Java Camp

Penn State University – Berks Campus
Room 142 Luerssen Hall
July 20-25, 2003
Instructor – Mr. Curt Minich

Course Objectives –

1. Be able to recite the Java hello world program.
2. Be able to use the JCreator IDE to type, compile, and execute a Java source file.
3. Be able to use primitive variables & constants of the types int, double, and boolean as well as String object variables in console programs.
4. Be able to use if statements, loops, and arrays with console programs.
5. Compile and execute a project that makes use of two or more classes & understand the role of object variables and methods.
6. Be able to write a GUI (graphical user interface) Java application.
7. Be able to compile and execute an applet within a web page.
8. Be familiar with the College Board AP Computer Science A Exam topics.
9. Be familiar with the vast Java tutorials on the Web, especially those on Sun's web site.

Daily Schedule –

8:30 am	computer lab is open
8:30 am – 9:00 am	day student drop-off
M & W 8:30 am – 10:00 am	self-directed Java review exercises & activities
T, R, & F 8:30 am – 10:00 am	instructor demonstrations, lecture, & discussion
10:00 am – 10:15 am	morning break
10:15 am – 12:00 pm	instructor demonstrations, lecture, & discussion
12:00 pm – 1:00 pm	lunch
1:00 pm – 2:30 pm	instructor demonstrations, lecture, & discussion
2:30 pm – 2:45 pm	afternoon break
2:45 pm – 5:00 pm	instructor demonstrations, lecture, & discussion
5:00 pm – 5:30 pm	day student pick-up

Final Project Option #1 - In addition to our Pet class, create 2 or more other classes such as Dog, Cat, Bird, etc. Add these separate class files to a folder named FinalProject and in a class named FinalProject, allow a store customer to select one or more pets from a menu and then use modifier methods to give names and other attributes to those pets.

Final Project Option #2 - Create a Java applet that uses 10 or more Graphics methods to draw an interesting picture.

Objective #1: Be able to recite the following hello world program.

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World");
   }
}
```

- Java is case-sensitive.
- The name of the class is HelloWorld. It is considered good style to always capitalize the name of a class.
- Since the name of the **class** is HelloWorld, this file must be saved as HelloWorld.java . In Java, a program typically consists of many classes each saved to a different file with the .java file extension.
- Like C++, most statements end with a semicolon.
- This simple program consists of just one **method** (which are called functions in C++ or procedures in Visual Basic) named `main`. However, you'll soon learn how to add other methods to the body of the HelloWorld class.
- At this stage of your learning, it's simplest if you simply memorize the `public static…` line of code. You will not be using the `args` parameter for any programming in an AP course.
- `System.out` is the name of an **object variable** (though often referred to as just "object") that is automatically included for use in every Java program. `println` is a method that is being used to display the string message "Hello World" to the console window (i.e. screen).
- There are three kinds of Java applications (i.e. programs) that we may have time to study:
    - Console programs, which run in the console window and even allow the user to type input directly into the console window.
    - GUI (graphical user interface) applications, which use the AWT, Swing, or another graphical Java library to present a more graphical, interactive environment to the user.
    - Applets, which are graphical, interactive Java programs that can execute in a browser window such as Internet Explorer.

  Your AP teacher and textbook may not use all three kinds of programs by the way.
- The HelloWorld program is an example of a console program. Console programs must include a `main` method in one of the class files that make up the program. The main method is where the program's execution begins.
- Activity: Type out the HelloWorld program from memory.
- Activity: Add more `System.out.println` statements to the body of the main method but change the string literals to another phrase.
- Activity: Add a `System.out.print` statement (i.e. without the `ln`) and explain the effect.
- Demo programs: HelloWorld as console program, as GUI application, and as an applet

Objective #2: Be able to use the JCreator IDE to type, compile, and execute a Java source file.

- Remember that when you type a source file, you must save it with the same exact file name as the name of the class that it defines. The file name and the name of the class are both case-sensitive.
- Follow these instructions to create, compile, and execute your first Java program. Use the HelloWorld program source code.
    1. Create a folder named HelloWorld on your floppy disk.
    2. Open the JCreator IDE (integrated development environment) program. Technically, JCreator is not a Java compiler since, behind the scenes, it calls upon a program called javac, that comes with the Sun Java 2 SDK (Software Development Kit), to do the actual compiling. But I often call JCreator a compiler out of habit.
    3. Click the File/New menu command.
    4. Select the Files tab.
    5. Select the Java File icon.
    6. Type the file name HelloWorld.java.
    7. Specify the HelloWorld folder on your floppy disk as the location of the file.
    8. Type the HelloWorld source code.
    9. Click the Build/Compile File menu command.
    10. Click the Build/Execute File menu command.
    11. Press any key to continue.
    12. Click the File/Exit menu command.
    13. Click Save if prompted.
    14. Use Windows Explorer to see that there are actually two files in your HelloWorld folder. One file is the HelloWorld.java source file that you typed out. The other file named HelloWorld.class was created when you compiled your code.
- When you use the Build/Compile File menu command, JCreator creates a .class file from your .java source file. The class file consists of **bytecode** which is compiled Java code that can be understood by the computer but not by a programmer. The .java file is the file that you must edit if changes must be made.
- Unlike the Visual C++ & Visual Basic compilers, when you make changes to a Java program, you must always re-compile a source file before re-executing it. If you don't re-compile, JCreator will execute the previous .class file and ignore the changes that you made.
- Mrs. Litvin gives tips to using JCreator at www.skylit.com/javamethods/faqs/jcreator2.html
- Mr. Wittry gives advanced tips to using JCreator at www.troyhigh.com/wittry/ap/JCreator.htm
- Activity or hacking contest: Write and execute a program that prints this face:

