Texts

Each unit includes an Internet link with additional information for enhancement and a different perspective. The provided links in this syllabus may be altered as information changes or new links become available. Frequently, links are provided to Sun’s Java tutorial web site.

Syllabus at a Glance

<table>
<thead>
<tr>
<th>General Topics</th>
<th>Sem/Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ch 1   Introduction to Computer Science</td>
<td>Sem1 1</td>
</tr>
<tr>
<td>Ch 2   Introduction to Java Programming</td>
<td>Sem1 2-3</td>
</tr>
<tr>
<td>Ch 3   Java Primitive Data Types and Arithmetic Operators</td>
<td>Sem1 4-5</td>
</tr>
<tr>
<td>Ch 4   OOP, a First Exposure; Using Class Methods, Parameters and Introduction to Graphics</td>
<td>Sem1 6-7</td>
</tr>
<tr>
<td>Ch 5   Control Structures</td>
<td>Sem1 8-9</td>
</tr>
<tr>
<td>Ch 6   Using Object Methods</td>
<td>Sem1 10-11</td>
</tr>
<tr>
<td>Ch 7   Creating Class Methods and Introduction to Program Design</td>
<td>Sem1 12</td>
</tr>
<tr>
<td>Ch 8   Creating Object Methods</td>
<td>Sem1 13</td>
</tr>
<tr>
<td>Ch 9   Inheritance and Composition</td>
<td>Sem1 14</td>
</tr>
<tr>
<td>Ch 10  Boolean Logic</td>
<td>Sem1 15-16</td>
</tr>
<tr>
<td>Ch 11  Control Structures with Compound Conditions</td>
<td>Sem1 17</td>
</tr>
<tr>
<td>Ch 12  Java Static Arrays and Arrays class</td>
<td>Sem1 18</td>
</tr>
<tr>
<td>Ch 13  Advanced Graphics and Animation</td>
<td>Sem2 1</td>
</tr>
<tr>
<td>Ch 14  Serious Object Oriented Programming</td>
<td>Sem2 2-3</td>
</tr>
<tr>
<td>Ch 15  Program Design; Working with Large Programs and Introduction to GridWorld Case Study</td>
<td>Sem2 4</td>
</tr>
<tr>
<td>Ch 16  String Processing and Number Systems</td>
<td>Sem2 5</td>
</tr>
<tr>
<td>Ch 17  Input/Output with Sequential Files</td>
<td>Sem2 6-7</td>
</tr>
<tr>
<td>Ch 18  Algorithms and Informal Algorithmic Analysis</td>
<td>Sem2 8-9</td>
</tr>
<tr>
<td>Ch 19  Recursion</td>
<td>Sem2 10-11</td>
</tr>
<tr>
<td>Ch 20  ArrayList Class; Redefining Methods toString and equals; Implementing compareTo; Autoboxing; Generics</td>
<td>Sem2 12</td>
</tr>
<tr>
<td>Ch 21  Interfaces, Abstract Classes and Polymorphism</td>
<td>Sem2 13</td>
</tr>
<tr>
<td>Ch 22  The GridWorld Case; Preparation for the AP Computer Science Examination</td>
<td>Sem2 14-17</td>
</tr>
</tbody>
</table>
## Correlation to AP Topic Outline - Computer Science A

### I. Object-Oriented Program Design

**A. Program design**
- 1. Read and understand a problem description, purpose and goals  
  Units 15, 22
- 2. Apply data abstraction and encapsulation  
  Units 8, 14
- 3. Read and understand class specifications and relationships among the class ("is-a," "has-a" relationships)  
  Units 9, 15, 22
- 4. Understand and implement a given class hierarchy  
  Units 9, 15, 22
- 5. Identify reusable components from existing code using classes and class libraries.  
  Units 7, 15, 16, 20, 21

**B. Class design**
- 1. Design and implement a class  
  Units 15, 22
- 2. AB only
- 3. Choose appropriate data representation and algorithms  
  Units 15, 18, 22
- 4. Apply functional decomposition  
  Units 7, 15, 22
- 5. Extend a given class using inheritance  
  Units 9, 15, 22

### II. Program Implementation

**A. Implementation technique**
- 1. Methodology
  - a. Object-oriented development  
    Units 4, 8, 15, 22
  - b. Top-down development  
    Units 4, 15, 22
  - c. Encapsulation and information hiding  
    Units 4, 8, 14, 15, 22
  - d. Procedural abstraction  
    Units 4, 6, 15, 22

**B. Programming constructs**
- 1. Primitive types vs. objects  
  Units 3, 8
- 2. Declaration
  - a. Constant declarations  
    3
  - b. Variable declarations  
    3
  - c. Class declarations  
    8, 9, 14, 15, 22
  - d. Interface declarations  
    21, 22
  - e. Method declarations  
    7, 8, 9, 14, 22
  - f. Parameter declarations  
    7, 8, 22
- 3. Console output (System.out.print/println)  
  2, 3
- 4. Control
  - a. Methods  
    4, 6, 8
  - b. Sequential  
    5, 11, 18
  - c. Conditional  
    5, 11, 18
  - d. Iteration  
    5, 11, 18
  - e. Recursion  
    11, 19

