 2D array of pixels o Class Exercise A7: Mirroring through part of an image o Class Exercise A8: Creating a Collage
Resources, Evaluations	**Exposure Java, Chapter XV, Two-Dimensional Arrays** **College Board AP Picture Lab Materials** **D&S Marketing APCS Prep, Chapter 2, Two-Dimensional Static Arrays** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Students participate with 7 AP Picture Lab exercises. Students do several class exercises creating different *Latin Squares* and *Magic Squares* on paper. **The Latin Square Two-Dimensional Array Lab Assignment**

	Students write a program that manipulates the numbers in a two-dimensional array to form a *Latin Square*. Students also are required to re-define the **toString** method to create a matrix display and re-define the **equals** method to compare two Latin Squares for equality. **The Magic Square Two-Dimensional Array Lab Assignment** Students write a program that manipulates the numbers in a two-dimensional array to form an *Odd Magic Square*. Students also are required to again re-define the **toString** method to create a matrix display.
Teaching Strategies, Best Practices	Students will rapidly learn that lab assignments are not a matter of creating Java program code. First and foremost is step 1: *Understand the Problem*. Class exercises on paper are very beneficial to insure that the problem is understood and the steps necessary to solve the problem are known. Students may struggle with the proper use of **length** until they truly understand the concept that a 2D Array is a 1D Array of arrays. It helps to demonstrate the program that displays a "triangular ragged" array. This is not tested on the AP Exam, but it really help to clarify the proper use of **length**.
Curriculum Requirements	CR#1, CR#2a, CR#2b, CR#7

Unit XVI	**Focus on OOP, Object Oriented Design - 2 Weeks**
General Objectives	Students will learn to design programs in the Object Oriented Programming style. They will learn to start with fundamental unit classes that are thoroughly tested and can then be used with other classes using composition and inheritance for reliable class interaction. Students also learn how abstraction is a good tool to start a large program before any implementations are considered.
Topics, Sub-Topics, Details to be Covered	• Timeless Design Issues o Read and understand a problem description, purpose and goals o Self-documenting identifiers and program comments o Use of **JavaDoc** with comment tags **@param**, **@return** and **@Override** o *Functional Decomposition* with *modular programming* • Testing and Debugging o Differentiate between *compile*, *logic* and *runtime* errors, including **exception**: o **Arithmetic, NullPointer, IndexOutOfBounds, ArrayIndexOutOfBounds & IllegalArgument** o Debugging with a *debugger*, output statements and hand-tracing code o Test classes and libraries in isolation and then perform integration testing o Identify boundary cases and generate appropriate test data • Object Oriented Design • Identify appropriate classes • Create and test the unit class • Use existing classes to create new classes • Class design and implementation a class hierarchy using inheritance and composition • Method design with assertions about pre-conditions and post-conditions • Testing and Debugging • Object Oriented Hands-On o Observes how Object Oriented Design is used in the AP Elevens Lab o Do exercises to see how the **Deck** class is used in the Elevens Lab o Using the **Deck** in a different card game selecting appropriate algorithms
Resources, Evaluations	**Exposure Java, Chapter 16, Focus on OOP, Object Oriented Design** **College Board AP Elevens Lab Materials** **D&S Marketing APCS Prep, Chapter 13, Program Design**

	2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab Assignments	Multiple Class exercises showing how the **Deck** class can be used in the AP Elevens card game. **The Alter the AP Elevens Lab Assignment** Students take the existing AP Elevens Lab and alter it functionality by changes some method, classes and also creating new necessary classes. **The Dutch Solitaire Assignment** Students work in groups to the take **Deck** class and design a new card game called "Dutch Solitaire." Students need to design new classes, select proper data structures and algorithms with proper class interaction to create a completely different game for the "Elevens" card game.
Teaching Strategies, Best Practices	Teaching program design can easily turn into very boring, dry class lectures. Students have already been doing Object Oriented Design of quite a number of chapters. It may not have been a focused unit, but students have already done many program design features. The AP Elevens Lab has done much work for students and has provided excellent example of proper program design. Students now can look at a second program where less work is done for them and they need to use the lessons already learned to complete the assignment. The completion of the "Dutch Solitaire" program is a combination of class exercises and group work.
Curriculum Requirements	CR#1. CR#2a, CR#2b, CR#3, CR#4
Unit XVII	**Recursion - 1 Week**
General Objectives	Students will learn how the computer's internal stack can facilitate a form of iteration that is called recursion. Students will examine the significance of the base case in recursion and how to handle double recursive calls in a single statement. Students will also learn when recursion can actually be simpler for implementing certain algorithms over non-recursive control structures.
Topics, Sub-Topics, Details to be Covered	• Recursion definition • Recursion requires an exit with a "base" case • Recursion fundamental rules • Understanding the difference between pre-recursion and post-recursion (tail-recursion) • Tracing through existing recursive methods o Recursive void method examples o Recursive return method examples o Evaluate recursive methods with two recursive calls in one statement o Use the "bottom-up" evaluate technique o Show the efficiency penalty of double recursive calls with *Fibonacci* • Manipulate parameters of recursive methods • Examples of multiple recursive statements in one method o The "Tower of Hanoi" problem o The "Alter Black/White" Grid" problem • When is recursion preferred over iterative control structures?
Resources, Evaluations	**Exposure Java, Chapter 17, Recursion** **D&S Marketing APCS Prep, Chapter 16, Recursion** 2 Reading Quizzes, 2 Free-Response Quizzes, 2 Lab Assignments and 1 Multiple-Choice Chapter Test. Students complete the ***Convert to Recursion*** lab assignment. Student complete the ***Square Fractal*** lab assignment
	Students participate with the solve "Tower of Hanoi" class exercise Students do many class exercises evaluating a wide variety of recursive methods.