```
   / / / / /
  |    o o    |
( |      ^    | )
  |    \_/    |
   - - - - -
```

- Activity: Visit JCreator.com to learn how to download the JCreator IDE and visit http://java.sun.com to learn how to download Sun's Java SDK. Visit http://www.minich.com/education/wyo/java/ch1/notes.htm for instructions on installing Sun's Java SDK.
- Activity: Watch Mr. Minich compile and execute Java console programs from the DOS prompt without using JCreator.
- Activity or hacking contest: Write & execute a program that computes & displays sum of the first 20 positive integers (i.e. 1 + 2 + …. + 20). Hint: The statement `System.out.println(3 + 5);` will display the value 8.
- Activity or hacking contest: Write & execute a program that computes the sum of the reciprocals 1/1 + 1/2 + 1/3 + … + 1/10.

Objective #3: Be able to use primitive variables & constants of the types int, double, and boolean as well as String object variables in console programs.

- When you declare a variable, you must declare it with a specific **type** (aka data type) such as int, double, or boolean. These kinds of variables are formally called **primitive variables** (as opposed to object variables like System.out and others that we'll study later) but you'll hear most people just refer to them as variables.
- The three most useful types for an AP course are:
  - `int` for integer (i.e. whole numbers)
  - `double` for floating-point (i.e. decimal numbers)
  - `boolean` for the values, `true` and `false`
- Variables can be declared and initialized with statements like:

```
int num = 0;
double price = 9.99;
boolean foundYet = false;
```

- You can assign values to variables with assignment statements like:

```
num = 10;
```

  as long as the variable has already been declared.
- The familiar mathematical operations +, -, *, /, and % (modulus) can be performed on int and double variables.
- The incrementing and decrementing operators (++ and --) can be used in Java like they are in C++.
- You can **cast** (aka typecast) an int to a double or vice versa by using the Java cast operators `(int)` and `(double)`. For example, the code segment:

```
int myInt = 0;
double myDouble = 3.6;
myInt = (int)(myDouble + 0.5);
```

  is valid and causes the integer value 4 to be stored in the integer variable `myInt`. This rounding algorithm is quite useful by the way.
- You can create a **constant** by using the keyword `final`. Add the following statements to the bottom of HelloWorld.java source file.