**C. Java library classes (included in the A-level Java Subset)**  
  16, 20, 21
### III. Program Analysis

| A. Testing | 1. Test classes and libraries in isolation | 15, 22 |
| 2. Identify boundary cases and generate appropriate test data | 15, 18 |
| 3. Perform integration testing | 15, 18 |
| B. Debugging | 1. Categorize errors: compile-time, run-time, logic | 11, 15 |
| 2. Identify and correct errors | 11, 15 |
| 3. Employ techniques such as using a debugger, adding extra output statements, or hand-tracing code | 15 |
| C. Understand and modify existing code | 22 |
| D. Extend existing code using inheritance | 9, 22 |
| E. Understand error handling | 15, 17 |
| F. Reason about programs | 1. Pre-conditions and post-conditions | 15, 22 |
| 2. Assertions | 15, 22 |
| G. Analysis of algorithms | 1. Informal comparisons of running time | 18 |
| 2. Exact calculations of statement execution counts | 18 |
| H. Numerical representations and limits | 1. Representation of numbers in different bases | 1, 16 |
| 2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error) | 3 |

### IV. Standard Data Structures

| A. Simple data types (int, boolean, double) | 3, 10, 23 |
| B. Classes | 8, 9, 25 |
| C. One-dimensional arrays | 12, 20, 26 |

### V. Standard Algorithms

| A. Operations on A-level data | 1. Traversals | 18 |
| 2. Insertions | 18 |
| 3. Deletions | 18 |
| B. Searching | 1. Sequential | 18 |
| 2. Binary | 18 |
| C. Sorting | 1. Selection | 18 |
| 2. Insertion | 18 |
| 3. Mergesort | 18 |
## VI. Computing in Context

<table>
<thead>
<tr>
<th>A. Major hardware components</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Primary and secondary memory</td>
<td>1</td>
</tr>
<tr>
<td>2. Processors</td>
<td>1</td>
</tr>
<tr>
<td>3. Peripherals</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B. System software</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Language translators/compilers</td>
<td>2</td>
</tr>
<tr>
<td>2. Virtual machines</td>
<td>2</td>
</tr>
<tr>
<td>3. Operating systems</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C. Types of systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Single-user systems</td>
<td>1</td>
</tr>
<tr>
<td>2. Networks</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D. Responsible use of computer systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. System reliability</td>
<td>1</td>
</tr>
<tr>
<td>2. Privacy</td>
<td>2</td>
</tr>
<tr>
<td>3. Legal issues and intellectual property</td>
<td>2</td>
</tr>
<tr>
<td>4. Social and ethical ramifications of computer use</td>
<td>2</td>
</tr>
<tr>
<td>Unit #</td>
<td>Starting Week</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>1</td>
<td>SEM 1</td>
</tr>
</tbody>
</table>
Exposure Java, Chapter 2
http://www.jcreator.com/
http://java.sun.com/javase/downloads/index.jsp

**Introduction to Programming in Java**

Getting started with Java
Platform dependent and platform independent languages
The Java bytecode concept
Java application programs and applet programs
Downloading and installing Java software
Downloading and Installing an Integrated Development Environment (IDE)
Responsible use of computer software, hardware
  - Maintaining system reliability
    - Hardware care
    - Protection against surges and power outs
    - Backing up data
    - Protection against viruses and identity theft
  - Protecting privacy concerns
    - Intellectual property, copyright issues, shareware, freeware
    - Social, ethical and legal implications of using computers
Setting up the Java programming workspace
Translators (Compilers and interpreters)
The Java Virtual Machine (JVM)
Compiling and executing Java applications and applets
Java input/output issues
Fundamental program text output with `print` and `println`
Java compile errors

**Evaluations**

Objective Quizzes/Exercises

Lab Assignment  *Copy, Compile and Execute*
Copy a short, provided, application program to demonstrate compile/execute skills

M.C. Chapter Test

The main purpose of this unit is to teach students how to use the Java JDK and an IDE to write programs. The lab assignment involves literally copying a working program. The intent at this stage is to learn the mechanics of working with the software and clearly understanding the compiling and executing process. In particular, students will learn the difference between compiler and interpreter translators. Once the different translators are 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 good introduction. Students are shown how to download the Java software and JCreator. 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 purchased, the topic can expand into general issues of the ethical use of the computer.
Java Primitive Data Types

Declaring and operating with numerical simple/primitive data types
- Integer types (int, byte, short, long)
- Real number types (double, float)
- Arithmetic shortcut notations
- Limitations of finite representations
  - Memory overflow resulting in imprecision
  - Mathematical accuracy and computer accuracy
- Round-off errors and real number representations

Other data types
- Character type (char)
- Boolean type (boolean)
- String type (String)

Type casting

Declaring constants with final

Programs documentation
- Single line documentation
- Multiple line documentation

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

Evaluations

Objective Quizzes/Exercises

Lab Assignment **Inches to Miles** or **Milli-seconds to Hours**
Write a program that converts a number of inches to miles, yards, feet and inches.
Write a program that converts a number of milli-seconds to hours, minutes, seconds and milli-seconds.