Class Tasks, Lab Assignments	**The Convert Iteration to Recursion Lab Assignment** Students are provided with a fully functional program that uses many methods. Every one of the method uses iterative loop structures. Students have to alter the program by rewriting all the methods that use loop structures into recursive solutions. **The Recursive Fractal Lab Assignment** Fractals are excellent examples of recursive displays where each feature duplicates multiple, smaller versions of itself. Students are provided with images of the "Square Fractal" and the "Sierpinski" Fractal. They need to write program that will recursive generate those fractal images.
Teaching Strategies, Best Practices	Students must start by understanding the difference between *pre-recursion* and *post-recursion.* Two program examples are shown with counting methods. When the method first display the number and then makes a recursive call, the output is in sequence from smaller to larger. When the method starts with a recursive call, unfinished business and values are stored on the stack, and this is completed in LIFO sequence resulting in reverse output. Students struggle with output of methods that have two recursive calls in one program statement. This process becomes simpler by starting with the base-case value and then working in reverse up to the start to computer the returned value. The Fibonacci sequence is a good example of such a problem.
Curriculum Requirements	CR#1, CR#2a

Unit XVIII	**Algorithms and Informal Algorithmic Analysis - 1.5 Weeks**
General Objectives	Students will learn the fundamental algorithms, used manipulating array location, for sorting and searching. Students will examine how objects with many data attributes need to be sorted based on a selected attribute. Students will also examine the processing of large amounts of data and observe informally the difference in data processing as a consequence of using different algorithms.
Topics, Sub-Topics, Details to be Covered	• Array traversal, insertion and deletion algorithms • Search Algorithms o The Linear Search o The iterative Binary Search o The recursive Binary Search • Sorting Algorithms o The Bubble Sort o The Selection Sort o The Insertion Sort o The Merge Sort with recursive helper method • Sorting an array of by different keys • Select appropriate data and algorithms for required data processing needs • Informal comparisons of algorithms o Students learn how to use the user-defined **Time** class to measure algorithm processing o Measuring algorithms with random data o Measuring algorithms with data ordered in ascending order o Measuring algorithms with data ordered in descending order • Appreciate the importance of **Information Hiding** by using a program that imports data
Resources, Evaluations	**Exposure Java, Chapter 18, Algorithms and Informal Algorithmic Analysis** **D&S Marketing APCS Prep, Chapter 15, Algorithms** 2 Reading Quizzes, 2 Free-Response Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.

Class Tasks, Lab Assignments	Students will learn how to use a special **Time** class. This class enables precise measurement of algorithm duration. Students will use this class on many algorithms to perform informal algorithmic analysis. **The King High School Data Processing Lab Assignment** Students will work with a team member to complete a large data processing program. Students are provided with a program that already has a functional class and methods to import a large file student data. Students can work with this program and enhance the program, even if they have no file structure knowledge. This is a good example of **information hiding**. Students need to complete all the methods and then test the program with the provided data file for such as *Selecting the Valedictorian, Compute class standing, Display a required student record, Print a student list alphabetically, Print a student rank list by GPA.* The file handling is done for the students, but students must select appropriate algorithms and data structures to make the program functional.
Teaching Strategies, Best Practices	This is one of the few second-semester lab assignments where program design is not an issue. The complete program is designed. Students need to implement multiple methods multiple ways and tests each one of the implementation to make conclusions about the effectiveness of the different algorithms. The lab assignment is preceded by students using a program that generates a large number of numbers in random, ordered and reverse order to test algorithms. The **Time** class is very accurate and uses **System.nanoTime** to measure the elapsed time of algorithm processing. It will really surprise students when they start to measure the quadratic sorts first and then go to Merge Sort with a large quantity of random numbers. The differences are very dramatic.
Curriculum Requirements	CR#1, CR#2a, CR#2b, CR#3, CR#4