```
final double PI = 3.14;
double radius = 5;
double area = radius * PI * PI;
```

  PI is a constant and cannot be changed with a statement such as:

```
PI = 5.23;
```

  since the keyword `final` was used in its declaration statement. It is good style to name constants in ALL CAPS.

- Strings can be declared and initialized similarly as in:

```
String name = "John";
```

  But strings can also be declared using the `new` operator

```
String name = new String();
```

  and later assigned a value with an assignment statement like:

```
name = "Jane";
```

Since `String` is technically a class and not a primitive variable type (note the capital 'S'), in the example above, you should probably refer to `name` as an object (i.e. object variable) rather than a variable (i.e. primitive variable). The `String` class is automatically included for use in every Java program.

- It is a little messy, to allow user input into a console program. First, add the following statements to the top of your HelloWorld.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
```

Then, add these lines of code to the bottom of the main method:

```
BufferedReader console = new BufferedReader(new
   InputStreamReader(System.in));
System.out.print("Enter a number: ");
String inputNumber = console.readLine();
int num = Integer.parseInt(inputNumber);
System.out.print("Enter a word: ");
String word = console.readLine();
```

and add "…. `Throws IOException`…" to the first line of the `main` method to change that line of code to:

```
public static void main(String[] args) throws IOException {
```

Later, you'll learn about object variables such as `console`, library classes such as `BufferedReader` and `InputStreamReader`, and methods such as `readLine` and `parseInt`. But for now, this code can be used to allow the user to input numbers and text directly into the console window. When you execute this code, `num` is a valid integer inputted by the user & word is a valid string inputted by the user.

The good news is that it is not required that you memorize or even understand these messy details regarding console input for the AP exam. AP exam questions will be worded in a way that avoid dealing with how data is inputted. However, receiving user input in console programs makes it easier to learn the Java language. Your AP teacher may not use this basic method for inputting data to console programs. Instead, your teacher may teach you how to use the EasyReader and EasyWriter classes that were written by Maria & Gary Litvin. These classes make it a bit easier to obtain user input into console programs and are freely available. More info is available at http://www.skylit.com/javamethods/appxe.html

- Demo programs: Fahrenheit to Celsius conversion with hardwired values, plenty of examples of user input, examples of casting and rounding, etc.

Objective #4: Be able to use if statements, loops, and arrays with console programs.

- if, if/else, and if/else if statements work in Java just like they do in C++. switch statements also work in Java but are not tested on the AP exam.
- while and for loops work in Java just as they do in C++. do/while loops also work but are not tested on the AP exam.
- The relational operators = =, !=, <, <=, >, and >= can be used in Java if statement and loop control expressions just as they are in C++. Be sure though to use the `Equals` method to compare String objects rather than the double equals (==).
- The logical operators && (AND) , || (OR), and ! (NOT) can be used in Java if statement and loop control expressions just as they are in C++.
- You can use arrays of primitive variables (such as int's and double's) or arrays of object variables. The following statement declares an array of integers named myList:

  ```
  int[] myList = {1, 2, 3};
  ```

  Note to veteran C++ programmers: It is good style to place the square brackets before the variable name in Java even though it is not an error to place them after the variable name.
- Demo Programs: Equals rather than == with strings, allow user to order one item from a store's menu, allow user to order several items from a menu, sequential search, binary search guess number between 1 and 100 game, etc.

Objective #5: Compile and execute a project that makes use of two or more classes & understand the role of object variables and methods.

