From C to Java

Hello World in Java

Consider the following simple Java program:

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

Things to note:

- all Java programs must contain a class, show above as: `public class HelloWorld { ... }`
- If the class name is HelloWorld, it must be in a file called HelloWorld.java

You compile your java program using `javac`, so to compile the above program, stored in the file HelloWorld.java, you would type:

```
     javac HelloWorld.java
```

You run your program by typing:

```
     java HelloWorld
```

Output

In Java, we use System.out.println() rather than printf for output. There are no format specifiers in Java, so to output the contents of an integer, you would :

```
     int x = 9;
     System.out.println("The value of x is:" + x );
```

Note that the + operator here is string concatenation – the compiler converts the integer x into a string automatically and concatenates that string to the string "The value of x is:" before doing the output.

Arrays

In C, an array was just a pointer to the first element in the array. In Java, arrays are actually objects and contain a property called length. There is no longer a need to maintain a separate variable to keep track of the length of your arrays.

Java supports the same style of array initialization as C. However, in Java, we like to use the following syntax for declaring the type of the array:

```
    int[]     a = {1,2,3,4};
```

Note that the [] is next to the type, not the variable name.

Arrays in Java are indexed from 0, just like C.

The syntax for declaring an array of a fixed size without providing an initializer is different in Java. Consider the following code to declare an array of length 100 and initialize the values:

```
    int[]     b = new int[100];
    for (int i = 0; i < b.length; i++)
    {
         b[i] = i;
    }
```

Note the use of b.length to determine the length of the array.

Strings

Java has a built-in string data type. We no longer have to worry about NULL terminators or arrays of characters! Consider the following code to print the characters of a string individually:

```
    String s = "abcdef";
    for (int i = 0; i < s.length(); i++)
    {
         System.out.println( s.charAt(i));
    }
```

Notice:

- the length of the string s is provided by s.length()
- to access the ith character of string s, you use s.charAt(i), you **cannot** use s[i]

Java supports concatenation of strings using the + operator:

```
String s = "foo";
String t = "bar";
String u = s + t;

System.out.println(u);        // will print foobar
```

Booleans

In C, we used integers to represent boolean values – 0 was false and anything else was true. Java has primative type called boolean that has two possible values: true or false. Consider:

```
boolean b;
b = false;
if (b)
{
     // do something
}
```

Static functions

If you want to practice java, you could rewrite some of your assignments from last term. Just be sure to declare all your functions static. Consider the program `Power.java` on the following page as a good place to start.

```
public class Power
{
    public static int power (int a, int b)
    {
        if (b==0)
        {
            return 1;
        }
        else
        {
            return a*power(a,b-1);
        }
    }

    public static void main (String[] args)
    {
        System.out.println("3^3 is:" + power(3,3));
        System.out.println("2^4 is:" + power(2,4));
    }
}
```