M.C. Chapter Test
Exposure Java, Chapter 4
http://java.sun.com/docs/books/tutorial/java/javaOO/classvars.html

Using Methods and Parameters

A brief history of program design
OOP, a gentle first exposure
Procedural abstraction
Using the standard Math class
  Method abs, pow, sqrt, random
  Methods floor, ceil, round, max, min
Fields PI, E
Using methods of a user-defined class
Accessing standard Java classes with packages using import
Using methods with multiple parameters
Compiling and executing applet programs
Using the Graphics class of the java.awt package
  Methods drawLine, drawRect, drawOval

Evaluations

Objective Quizzes/Exercises

Lab Assignment  Cube, Sphere, Triangles
Write a program that displays a cube, sphere and triangles in an applet.

M.C. Chapter Test

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.

Students will learn all the different OOP aspects during the course, but in this chapter the focus is to learn how to call methods of existing classes. The method calling is identical for standard Java classes or user-created classes.

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 many parameters.

Another nice benefit of a graphics lab assignment is that it reinforces the concepts of Coordinate Geometry that they have learned in math classes.
Unit 5 is intentionally called Control Structures I. The intention is to introduce Object Oriented Programming early in the course. The next four chapters will be devoted to various OOP concepts. However, a fundamental understanding of control structures will make the creating of classes and methods far more interesting.

In this unit all control structures use single conditions. Unit 10 teaches students Boolean logic. Immediately following the Boolean logic chapter, students return to control structures and learn to use nested structures and compound conditions.

The optional Scanner class is introduced in this chapter to provide an easy mechanism for entering keyboard input. Testing border cases with control structures is simpler when keyboard is input available.

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.
SEM 1

Week 10 and Week 11

Exposure Java, Chapter 6

Using Object Methods

 Classes and objects
 Calling the constructor method with new
 Calling object methods
 Overloaded constructors
 Using the Random class
   Methods nextInt, nextDouble, setSeed
 Using the DecimalFormat class
   Method format
 Using the Graphics class
   The hidden new operator of the Graphics object
     Additional Graphics methods setColor, drawPolygon, fillPolygon
 Using the Polygon class
   Method addPoint
 Constructing custom Color objects
 Anonymous objects
 Displaying random graphics objects with random colors
 Using the Scanner class
   Methods nextLine, nextInt, nextDouble
 Clearing the buffer with a dummy String variable

Students are intentionally taught class methods first. With class methods the fundamental syntax of calling a method and using parameters is taught without the need to create an object with new.

Unit six teaches students to use many classes that use object methods. It is now necessary to define an object and call the Class constructor with the new operator.

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.

The Scanner class is now looked at closely with its three methods for string input, whole number input, and real number input.

Students also need to learn that bizarre things can happen when the buffer is not cleared after an input with nextInt or nextDouble.

Evaluations

Objective quizzes/exercises

Lab Assignment
Write a program that displays random lines, random squares and random ovals with random colors.

M.C. Chapter Test
Creating Class Methods

The Math class revisited
Modular Programming and user-created methods
User-created parameter methods
Actual and formal parameters
Parameter passing rules
void methods and return methods
Making a utility library class
Introduction to program design with the Payroll class case study
Program input with GUI Windows

Evaluations

Objective exercises
Objective quiz and free response quiz
Quizzes/Exercises  Rewrite The Bad Program
Lab Assignment
Take a provided, very poorly, designed program and rewrite the program with separate classes and methods.

M .C. Chapter Test

Unit seven teaches students many new concepts. Up to now, students have learned to use both class methods and object methods. However, little program design is possible until students learn how to create their own classes and methods.

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 that does not require creating constructors.

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; however, it is a good start to teach important programming concepts. As additional concepts are taught additional features of program design will be introduced.

This unit introduces the new "write-a-method" quiz style of evaluation. This style of quiz will steadily increase in quantity and help prepare students for the free response section of the AP Computer Science Examination.
### SEM 1

#### Week 13

<table>
<thead>
<tr>
<th>Exposure Java, Chapter 8</th>
<th><a href="http://java.sun.com/docs/books/tutorial/java/concepts/">http://java.sun.com/docs/books/tutorial/java/concepts/</a></th>
</tr>
</thead>
</table>

#### Creating Classes with Object Methods

- Object method syntax
- Constructor methods
- Program reliability issues like side effects
- Using encapsulation to increase reliability with **private** and **public** access
- Get return object methods
- Set modifying void object methods
- The *CardDeck* case study
- The *Bank* class revisited
- The *Cube* Case study
- Do you understand methods and parameters?