- Classes & Projects
  - A Java program that uses two or more classes can be called a **project**. However, each class must be typed into a different file with the extension .java . For now, you should always place the separate class files that make up a project into the same folder. Until you learn how to formally create a project with JCreator, you should fall into the habit of first manually creating a folder to store a project's class files. I recommend naming the folder with the same name as the class file that contains the `main` method.
- Object Variables
  - In the HelloWorld program, `out` is an **object variable** of the `System` class. In this case, you are free to use the `out` object without declaring it, since the `System` class is automatically included with every Java console program. However, you'll soon learn how to create your own object variables by writing your own classes. Therefore, object variables are very different from primitive variables which are always of the type `int`, `double`, or `boolean`. By the way, object variables are often called objects.
  - You typically have to use the `new` operator when declaring most objects. (In the HelloWorld program, the `System.out` object is automatically created by Java.) Here is one way to use a `Rectangle` object:

    ```
    System.out.println(new Rectangle(5, 10, 20, 30));
    ```

    In a console program, this does not cause a rectangle to be drawn on the screen but it does produce a line of output that you'll have to check for yourself. The 4 parameters represent a top-left corner position of (5, 10) along with a width of 20 and a height of 30.
  - You have access to hundreds (or maybe even thousands) of different objects due to the classes that come in many **packages** which are installed along with Java. One example is a Rectangle object.
  - Unlike the System & String classes, you must import the Rectangle class by typing the statement

    ```
    import java.awt.Rectangle;
    ```

    at the top of the source file. The Rectangle class is found in the java.awt package. (Technically, java is considered to be the package with awt as the name of a subpackage. The statement

    ```
    import java.awt.*;
    ```

    could be used instead but that would unnecessarily include all of the classes in the java.awt package.)

  - You typically have to use the `new` operator when declaring objects. (In the HelloWorld program, the `System.out` object is automatically constructed by Java.) Here is one way to use a `Rectangle` object:

    ```
    System.out.println(new Rectangle(5, 10, 20, 30));
    ```

    The 4 parameters represent a top-left corner position of (5, 10) along with a width of 20 and a height of 30. By the way, in a console program, this does not actually draw a rectangle on the screen but it does produce some output that you'll have to check for yourself.
  - Or, you can first declare a Rectangle object variable with one statement and then display it with a separate statement as in:

    ```
    Rectangle myBox = new Rectangle(5, 10, 20, 30);
    System.out.println(myBox);
    ```

    In this case, the object variable `myBox` can also be reused later in the program.
- Methods
  - `println` is a **method** of the System class. Methods are somewhat similar to functions in C++ and procedures in Visual Basic. Methods perform actions when used in conjunction with an object.

Typically, you type the name of an object followed by the dot operator (i.e. period symbol) and then the name of the method.

- Technically, the string "Hello World" is an even treated as an object of the String class and length is a method of the String class. The String class is another class that is automatically included with every Java console program. Try executing the lines of code

```
int num = "Hello World".length();
System.out.println(num);
```

and you'll see that the number of characters in "Hello World" (11) displays! This may look odd to veteran C++ programmers.

- Defining your own classes with methods
  - You can define your own class and include as many methods in it as you would like. The class does not have to have a `main` method as long as you don't plan to execute its class file.
  - Use the File/New menu command to create another source file and then type out the following code and save it as Dog.java in the same folder as your HelloWorld program. Use Build/Compile File to compile this class into its own .class file.

```
public class Dog {
   public String bark() {
         return "Woof";
   }
}
```

Next, add the following statements to your HelloWorld program and execute it.

```
Dog myPet = new Dog();
Dog yourPet = new Dog();
System.out.println(myPet.bark());
System.out.println(yourPet.bark());
```

You have declared two object variables named `myPet` and `yourPet`. The `bark` method that you created is called, it returns the string "Woof" which is then displayed by the `println` method.

  - You can **overload** a method in a class by defining another method with the same name but that has a different **signature** (i.e. the number, types, and order of its parameters). Type the following code just below the bark method in the Dog.java class file.

```
public String bark(int numBarks) {
   int i = 0;
   String myBark = new String();

   for (i = 0; i < numBarks; i++) {
         myBark = myBark + "Bark";
   }

   return myBark;
}
```

and add the line of code

```
System.out.println(myPet.bark(3));
```

towards the end of your HelloWorld.java source file and execute that file. This is called overloarding a method since there are two bark methods in the Dog class. The compiler understands that this bark method is different from the first because it requires an integer parameter. The value 3 is passed to the bark method which accepts the numBarks parameter. That bark method then returns the string value "BarkBarkBark" since the + symbol is the string concatenation operator when surrounded by one or

more string values.

