Anatomy

OF A

JAVA Program

What is a Program?

Before we can see the parts that make up a program, we must first understand what a computer program actually is. A computer program is a series of steps, or commands (also called an algorithm), that the computer executes in order. Any programming language will have rules about how those steps must be written. When we look at the anatomy of a program in JAVA, we are specifically seeking to understand the rules we must follow in order to write a series of steps so that they will work correctly.

In this class, we will be using a program to write, edit, and test our code called Eclipse. Eclipse is an Integrated Development Environment (IDE). An IDE is a program designed to help a user write computer code in a similar way that Microsoft Word is a program designed to help write documents for other people to read. An IDE can be configured to make it easier to write computer code in one or more specific languages. The Eclipse IDE for JAVA makes it easier to read and write computer code by changing the color of text to help indicate what it is doing.

For example, any text that follows two forward-slash marks is considered to be a code-comment, written not for the computer but for a programmer to read.

Code Example 1.1:
```
// This is a code comment
```

Code comments provide information that is useful to a programmer or user reading the code. It can clarify confusing elements in code or simply provide useful information such as the author's name and the date the program was written.

For purposes of this class, you should always begin your programs with comments that identify the Assignment that the program fulfills, a short description of the program, your name, and the date you wrote the code.

Code Example 1.2:
```
// Assignment or Program Title
// A short description of the program goes here
// Author's Name
// Date the Program was written
```

In the JAVA language there are some words that have special meanings. The Eclipse IDE will make these words purple. Some examples include the words `public`, `class`, `static`, and `void`.

After the comment code that identifies the program, author, and date, the next element in a JAVA program is the primary class that serves to identify name of the program.

Code Example 1.3:
```
// Example 1.3
// An empty program that will execute, but DO nothing
// Written by Aaron Braskin
// Created 7/8/2013
public class DoNothing {// Primary Class
  public static void main(String[] args) {// Main Method
    // Your program code starts here!
  } // End of the Main Method
} // End of the Primary Class
```

Read the code comments to find the start of the start of the PRIMARY CLASS. This is identifies the class (also known as the program) that the Java Virtual Machine is going to try to run. The primary class must always have the keywords `public class` before it's name.

After the Primary Class definition comes a special METHOD called the `main()` method. A method is a block of code that can be executed. The `main()` method is the first method that will be executed within the Primary Class. Example 1.4 shows the form the `main()` method declaration should take. It begins with the JAVA keywords `public static void` followed by the method name, `main`, and then it includes a parameter (or argument) list in parenthesis.

Code Example 1.4:
```
public static void main(String[] args)
```

A `main` method will always have the parameters you see in this example, though the use of these parameters is not covered in the AP Subset so we can ignore them for purposes of this class!

The last item you need to be aware of in this simple example is the use of the curly braces `{}`.

In JAVA, curly braces create a CODE BLOCK. A code block groups code together in a hierarchical arrangement.

In this example, the first pair of curly braces comes after the Primary Class declaration. All the code between the two curly braces is part of the `DoNothing` class!

Code Example 1.3:
```
// Example 1.3
// An empty program that will execute, but DO nothing
// Written by Aaron Braskin
// Created 7/8/2013
public class DoNothing {// Primary Class
  public static void main(String[] args) {// Main Method
    // Your program code starts here!
  } // End of the Main Method
} // End of the Primary Class
```

Notice that all the code inside the `DoNothing` class curly braces is indented to show that it is part of this class!

The next pair of curly braces is in the `main()` method. All of the code between the open curly brace "`{`" and the matching close curly brace "`}`" is part of the `main()` method!

This very simple program shows the complete outline of a very basic JAVA program.

We can add a line of code to it to make it do something by adding a line in the `main()` method.

Code Example 1.5:
```
// Example 1.5
// A basic program that outputs a message to the console
// Written by Aaron Braskin
// Created 7/8/2013
public class DoNothing {// Primary Class
  public static void main(String[] args) {// Main Method
    System.out.println("Hello World!");
  } // End of the Main Method
} // End of the Primary Class
```

Example 1.5 simply adds a special command that uses a built in method that is part of the `System` class to output a message to the console.

By default your code has access to a built in set of code libraries to handle certain common operations.

Another part of the code library is the Math class. You can use the Math class to perform simple math operations like calculating a square root.

Example 1.6 demonstrates the use of the `Math.sqrt()` method.

Code Example 1.6:
```
// Example 1.6
// A basic program that outputs a square root to the console.
// Written by Aaron Braskin
// Created 7/8/2013
public class DoNothing {// Primary Class
  public static void main(String[] args) {// Main Method
    System.out.println("The square root of 4 is"+Math.sqrt(4));
  } // End of the Main Method
} // End of the Primary Class
```