#### Evaluations

- Objective exercises
- Objective quiz and free response quiz
- The chapter finishes with 25 programs that make mistakes with using methods and parameters
- For each program students need to identify the problem and offer a solution

**Lab Assignment**  *The Rational Class*

Write a program that performs arithmetic operations with rational numbers. A rational number can be represented as a/b where a and b are integers and b is non-zero. A Rational class needs to be designed and implemented so as to perform each one of the binary operations. Additionally, the Rational class needs a `computeGCF` method that will be used to represent the final fraction in reduced format.

**M.C. Chapter Test**

---

The main focus of this chapter is for students to learn the importance of encapsulation. They have learned in a prior chapter how to break up program code and place this code in modules. They have also learned how to declare parameters and make a distinction between void methods and return methods. However, they have only worked with static methods.

In this chapter, the issue of program reliability is addressed and students learn that first, and foremost, Object Oriented Design exists for the purpose of program reliability. Unit eight teaches students the importance of class member access in a proper manner by making a distinction between private and public access.

The importance of understanding encapsulation is reinforced by three case studies. Each case study starts from a very simple program and slowly works into a complete program with properly designed encapsulation.

The first case study features a *PiggyBank* class. The second case study designs a *CardDeck* class. The second case study intentionally is presented as if the first case study was never taught or understood. This approach with repeated reinforcement using different examples has proven very effective with students who are overwhelmed by Object Oriented Programming.
## Inheritance and Composition

Class specifications and relationships
- The "is-a" inheritance relationship
- The "has-a" composition relationship

**Inheritance**
- Inheritance syntax with **extends**
- Super classes and sub classes hierarchy
- Passing information to super class constructors with **super**
- **private**, **protected**, and **public** access with inheritance
- Multi-level inheritance
- Can sub classes inherit from multiple super classes?

**Composition**
- Inheritance syntax
- Handling constructors with composition
- Multi-level or multi-nested composition
- Can classes contain multiple classes?

The **JackO'lantern** case study
The **Train** case study
The **Object** class

## Evaluations

- Objective exercises
- Objective quiz and free response quiz

Lab Assignment  **Open Ended Inheritance/Composition Graphics Program**
Write a graphics program that displays both inheritance and composition. The program is open-ended, because there is no required output shown. The program needs to be in the style of the **JackO'lantern** class which is-a **Pumpkin** and has-a **Face**.

M.C. Chapter Test

Unit nine continues the Object Oriented Program introduction with a chapter on inheritance and composition.

Students start by learning to make the distinction between inheritance and composition using many examples in real life that are "is-a" and "has-a" relationships. These are examples like a square is-a rectangle and a car has-an engine.

Considerable time is devoted to the manner in which information is passed to a constructor in multi-level designs for both inheritance and composition.

There are many composition examples and many students struggle much more with composition than inheritance. In particular, students need to understand clearly how to call and instantiate a constructor of a class that is nested inside another class.

Two graphical case studies are presented to the students. Graphics is effective for students with many topics, but inheritance/composition topics especially benefit from the visual relationships.

This chapter presents the first "open-ended" lab assignment. This is somewhat scary to students, but it is a very important technique to start getting students thinking about how programs are designed.
**Exposure Java, chapter 10**

http://en.wikipedia.org/wiki/Boolean_algebra

---

**SEM 1**

**Week 15 and Week 16**

**Boolean Logic**

- A brief history of Boolean Algebra
  - George Boole
  - The birth of Boolean Algebra
  - The significance of Boolean Algebra to computers
- Boolean statements
- Boolean operators and **(&&)**, or **(||)** and not **(!)**
- Truth tables
- Laws of Boolean Algebra
  - Common Boolean Algebra laws
  - Special focus on DeMorgan's law
- Venn diagrams and Boolean Algebra
  - Representing and **as intersection**
  - Representing or **as union**
- Sample Boolean problems
- The **boolean** data type
- Boolean logic exercises

---

**Evaluations**

- Objective Quizzes/Exercises
- Lab Assignment
  - This a mathematical chapter designed to help students design programs that use compound conditions in control structures. There is no lab assignment.
- M.C. Chapter Test

---

Most computer science text books include Boolean logic in chapters where conditional statements are introduced.

An Educational Testing Service (ETS) study performed in the early Nineties showed that students who performed quite poorly on Boolean Algebra questions tended to perform poorly overall on the APCS Examination.

This chapter is intentionally placed between the Control Structures I and the later Control Structures II chapters. Students can learn about Boolean logic without being concerned with writing any type of program. With the exception of the few program examples that demonstrate the **boolean** type, the chapter is completely language independent.