- Instance fields
  - You can give an object "**state**" by supplying one or more **instance fields** (sometimes simply called fields or member variables) in a class. By the way, it is proper to say that the methods of a class give an object "**behavior**". This allows two or more object variables to be different with regard to the values that are stored in each one's instance variable.
    Above the bark method, add the following methods, getName and setName, to the Dog.java class file.

```
public String getName() {
   return name;
}

public void setName(String myName) {
   name = myName;
}
```

    and add this line of code to the very bottom of the Dog.java class file.

```
private String name;
```

    The variable `name` is called an instance field. Instance fields are usually made private (as opposed to public) in order to hide this data in the name of **encapsulation**. The programmer who codes the HelloWorld class file that uses a Dog object variable cannot directly use the name instance field since it is private.
- Making your own methods
  - For practical reasons, this then requires an **accessor** method like `getName` as well as a **modifier** method like `setName`. Accessor methods are often named with the prefix `get` and modifier methods are often named with the prefix `set`. Both accessors and modifiers are usually public methods so that the programmer who codes the HelloWorld class file that uses a Dog object variable, can assign and access the name of a Dog object. Notice that myName is a parameter to the method setName. Also notice that getName returns a String value.

    Next, add the following code to your HelloWorld.java source file:

```
myPet.setName("Gizmo");
yourPet.setName("Fido");
System.out.println(myPet.getName() + " " + yourPet.getName());
```

  - Rather than always having to use accessors to assign values to instance fields, it is even more efficient (and good style) to add a **constructor** to the Dog class. Type the following code just below the name declaration statement in the Dog.java source file.

```
public Dog(String myName) {
   name = myName;
}
```

    Now change the Dog object variable declaration statements to:

```
Dog myPet = new Dog("Gizmo Jr.");
Dog yourPet = new Dog("Fido Jr.");
```

    and comment out the two `setName` statements**. Comment statements** in Java can be created by beginning the line of code with the `//` operator.

    Note that as soon as you add one or more constructors to a class, Java no longer supplies a default constructor so the statement:

```
Dog myPet = new Dog();
```

would no longer work in HelloWorld.java source file.

- One or more methods in a class can be designated as **class methods** with the keyword `static`. We sometimes call these static methods. The Math object that can be used from the Math class that is part of the java.lang package includes the useful static methods `abs`, `pow`, and `sqrt`.

  Add the following code to the end of HelloWorld.java to test these methods:

```
System.out.println(Math.abs(-3));      // absolute value of 3
System.out.println(Math.pow(2, 3));    // 2 to the 3rd power
System.out.println(Math.sqrt(81));     // square root of 81
```

  To use class methods, you simply refer to the name of the class as if it was an object itself and then type the dot operator (.) followed by the name of the method along with any required parameters.
- Add this class method to the Dog.java class file below all of the other methods:

```
public static String family() {
   return "Canidae";
}
```

  and add this line of code to the bottom of HelloWorld.java:

```
System.out.println(Dog.family());
```

  Since the biological family name of any dog is "Canidae", it makes no difference whether it is `myPet` or `yourPet` object. Therefore, the family method is more suitable as a class method rather than a regular method.
- `substring` is a particularly useful class method of the String class and can be used like this:

```
String name = "Jingleheimerschmidt";
String sub = name.substring(0, 6);     // sub is "Jingle"
```

  where the first parameter is the starting point (the first character `J` is in position zero) and the second parameter is the position of the character <u>just beyond</u> the end of the extracted substring.

