Java Basics Unit

Lesson 01: Java Installation and Hello, World CLI- Notes

The Learning House
427 S 4th Street #300
Louisville KY 40202

Lesson 01: Java Installation and Hello, World CLI

Overview

Before we can write any Java code, we need to install the Java Development Kit (JDK). The JDK contains the compiler and other tools needed to create Java programs. We will be using Ubuntu Linux during the Java cohort, but this step includes directions for installing the JDK on Windows, Mac, and Linux so that you can install the JDK on your own machine no matter what operating system you are using. After the JDK is installed, we will check the configuration by creating, compiling, and running the obligatory "Hello, World!" program.

JDK Installation

Windows

Open a browser and go to the Java download site.

Click on the big download icon.

Accept the license agreement and click the link to the Windows x86 (for 32-bit Windows) or Windows x64 (for 64-bit Windows) to download the JDK. If you don't know which one to download, you probably need the x86 version. Once the .exe is downloaded, run the installer and follow the prompts (you will need Administrator access for this).

After installation is complete, you must add the directory in which the JDK executables reside (bin) to your PATH. Follow these steps to add the bin directory to your path:

1. Find the location of your newly-installed JDK. Write this down or copy it to the clipboard. It will most likely be in one of these locations:
   a. C:\Program Files\Java\jdk1.8.0\bin (or similar)
   b. C:\Program Files (x86)\Java\jdk1.8.0\bin (or similar)
2. Add this location to your PATH variable. To do this:
   a. Go to: Start → Computer → System Properties → Advanced System Setting → Environment Variables → System Variables → Path
   b. Now paste in the location of your JDK (as determined in Step 1 above) **in front of** whatever is already there. **You must follow your entry with a semicolon!** So you will add the following: **C:\Program Files\Java\jdk1.8.0\bin;** (replace this with the path to your JDK installation if different).
   c. Click OK three times to close all the dialog boxes.

To check that the installation is complete, open a command prompt (Go to Start → All Programs → Accessories → Command Prompt). Type the following command and hit enter:

```
javac -version
```

It should print **javac 1.8.0_u51** or something similar. If it prints an error message saying that 'javac' is not a recognized program, review the steps above to make sure your PATH variable is set correctly.

Mac OS X

OS X comes preinstalled with the JDK. To verify this, open the Terminal application and type:

```
javac -version
```

It should print **javac 1.8.0** or something similar. If the JDK is not installed, click on the Mac OS X x64 link on the Java download site to download the installer. Once downloaded, run the installer and follow the prompts.

Ubuntu Linux (works for any Debian variant)

To install the JDK on Ubuntu, open the terminal and type the following:

```
sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-java8-installer
```

You will be prompted for a password — type in your **login password**. After typing in your password, the installation should execute and finish successfully. To verify your installation, type the following into the terminal and hit enter:

```
javac -version
```

It should print out **javac 1.8.0**.

Hello, World! CLI

Now, we will create our first Java program. When learning a new language, it is customary that your first program simply print out "Hello, World!" to the screen. We'll do this following these steps:

1. Open a text editor (gedit will do)
2. Type in the Java code
3. Save the file as Hello.java
4. Compile the code with the java compiler (javac)
5. Run the program

Open your editor and type in the following code (don't worry about what all of it means, we'll get to that soon enough…):

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

Some things to notice about this file:

1. A Java program is **class** with a **main** method in it.
2. The **main** method is a special method that is the entry (or starting) point for every Java program. The **main** method must be declared exactly as in the example and it must be contained inside a **class**.
3. The curly braces { } denote the beginning and ending, respectively, of code blocks.
4. `System.out.println("Hello, World!");` is the magic code that sends things to the console (we'll learn much more about this later).

Save the file as "Hello.java". Now, open the terminal and cd into the directory containing Hello.java. We will now compile our program by typing:

```
javac Hello.java
```

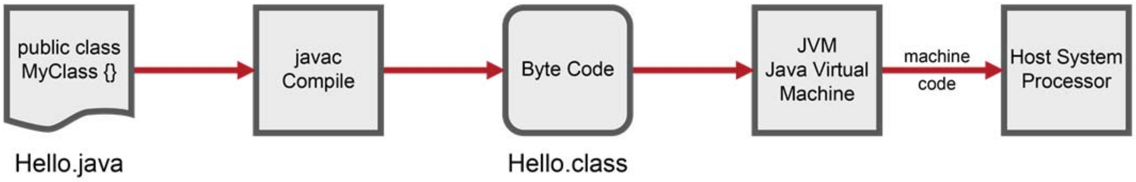
After our program is compiled, we can run it by typing:

```
java Hello
```

This will print "Hello, World!" to the screen.

The Compiler and Java Virtual Machine

Let's look at what is happening behind the scenes when we compile and run our program.



In the first step, we create our Java file: Hello.java. When we typed in 'javac Hello.java' in the second step, it invoked the Java **compiler**. The compiler converts the text that we typed in Hello.java into **bytecode**. Bytecode is a set of instructions that the **Java Virtual Machine (JVM)** understands. The JVM is basically a virtual computer that is implemented in software — it is what gives Java its ability to run on Linux, Windows, Mac OS X, BSD, and many other operating systems. Its job is to interpret the bytecode from your program and convert it into **machine code** that can run on the processor of your machine.

Wrap-up

Here is what we covered in this lesson:

1. We installed the Java Development Kit (JDK). The JDK contains the tools we need to create and run our own Java programs.
2. We created, compiled, and ran our first Java program called Hello.java. We learned:
    a. Java programs are made up of a **class** that contains a **main** method and that the **main** method is the special starting point for every Java program.
    b. Curly braces **{ }** mark the start and end of code blocks in Java.
    c. `System.out.println("Hello, World!");` is the magic code that sends text to the console to be displayed.
3. We learned about the **Java Virtual Machine** and the process that a Java file goes through from creation to execution.

In the next lesson, we will download, install, and configure the NetBeans **Integrated Development Environment (IDE)**. An IDE provides a complete environment for coding, compiling, running, and debugging Java applications. We will develop all of our code in the IDE from here on.