The laws of Boolean Algebra are not presented with the intention to learn the name of each law, with the exception of DeMorgan's Law. Each law can be presented in an investigative, intuitive manner. Students can apply the proper Boolean logic to a problem without actually knowing the name of the law that is being used.

The chapter finishes with a set of Boolean Logic exercises to test understanding of the chapter before students take a chapter test.
### Control Structures II

The for loop revisited  
Which is the best loop to use?  
Nested selection structures  
Nested looping structures  
Compound conditions  
Program input protection  
  Common logic errors  
  Using DeMorgan's Law properly  
Short-circuiting compound conditions  
Recursion, a sneak preview  
Output exercises

### Evaluations

Objective exercises  
Objective quiz and free response quiz

Lab Assignment  
**Watch What You Borrow Program**  
Write a program that enters information pertaining to loan/credit card balances, interest rates and payback time. The program then computes the amount of a monthly payment, an amortization schedule, a credit card payoff and also displays the total payments and total interest paid.

M.C. Chapter Test

Unit eleven introduces students to programs that involve considerable complexity. In the real world conditional statements are rarely simple. They tend to be compound. In the real world there also needs to be input protection against erroneous data entry. This chapter addresses these issues.

The chapter also introduces recursion very briefly as a means to control program execution sequence. It is meant to set the stage for a future chapter.

In addition, this chapter also presents a set of output exercises to test knowledge of variable tracing.

The program assignment is a large program involving formulas and control structures that are used to compute interests and payments. This assignment is so much more than simply an application of compound conditions and nested looping. Few students, and often adults, truly understand the nature of interest. This lab assignment demonstrates the size of the monthly payment that is made for various loans. It shows how incredibly long it can take to pay off a credit card when the minimum payment is made. Students are often quite amazed when they run their programs and see that on many credit card scenarios they will not live long enough to reduce the balance to zero by making minimum payments only.
Chapter 12 introduces the Java static array. This array does involve the creating of objects with the `new` operator, but students quickly learn that these arrays do not behave like most classes. There is no constructor call and there are no methods.

Students need to learn that `length` is a `final` field and not a method.

Using an array, static or otherwise, inside the `main` method is poor Object Oriented Design. The `List` case study places the array data structure inside a class and performs fundamental access procedures. Implementing sorting and searching methods will be shown in a later algorithm chapter.

Students do learn about sorting and searching at an abstract level. They will use Java’s `Arrays` class to perform various array processing, including sorting and searching.

In this unit students will finally understand the meaning of the `main` method parameter list by learning how to enter values at the command prompt that will be stored in the `args` array variable.

The chapter finishes by importing static packages that allow access to the static methods of classes like, `System`, `Math` and `Arrays` without using class identifiers.

**Java Static Arrays**

Data structures in general
Static array definition
Declaring a one-dimensional array
Traversing an array
   - With a loop structure using index access and the `length` field
   - With the enhanced `for..each` loop structure
Placing an array as a private attribute in a class
The user-created `List` case study I
Using the `Arrays` class
   - Displaying array elements with `Arrays.toString`
   - Storing identical array elements with `Arrays.fill`
   - Sorting array elements with `Arrays.sort`
   - Searching for an array element with `Arrays.binarySearch`
Potential problems with using a binary search
Input at the command prompt into the `main` method `args` array
   - Executing a program with command line input
   - Accessing `args` array values
   - Converting input with `Integer.parseInt` and `Double.parseDouble`
Static Imports
   - Package requirements to use static imports
   - Static import examples with `System`, `Math` and `Arrays`

**Evaluations**

Objective Quizzes/Exercises

Lab Assignment  
   - Sieve of Erasthenes and Graphics Sorting
Write a program that computes prime number using the “Sieve of Erasthenes.”
Write a program that generates random rectangles, which are displayed on half the screen. The bottom part of the screen shows the same rectangles sorted in ascending order.

M.C. Chapter Test
Advanced Graphics

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
  - Reserving virtual memory with `Image` and `getGraphics`
  - Page flipping with `drawImage`

Improving animation flicker with the `update` method

Evaluations

There are no exercises/quizzes

Lab Assignment  *Paint Program*
Write a program that performs the fundamental functions shown by the Windows *Paint* program. The assignment is open-ended in the sense that a precise display is not required. Students are expected to create their own graphical interface. The assignment is scored on the number of *Paint features* that are shown.

There is no chapter test
### Serious OOP

OOP terminology
- Modules, Structured Programming, OOP Definition, Encapsulation, Instantiation, Inheritance, Polymorphism, Class, Object, Instance, Attributes, Instance Variables, Methods
- Default, no-parameter constructors
- Constructor overloading
- Accessing attributes and methods
  - `private` access
  - `public` access
- Accessing multiple files and classes
- Get methods and Set methods
- Copy constructors
- Scope of an object
- Objects are references
- All parameters are passed by value
  - Primitive actual parameter values can never be altered
  - Object actual parameter attributes can be altered
- Using the "this" reference
- Mixing class methods and object methods

### Evaluations

Objective Quizzes/Exercises

Lab Assignment  *The FishGfx Program*
For Lab14a write a program that uses a provided FishGfx class to manipulate the movement of fish.
For Lab14b write an implementation of a FishGfx class.