- Be careful to understand the true difference in the values that are stored in primitive variables versus object variables. When you assign one primitive variable's value to another primitive variable, you are actually copying the value stored in the original variable and then storing that value in the second variable.

```
int num1 = 5;
int num2 = num1;           // the value 5 is being stored in num2
num1 = 8;
System.out.println(num1);  // displays the value 8
System.out.println(num2);  // displays the value 5
```

  The code above works just as C++ and Visual Basic programmers would expect. However, most of the time when you assign an object variable to another object variable, you are actually only copying a reference (i.e. a memory address) rather than an actual value. Object variables themselves don't really store numeric or string values. Instead, they store references (which you can consider to be memory addresses.)

```
(1)      Dog myDog = new Dog("Gizmo");
(2)      Dog yourDog = new Dog();
(3)      yourDog = myDog;
(4)      myDog.setName("Fido");
(5)      System.out.println(myDog.getName());
(6)      System.out.println(yourDog.getName());
```

  In Line 1, a "reference" to the string object "Gizmo" is stored in the object variable `myDog`

In Line 2, a null reference is stored in the object variable yourDog.
In Line 3, the reference to the string "Gizmo" is stored in the object variable yourDog
In Line 4, the accessor method setName stores a reference to the string "Fido" in the object variable myDog
In Line 5, as you would expect, "Fido" is displayed since it is the value of the name instance field of the object variable myDog
In Line 6, "Fido" is displayed since yourDog has only been a reference to the myDog object all along and the value of the name instance filed of myDog is "Fido"!

However, since String objects are considered to be "**immutable**" in Java, none of the String class methods (including assignment with = symbol) change the state of each String object.

```
String name1 = "Gizmo";
String name2;
name2 = name1;
name1 = "Fido";
System.out.println(name1); // displays Fido as expected
System.out.println(name2); // displays Gizmo as expected
```

- Demo programs: using JCreator to setup a workspace with one or more projects, the Dog class at each step, classes that use the Math methods, classes unrelated to Dog (Square, Circle, Student, Tank, Team, etc.) that have instance fields, accessors, modifiers & constructors
- Activity: Look up other methods of the System and String classes (besides println and length) in the online Java API.
- Activity: Add other appropriate instance fields (e.g. age, weight, etc.) to the Dog class along with an accessor and a modifier for each one. Also, add several more constructors to the class. Call these new methods from the HelloWorld program.
- Activity: Add other non-accessor, non-modifier methods such as bark to the Dog class and call them from the HelloWorld program.
- Activity: Trace some tricky worksheets to better understand the differences between copying primitive variables and object variables.
- Activity or Final Project: In addition to our Dog class, create 2 or more other pet classes such as Cat, Bird, etc. Add these separate class files to a folder named FinalProject and in a class named FinalProject, allow a store customer to select one or more pets from a menu and then give names and other attributes to those pets.

Objective #6: Be able to write a GUI (graphical user interface) Java application.

- A more interesting kind of Java program is a GUI application which has windows, buttons, and other common graphical and interactive features. Typically, Java programmers use the AWT or Swing library classes in GUI applications.
- There are many ways to add interactivity to a GUI application. One simple but effective way to allow the user to input values is to use the `InputDialog` class method of the `JOptionPane` class.

  First, to use the `JOptionPane` class, you must add this import statement to the top of your HelloWorld.java file:

  ```
  import javax.swing.JOptionPane;
  ```

  and then add the statements

  ```
  String input = JOptionPane.showInputDialog("Enter your name");
  System.out.println(input);
  System.exit(0);
  ```

  to the bottom of your HelloWorld.java file. The `showInputDialog` method of the `JOptionPane` class brings up a graphical dialog box that allows the user to input data (similar to an InputBox in Visual Basic). The user's typed data will be stored in the object variable `input`. The `System.exit(0);` statement is necessary in order to properly exit a program that uses input in this manner.