Unit XIX	**Preparing for the AP Computer Science Exam - 3 Weeks**
General Objectives	Students will understand thoroughly what to expect on the AP Computer Science Examination and in particular focus a group of multiple-choice question in various topics that include components easily overlooked by many students. Students will learn specific strategies for time management and completing free-response question
Topics, Sub-Topics, Details to be Covered	• The AP Computer Science Examination format o The multiple-choice section o The free-response section o Time issues for both section of the exam • Test taking strategies o Be aware of snooker questions o Multiple snooker question examples on different topics o Dealing with time-consuming questions o Does guessing hurt AP Exam scores o How free-response questions are scored o Using methods created in Question 4(a) - right or wrong - in Question 4(b) • Take AP Sample Exam I • Take AP Sample Exam II • Take AP Sample Exam III
Resources, Evaluations	**Exposure Java, Chapter 19, Preparing for the AP Computer Science Examination** **D&S Marketing APCS Prep, Chapter 18, Snooker Questions** **D&S Marketing APCS Prep, Chapter 20, Sample APCS Exam I** **D&S Marketing APCS Prep, Chapter 20, Sample APCS Exam II** **D&S Marketing APCS Prep, Chapter 20, Sample APCS Exam III** Three sample AP Computer Science Examination

Class Tasks, Lab Assignments	Over a number of review periods students take multiple-choice snooker-style questions to learn to pay attention to details and learn techniques for certain topics. Sample AP exams are given partially at home and partially in class depending on available time.
Teaching Strategies, Best Practices	A standard review where teachers drone on from one subject to the next is deadly for the teacher, worse for the student and may lower students' scores dramatically. Students need to realize that a given topic is worth reviewing. The snooker questions are ideal. Most snooker questions are missed by 75% of the students. Questions are missed because: 1) The topic is misunderstood. 2) The student made an silly mistake. 3) The student was snookered. Review on the topic of questions that most students miss is very effective. If possible students come on a Saturday to take a complete sample AP Exam. This simulates the real conditions the best. If this is not possible. The sample exams are broken up between home and school and sometimes an entire exam may be taken at home.
Curriculum Requirements	CR#1, CR#2a, CR#2b, CR#3, CR#4, CR#5, CR#7

Unit XX	**Post AP Exam I, Sequential Files**
General Objectives	Students learn how to read data from an external database and how to write data to an external database. The file handling is done with sequential files and students learn how to create new files, update existing files and delete existing files.
Topics, Sub-Topics, Details to be Covered	• Different types of files o Sequential access files o Random access files • Using the File class o Determine file existence o Determine file properties • File IOException handling and throwing file handling exceptions • Input text files with *BufferedReader* and *FileReader* o Wrapper class concept with anonymous objects o File buffer concept • Output text files with *BufferedWriter* and *FileWriter* • Handling text files with integer and double values o Storing numerical values in a text file o Converting numerical strings into **int** and **double** values • A note about the *Scanner* class and file handling
Resources, Evaluations	**Exposure Java, Chapter 20, Sequential Files** 2 Reading Quizzes, 1 Lab Assignment and 1 Multiple-Choice Chapter Test.
Class Tasks, Lab	**The Student Records Lab Assignment** Students are provided with the same student database used for the earlier algorithm chapter. This time students need to complete the "hidden" part of the program that was used, without knowledge in the

Assignments	previous lab assignment.
Teaching Strategies, Best Practices	This is an optional topic that is optional, because it is not tested. However, it is an important topic for student who will continue in computer science at the college-level and receive credit for the first semester.
Curriculum Requirements	Not Applicable; After AP Exam

Unit XXI	**Post AP Exam II, Advanced Graphics Project**
General Objectives	Students learn advanced graphics techniques of mouse handling, animation with double-buffering and mathematics to create a team graphics project.
Topics, Sub-Topics, Details to be Covered	• Review of basic *awt* graphics • Methods *drawLine*, *drawRect*, *fillRect*, *drawOval*, *fillOval*, *drawArc*, • *fillArc*, *setColor*, *drawPolygon*, *fillPolygon* • Controlling graphics text with *drawString* and *setFont* • Mathematics and graphics • Drawing circles and regular polygons with *Math.cos* and *Math.sin* • Using x and y coordinate arrays with the *Polygon* class • Using mouse interaction with graphics • The event method concept • Methods *mouseDown*, *mouseEnter*, *mouseExit*, *mouseMove*, • *mouseUp*, *mouseDrag* • Creating graphics animation • Fundamental draw-and-erase animation • Virtual memory and video buffering o Reserving virtual memory with *Image* and *getGraphics* o Page flipping with *drawImage* • Improving animation flicker with the *update* method
Resources, Evaluations	**Exposure Java, Chapter 21, Advanced Graphics** 1 Large Team Graphics Project

Class Tasks, Lab Assignments	**The End-Of-Semester Advanced Graphics Team Project** Students work with a team on this large graphics assignment. There are no quizzes at this time. There is no homework or chapter tests. Student devote all their time to this project creating some graphics project like a video game.
Teaching Strategies, Best Practices	Students enjoy working on this graphics project and information about the project is already distributed at the first class of the second semester. Students work on this assignment at the conclusion of chapter tests or lab assignment during the4 school year. For this project up to four students may work together.
Curriculum Requirements	Not Applicable; After AP Exam