M.C. Chapter Test

Unit fourteen is very important. Teaching computer science suffers from trying to talk seriously about a topic when students have insufficient knowledge to appreciate all the good points teachers try to make.

Object Oriented Programming has been introduced since early in the course. There have been many chapters where different OOP topics have been emphasized.

This chapter once again discusses many OOP topics, starting with a lot of the OOP vocabulary. Keep in mind that this chapter is not strictly a review or rehashing of previous topics. As old topics are reviewed they then continue and include a more thorough look at the technical details.

One of the most important concepts that students need to clearly understand at this time is the idea that all objects are references and that parameters are all passed by value. Understanding these two related topics allows students to design correct methods that alter values as intended without side effects caused by unwanted aliasing.

The chapter finishes with some examples of how and why you may want to mix class methods and object methods. These may not be tested directly, but this concept gives a deeper understanding of class methods and object methods.
Week 4

Working with Large Programs and Object Oriented Design

- Simple projects
- Complex projects with JAR files
- Advanced Java comments with Javadoc

Timeless Design Issues
- Read and understand a problem description, purpose and goals
- Program documentation
  - Self-documenting identifiers & Program comments
- Modular programming
- Functional Decomposition (Top-down design & step-wise refinements)
- Testing and Debugging
  - Compile, runtime and logic errors
  - Understand common runtime exceptions; throw runtime exceptions
  - Debugging with a debugger, output statements and hand-tracing code
- Test classes and libraries in isolation and then perform integration testing
- Identify boundary cases and generate appropriate test data

Object Oriented Design
- Identify appropriate classes
  - Using existing classes & Creating new classes
- Class design and implementation
  - Establishing a class hierarchy with inheritance and composition
- Method design with assertions about pre-conditions and post-conditions
- Java program writing code conventions

Evaluations

- Free response exercises and quizzes
- Lab Assignment  The Design Program
  Design a program based on provided specifications. The program requires classes and method stubs. Method implementations are not required.
- M.C. Chapter Test

Teaching program design lacks the clean objectivity of teaching program code features. Programs design is one of the topics that is taught throughout the course as design principles become more and more necessary.

Chapter VII introduced program design in an elementary style. The emphasis was on modular programming. The early design chapter could not yet benefit from a fundamental knowledge of Object Oriented Programming.

After concluding the previous chapter, titled Serious OOP, Object Oriented Design becomes a viable topic.

This chapter has a second goal of working with large programs. It is precisely when programs become large that program design issues are properly motivated.

At this stage, working with projects is introduced as a means to organize the multiple files that are used by large programs. The chapter concludes with the GridWorld Case Study. The case study is an excellent example of a large program and it demonstrates how to use JAR files with a project. Additionally, the case study also includes good examples of using the "Javadoc" style of commenting.
# ExposureJava, Chapter 16


## Week 5

### String Processing and Number Systems

- Constructing different **String** objects
- **String** concatenation
- Working with substrings
- Changing immutable **String** objects
- Converting **String** objects
- Comparing **String** objects
- Counting in other number systems
- Counting in base-16
- Converting from base-x to base-10
- Converting from base-10 to base-x
- Converting between base-2 and base-16

String processing is an important computer science topic. This chapter includes all the required AP Java subset String methods and a few extra methods.

The number systems topic is intentionally added to this chapter. Number system conversion is an AP Computer Science topic and can be handled as a stand-alone unit. Any number system conversion between base-16 and other bases involves the processing of numbers and letters, in short it involves string processing.

The lab assignment at the end of this unit combines the two apparently unrelated topics. Students need to write a program that converts numbers between any two bases.

This assignment requires a clear understanding of number system conversion and continues with an understanding of string processing to implement the program assignment. The string processing becomes necessary due to the conversion with Base-16 numbers, which includes "non-numerical digits".

### Evaluations

- Objective exercises
- Objective quizzes and free response quizzes

Lab Assignment: **Number System Conversion**
Write a program that converts a number from any base into any other base

M.C. Chapter Test
### Exposure Java, Chapter 17

http://java.sun.com/docs/books/tutorial/essential/io/

### Week 6 and Week 7

#### Input/Output with Sequential Files

- Different types of files
  - Sequential access files
  - Random access files
- Using the File class
  - Determine file existence
  - Determine file properties
- File IOException handling and throwing file handling exceptions
- Input text files with BufferedReader and FileReader
  - Wrapper class concept with anonymous objects
  - File buffer concept
- Output text files with BufferedWriter and FileWriter
- Handling text files with integer and double values
  - Storing numerical values in a text file
  - Converting numerical strings into `int` and `double` values
- A note about the Scanner class and file handling

#### Evaluations

- Free response exercises
- Free response quizzes
- Lab Assignment  **The Student Records Program**
  - Write a program that reads in and manipulates student data
- M.C. Chapter Test

---

The entire file unit is optional from an AP Computer Science Examination point of view. On the other hand, files handling is extremely important. Any student who receives college credit and continues computer science in college will be expected to have at least an elementary file-handling understanding.

