A First Java Program

Writing an independent Java program from scratch is new for most of us. Unfortunately, there are a few things that every Java program must do, and I wouldn't exactly call those things intuitive. We will go over the deeper significance of the language and syntax as the semester progresses. For now, here are two observations:

1. Everything in Java is a `class`. When you create a Java program, it is advisable to name the file with the same name as the Java class. For example, I would create a file called `Helloworld.java`, and in it, I would create the class `Helloworld`.

2. The `main` method is the entry point to your Java program. When you compile a Java program using `javac`, the program may comprise more than one file (in other words, objects can interact with other objects that appear in independent files). However, when you run a program using `java`, the main method from that single file is executed—even if there are main methods in multiple files.

   So, as a rule of thumb, every Java file that you write will contain one or more `public` classes and exactly one `main` method will be executed.

Many Ways to Add Two Numbers

Let's build up some functionality piece by piece. Adding two numbers sounds like an essential building block for more complex programs, so we will do just that... with varying degrees of complexity.

Literals

First, you can add literal `Integer` values:

```
System.out.println(3 + 2);
```

We will call any literal number that is not stored in a variable a "magic number" (mostly becuase it is rarely clear why that number is chosen or what that number represents). We should avoid magic numbers *whenever possible*, and instead use variables with descriptive names. Since we are just adding two numbers, we'll use `x` and `y`, but a real program would make better choices.

Variables

You must *declare* all variables before using them, but as we see below you can combine declaring the variable (`int x;`) and assignment (`x = 3;`) into one line (`int x = 3;`)

```
int x = 3;
int y = 2;
System.out.println(x + y);
```

User input (command line arguments)

Sometimes we don't know the numbers ahead of time; we want to accept user input. Java passes all *command-line arguments* into the `main` method as `String` objects.

```
public static void main(String args[]) {
    x = args[0];
    y = args[1];
    System.out.println(x + y);
}
```

Unfortunately, if I compile (`javac Sum.java`) and run (`java Sum 3 2`), I don't get the results I expect.

```
  $ java Sum 3 2
  32
```

This is because java adds two `String` objects by concatenating them together. I must instead convert the `String` arguments into the appropriate type for my calculation: `Integer` .

```
public static void main(String args[]) {
    x = Integer.valueOf(args[0]);
    y = Integer.valueOf(args[1]);
    System.out.println(x + y);
}
```

There are other problems with this program. What if I don't pass in exactly 2 arguments? Passing in more than 2 numbers won't add them all, but it won't crash my program. Passing in more than 2 numbers will cause an `Exception` and crash my program. We can use `if` statements to protect from bad inputs:

```
public static void main(String args[]) {
    if (args.length != 2) {
        System.out.println("Please enter exactly 2 Integer arguments.");
        return;
    }
    x = Integer.valueOf(args[0]);
    y = Integer.valueOf(args[1]);
    System.out.println(x + y);
}
```

We can also accommodate different numbers of inputs using loops:

```
public static void main(String args[]) {
    int sum = 0;
    for (int = 0; i < args.length; i++) {
        sum = sum + Integer.valueOf(args[i]);
    }
    System.out.println(sum);
}
```

User input (interactive input with `java.util.Scanner`)

What if we don't know the arguments when we run the program, but we want to ask the user while the program is running? We can use a `Scanner` for that.

```
import java.util.Scanner;
public class Sum {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Please enter a number: ");
        int x = in.nextInt();
        System.out.print("Please give me another number: ");
        int y = in.nextInt();
        System.out.println( x + " + " + y + " = " + (x + y));
    }
}
```