- The static methods, `parseInt` and `parseDouble`, from the Integer & Double classes can be used to turn a String object (like "13") into an actual `int` or `double` numerical value. For example,

  ```
  int num = Integer.parseInt("13");
  double price = Double.parseDouble("9.99");
  ```

  or

  ```
  String input = "9.99";
  double price = Double.parseDouble(input);
  int price2 = Integer.parseInt(input);        // price2 is 9 not 9.99!
  ```

  Combined with the `showInputDialog` method, you can use

  ```
  String userNum = JOptionPane.showInputDialog("Enter your age");
  int num = Integer.parseInt(userNum);
  ```

  to allow the user to input values into a dialog box. We will assume that the user will always input numerical values when requested. It is beyond the scope of this workbook to explain how to correct the problem that occurs in the example above if the user types a word like "nine" instead of typing the number 9.
- The techniques outlined above require the use of the Swing library that automatically comes with Java 2. But using the AWT (Abstract Windowing Toolkit) or Swing libraries that professional Java programmers often use to build graphical interfaces is quite complicated. So depending on high school's Java compiler, IDE, teacher, and textbook, you may not be using `JOptionPane` or the `javax.swing` package. But a great feature of Java in general is that in only 10 years since the language was invented, many graphical libaries have been made freely available. One graphical library that simplifies things for novice Java programmers (like apstring simplified the C++ string class for AP C++ students) is the BreezyGUI library. Kenneth Lambert & Martin Osbourne have made this library freely available at
  http://faculty.cs.wwu.edu/martin/Software%20Packages/BreezyGUI/breezyhome.htm
- Demo programs: same demos as console program section except with GUI interface.

Objective #7: Be able to compile and execute an applet within a web page.

- Besides console programs and GUI applications, a third, popular kind of Java program is an applet. Applets are exciting because they are interactive Java programs that can be uploaded to the Web and executed there by your friends and family. Some online games are written with Java such as those at www.freearcade.com and www.thinks.com/games
- Currently, Microsoft and Sun are engaged in a lawsuit because Microsoft refuses to include the Java Virtual Machine (JVM) with its Windows operating system. Depending on your computer and web browser, you may not be able to use MS Internet Explorer to execute the applets that you'll soon learn how to make. However, you can download and install the Java Plug-In at http://java.sun.com/getjava.
- Memorize this HelloWeb applet:

```
import java.awt.Graphics;
import javax.swing.JApplet;

public class HelloWeb extends JApplet {
   public void paint(Graphics g) {
         super.paint(g);
         g.drawString("Hello Web", 25, 25);
   }
}
```

This applet uses the more up-to-date `JApplet` class from the `javax.swing` package rather than the older `Applet` class from the `java.applet` package.

Another version of a HelloWeb applet is:

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWeb extends Applet {

   public void paint(Graphics g) {
         g.drawString("Hello Web", 25, 25);
    }
}
```

This applet uses the original `Applet` class from the `java.applet` package.

- An applet does not need a `main` method. However, an applet does need to have a `paint` method. The `paint` method is automatically executed any time the browser window decides to redraw the applet to the screen. Most of your drawing instructions will be placed in the `paint` method. The `super.paint(g);` statement is necessary for more complex applets so it's good to get into the habit of placing that statement at the beginning of the `paint` method even if you don't understand its purpose at this stage.
- The `g.drawString` statement does the actual work in HelloWeb. The 25, 25 coordinates represent the x and y location of the first letter "H" in the "Hello Web" string.
- The parameter g of the paint method is used to keep track of the state of the graphics window. This means that it remembers the current paint color, font, etc.
- Visit the online Java API specifications at http://java.sun.com/j2se/1.3/docs/api to learn more methods of the `Graphics` class. Add statements such as `g.drawRect(10, 20, 30, 40);` to HelloWeb.java.
- Since the Sun Java SDK was installed in this computer lab, you can simply open the .htm web page file that contains the
  `<applet code = "HelloWeb.class" width = "500" height = "300"></applet>`
  tag with Internet Explorer. JCreator named this file HelloWeb.htm.
- The following MouseEventExample.java applet allows you to work with a mouse event.