The CollegeBoard has decided not to test file handling with the Java language due to the tremendous number of classes that are available for this concept. However, the importance of this topic is expressed in the course description even if it is not tested.

All the file handling in this unit is done with a minimum of classes and only with text files. Numerical files are still treated as text files and then converted where necessary.

Students have already used the `Scanner` class for keyboard input. It is possible to use the `Scanner` class for file input as well. However, this creates an odd symmetry between file input and file output. Students also learn the file buffer concept better by using two file handling objects wrapped around each other. This approach has proven to provide a deeper understanding of file processing.
SEMN 2

SEM 2

Exposure Java, Chapter 18
http://en.wikipedia.org/wiki/Algorithm

Algorithms I and Informal Algorithmic Analysis

The user-created List class case study
Improving input and output
The linear search
The bubble sort
Array traversals, insertions and deletions
The selection sort
The insertion sort
The binary search
Sorting an array of records
The merge sort concept
Select appropriate data and algorithms
Testing algorithms with the user-created TimeTest class
Informal comparisons of algorithm running times
Exact calculation of execution counts

Evaluations

Free response exercises
Free response quizzes

Lab Assignment The Student Records Program
Write a program that continues the Lab17 assignment and now includes the sorting and searching of student information using static Java arrays in a List class.

M.C. Chapter Test

Students continue to build the List case study that was started in the earlier static array chapter. In this chapter the sorting and searching algorithms are now implemented.

The Bubble Sort is intentionally taught first. It is a very simple quadratic sort to teach and it is an excellent stepping stone for informal algorithmic analysis. Students learn to identify algorithmic weakness and explore improvements. For instance, the Bubble Sort is continuously swapping during each comparison pass. If the swapping is postponed until the end of the comparison pass, then only one swap routine is required and this motivates improvement with the Selection Sort.

Students observe the comparison between different algorithms by using a TimeTest class. This user-created class provides a convenient method to display elapsed time. Students are not expected to calculate analysis with formal Big-O notation, but they can informally observe and discuss the behavior of algorithms. Students also learn to perform exact execution counts.

The student record program, started in Unit 17, is continued. Convenient file access reads in all the records, which can now be processed for sorting and searching.
<table>
<thead>
<tr>
<th>Week 10 and Week 11</th>
<th>SEM 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recursion I</strong></td>
<td></td>
</tr>
<tr>
<td>Recursion definition</td>
<td></td>
</tr>
<tr>
<td>Recursion requires an exit with a &quot;base&quot; case</td>
<td></td>
</tr>
<tr>
<td>Recursion fundamental rules</td>
<td></td>
</tr>
<tr>
<td>Recursive void method examples</td>
<td></td>
</tr>
<tr>
<td>Recursive return method examples</td>
<td></td>
</tr>
<tr>
<td>Fibonacci, an inefficient recursive solution</td>
<td></td>
</tr>
<tr>
<td>Evaluating recursive methods</td>
<td></td>
</tr>
<tr>
<td>Manipulating parameters of recursive methods</td>
<td></td>
</tr>
<tr>
<td>Multiple recursive calls and the &quot;Tower of Hanoi&quot;</td>
<td></td>
</tr>
<tr>
<td>Why recursion?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evaluations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective and free response exercises</td>
<td></td>
</tr>
<tr>
<td>Objective and free response quizzes</td>
<td></td>
</tr>
</tbody>
</table>

Students need to have a clear understanding of the recursive process and need to know how to evaluate recursive methods.

This chapter has extensive exercises and quizzes to give students practice in evaluating recursive methods.

This unit has two lab assignments. The first assignment involves rewriting existing iterative methods into recursive methods. The second assignment requires writing a fractal program. Many fractal programs involve sophisticated mathematics, such as complex numbers. The square fractal focuses completely on the recursive process.

**Lab Assignment** *The Recursive Method Program and The Square Fractal Program*

Complete a program, which contains seven iterative methods and rewrite the method recursively.

