Primitive Types and Math

Tim Magoun and Aravind Koneru

Compiled on Saturday 16th July, 2016 at 14:29

Abstract

In this lesson we will set up your computer for Java programming, and learn about some of the primitive types in Java. In the second half of this lesson, we will learn about the simple arithmetic operations that we'll use in the future.

Foreword

Programming is an ever-increasingly useful skill to have in the digital world. Before learning about the basics of Java, one must realize the following:

- Programming is the act of writing instructions for a computer
- The computer could only do one thing at a time
- Programming is supposed to make repetitive tasks easier
- It is more important to understand the concept rather than memorizing syntax

Feel free to ask questions, they don't have to be about the current exercise.

1 Installing Eclipse

Just follow the instructions on the course syllabus, which is also included below:

1. Go to `https://eclipse.org/downloads/eclipse-packages/`
2. Click on the corresponding installer, 32 bit or 64 bit (if you don't know the version of OS present, choose the 32 bit installer)
3. Download the installer to a known location (ex. Downloads or Desktop)
4. Execute the installer file
5. Select Eclipse IDE for Java Developers
6. Confirm install location and select preferred shortcut locations
7. Accept EULA
8. Bogosort the digits of π
9. Launch Eclipse Neon and set up preferences, line numbers are highly recommended

2 Creating a Java Project

1. Start up Eclipse and make the workspace in a known location (ex. Documents)
2. Enter into the Java Perspective
3. Right click on **Package Explorer**, which is on the left part of the screen, and select **New → Java Project**
4. Name the project **Lesson 1** and click **Finish**

3 The Primitive Types

Create a new class in your **Lesson 1** project by expanding the project, right click on **src**, and select **New → Class**. Name your class **PrimativeTypeExplorer** and select the check-box **public static void main String[] args)** under the last large text field, and 'Which method stubs would you like to create?'

Your screen should look somewhat like this:

Variables

What are variables? If you know some Algebra or Pre-Algebra, you know that we use variables in math to represent numbers in our equations. Essentially, a variable is an unknown value that we are trying to solve for. In programming, variables are also used to represent values.

In programming, we use variables all the time to store information that we would want to access later. There are various forms of information we could store. For example, we may want to store numbers, words, or even a variety of these things. The type of information we want to store is usually called the `type` of the variable. For example, the variable storing the phrase "sit vis vobiscum", would be of `type` string (in java, words are referenced as strings).

Some key words that you need to be aware of when discussing variables:

1. **Declaration**
2. **Initalization**
3. **type**

We have previous defined type, but we still need to define declaration and initialization.

A **declaration** is defined as the act of creating a new variable, but not assigning it a value. In Java, we declare variables as follows:

type name

Say that we wanted to make a variable to store an integer. We would do that by doing:

```
int anIntegerVariable;
```

In this case, we have **declared** a new variable named `anIntegerVariable` of **type** `int`.

Exercise: Look at the following variables and then break down them down like we did for `anIntegerVariable`.

```
    String myName;
    double pi;
    int aName;
    float someNum;
```

So far, we have only covered how to declare a variable. This means that all of our variables are set to `null` and we can't use them in our code. In order to assign values to our variables we must **intialize** them. We can initialize variables by setting them equal to some value. For example:

```
String myName;//declaring the variable
myName = "Aravind";//initializing the variable
```

We first declared the variable (`myName`) and then we assigned it the value "Aravind". This is an example of initalizing a variable. It is important to note that in java, variables accept their assignments from the right. This means that

```
String myName;
"Aravind" = myName;
```

will throw an error. This is the case for most programming languages you will encounter.

Exercise: Declare and initialize a variable for each of the following types:

```
int
float
double
String
```

It gets annoying to declare and initialize a variable on different lines. It's much easier to do both at the same time. We can easily do this by:

```
int myAge = 17 //declaring and initializing a variable at the same time
```

Note that it is common practice to declare and initialize variables on the same line. There are situations where it may not be possible to do both, but when you can, you should.

Now that we know about variables and how to use them, we can start doing some basic tasks in java. I'm a strong believer in application, so many of my example will come in the form of code. Here is the first example:

```
int num1 = 10;
int num2 = 20;
int num3 = num1 * num2;//What are we doing here?
int num4 = num3 / num1;//Is num4 always equal to num2?
```

In this example, we are introduced to something we haven't seen before: defining a variable in terms of other variables. In our example, `num3` is defined to be the product of `num1` and `num2`.Although we will always know the numerical values of `num1` and `num2`, we won't always know the numerical value of `num3`. Sure when `num1` and `num2` are small, we can calculate the value of `num3`, we won't be able to do so when `num1` and `num2` are large. The beauty of this situation is that we always know what the value of `num3` is in relation to that of `num1` and `num2`, but we may not always know the numerical value.

Exercise: Initialize two `String` variables with your first and last name. Make a third variable that will store your entire name, but you must define the third variable in terms of the first two.