```
import java.applet.Applet;
import java.awt.*;
```

```
import java.awt.event.*;
import javax.swing.*;
import java.util.Random;

/** user may place a box anywhere in applet window */
public class MouseEventExample extends Applet {

    private Rectangle box;
    Random diceRoller = new Random();

    public MouseEventExample () {
         box = new Rectangle(100, 150, 20, 30);
         JButton button = new JButton("Click", new ImageIcon("button.gif"));

         class ClickButtonListener implements ActionListener {
               public void actionPerformed(ActionEvent event) {
                     int xPosition = diceRoller.nextInt() % 100;
                     int yPosition = diceRoller.nextInt() % 100;
                     box.setLocation(xPosition, yPosition);
                     repaint();
               }
         };
         ActionListener myActionListener = new ClickButtonListener();
         button.addActionListener(myActionListener);

         class MousePressListener implements MouseListener {
               public void mousePressed(MouseEvent event) {
                     int x = event.getX();
                     int y = event.getY();
                     box.setLocation(x, y);
                     repaint();
               }

               public void mouseReleased(MouseEvent event) {}
               public void mouseClicked(MouseEvent event) {}
               public void mouseEntered(MouseEvent event) {}
               public void mouseExited(MouseEvent event) {}
         }

         MouseListener listener = new MousePressListener();
      addMouseListener(listener);

      JLabel promptLabel = new JLabel();
      JPanel panel = new JPanel();
      panel.add(promptLabel);
      panel.add(button);

      JFrame frame = new JFrame();
      frame.setContentPane(panel);
      frame.pack();
      frame.show();
   }

   public void paint(Graphics g) {
         Graphics2D g2 = (Graphics2D) g;
         g2.draw(box);
   }

}
```

- Demo programs: HelloWeb.java & HelloWeb.htm, etc.
- Activity: Write an applet that draws your name in red centered inside a large blue rectangle.
- Activity: Visit http://javaboutique.internet.com/cathome.html or a similar site to download the class files for cool Java applets (including games) that you can upload to your own web site (or test locally on your PC).
- Activity or Final Project – Create a Java applet that uses 10 or more Graphics methods to draw an interesting picture.

Objective #8: Be familiar with the College Board AP Computer Science A Exam topics.

- See http://www.collegeboard.com/ap/students/compsci/java_subsetA.html to familiarize yourself with the official AP Computer Science A Exam Java Subset. These topics are considered to be "fair game" on the AP exam. However, there are many other parts of the Java language that are fun and useful.
- Marine Biology Case Study
    - Maria Litvin has a guide to help you set up Marine Bio with JCreator at http://www.skylit.com/javamethods/faqs/jcreator2MBS.html
    - We will execute the demo Marine Bio GUI applications in class.
    - Differences from Marine Bio with C++: fish don't swim backwards, dynamic fish in Ch. 3 can breed and die, Ch. 4 is about slow & darter fish. Ch. 5 is only tested on AB exam.
- College Board is posting interfaces that all students should look at.
- Arrays and ArrayLists will be covered on the exam but there are no special classes like apstring and apvector.

Objective #9: Be familiar with the vast Java tutorials on the Web, especially those on Sun's web site.

- See Mr. Minich's Java Links page at http://www.minich.com/education/wyo/java/links.htm
- The Java programming language was invented by programmers at the company Sun Microsystems in the early 1990's. The Sun web site at http://java.sun.com is a great place to go to learn more about Java. That web site is also the place where you can download the free Java 2 SDK compiler as well as other tools.
- The complete documentation for the Java library of hundreds of classes such as Math and String can be found at http://java.sun.com/j2se/1.4.2/docs/api
- Use good style when typing Java code. If your teacher, school, or company does not enforce certain coding standards, you can use those published by Sun at http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
- A great deal of help writing applets is found at http://java.sun.com/applets
- Using javadoc with JCreator help at http://www.aul.fiu.edu/docs/UsingJavaDoc.pdf
- If your high school doesn't offer a specific Java class, you can sign up for an online course or learn Java from tutorials like:
    - psvm.org
    - eimacs.com