Write a program that displays a square fractal

M.C. Chapter Test
| SEM 2 | Exposure Java, Chapter 20  
http://java.sun.com/docs/books/tutorial/collections/interfaces/index.html |
|-------|-------------------------------------------------------------------|
| Week 12 | **The ArrayList Class, Redefining Methods, Autoboxing and Generics**  
Declaring the **ArrayList** class  
**ArrayList** methods  
Handling **ArrayList** objects with class casting  
Handling **ArrayList** objects with a "templated class" declaration  
Accessing an **ArrayList** of records  
Redefining Methods  
The **Object** class  
Redefining the **toString** method  
Redefining the **equals** method  
Autoboxing  
Automatically wrapping a primitive data value into an object  
Automatically "unwrapping" an object into a primitive data value  
Generics  
Declaring an **ArrayList** object with specific class elements  
Combining autoboxing and generics with **ArrayList** objects |

| Evaluations | Students need to see a comparisons between Java static arrays and the **ArrayList** class to appreciate the need to learn both array implementations.  
Using **ArrayList** methods is quite simple for students, but processing objects nested with multiple levels inside other objects is quite complex.  
The program assignment intentionally returns to the Student Records program. This program has multiple levels of composition and teaches students precisely at which level various **ArrayList** objects and other objects need to be instantiated.  
The **ArrayList** class also motivates the concept of redefining and implementing methods. Static java arrays require a loop structure to display individual elements. **ArrayList** objects simply display all the elements with a **print** method call. It is at this point that the true nature of the **toString** method and other methods like **equals** can be explained for maximum understanding.  
The **ArrayList** chapter is a convenient location to introduce the new autoboxing and generics features found in Java 5.0. When both autoboxing and generics are combined, **ArrayList** processing with primitive data types becomes much simpler. |

| | Objective and free response exercises  
Objective and free response quizzes  
Lab Assignment  **The Student Records Program**  
Repeats the Lab18 program, but now uses ArrayList objects  
M.C. Chapter Test |
## Interfaces, Abstract Classes and Polymorphism

Classes and interfaces  
Implementing interfaces  
  Implementing existing interfaces like `Comparable`  
  Implementing user-created interface  
Implementing multiple interfaces  
Using fields in an interface  
Abstract classes  
Polymorphism  
The Object class and polymorphism

## Evaluations

Objective and free response exercises  
Objective and free response quizzes  

Lab Assignment  
There is no lab assignment meant specifically for this chapter, because students will immediately do a major unit with many labs for the Grid World Case Study, which uses interfaces and polymorphism.

M.C. Chapter Test

Students first learn the difference between classes and interfaces and then learn how to implement interfaces.

This unit also continues the concept of redefinition shown in the last chapter with `toString` and `equals`. This time method `compareTo` is defined; although not exactly redefined, but implemented.

The chapter concludes with an explanation of the usefulness of interfaces and abstract classes. At this point students learn about polymorphism.
Exposure Java, Chapter 22

practice exam questions from Georgia Tech Univ.
http://manatee.cc.gt.atl.ga.us/apExam/

GridWorld Case Study, Narrative, Chapters 1-4

Barron’s AP Computer Science 2008:
"A" Sample Exam I and "A" Sample Exam II

Barron’s Review chapters:
- Introduction
- Ch1 (Intro. Java Language Features)
- Ch2 (Classes & Objects)
- Ch3 (Inheritance & Polymorphism)
- Ch4 (Some Standard Classes)
- Ch5 (Program Design & Analysis)
- Ch6 (Arrays & Array Lists)
- Ch7 (Recursion)
- Ch13 (GWCS)

The GridWorld Case Study and Preparing for the AP Exam

Why is there a case study on the AP Computer Science Examination
What is the GridWorld Case Study?
Compiling and executing the GWCS
Understanding the classes and methods of the GWCS abstractly with Javadoc
Understanding the GWCS classes and methods at the implementation level
Altering existing GWCS methods
Creating new GWCS methods

The AP Computer Science Examination format
The significance of the AP Java subset
The AP Java standard libraries
How the GWCS is tested
Sample multiple choice questions
Sample free response questions
The grading of the response questions
Exam day important reminders
Take two APCS "A" sample examinations

(The Evaluations section is on the next page)

At this stage the end of the course is rapidly approaching. Students must realize that the AP Exam is not at the end of the school year, but the first week in May.

Students will work through a very concentrated three-week unit on the GridWorld Case-Study. Work with the GWCS is not simply included, because it is on the test. Students will get an excellent review of all computer science topics during this unit.

The case study evolves during the three-weeks where students start by observing the program execution, continue with investigation of the core classes and supporting classes. After students have gained more understanding, they will then start to make small changes to existing methods. Near the end of the unit students will implement new methods and make major changes to the program.

The last Exposure Java chapter includes a thorough preparation for the AP Computer Science Examination. Students have already learned all the knowledge to take the examination. This final review makes them totally familiar with the exam format and warns students about the grading process to maximize their performance.
Evaluations

Objective and free response exercises
Objective and free response quizzes

Lab Assignments
There are extensive lab assignments for the GridWorld Case Study.
At the "A-level" there are eight labs starting with compiling the GWCS and steadily evolving by altering existing methods and actor behavior and then concluding by creating original methods that enhance the GWCS capabilities.

M.C. Chapter Test

Students also take two AP Computer Science "A" Practice Examinations and review all topics using Barron’s AP Computer Science book

Students practice online questions from Georgia Tech University’s site