Java Introduction

Intro

In this exercise, you'll complete your installation of the software you need to write and run Java code. Then, you'll run your first Java program– a "Hello World" program.

The goal of this exercise is for you to be able to compile and execute Java code.

Part 1: Installation

Overview

1. Download JDK
2. Set Java HOME env variable
3. Update PATH to include dir so that you can use on command line
4. Verify success

Download JDK

- Visit jdk.java.net
  - Alternatively AWS provides an easy to install jdk known as Amazon Corretto.
- Click on largest version in "Ready for use"
- Click on your preferred OS (Mac, Windows, or Linux)

Windows Users

- Download the file and unzip into ***Program Files*** directory/folder
- Copy path to the unzipped program
- In Windows Settings app, search for env, and click ***Edit System Environment Variables***
- Under ***System variables*** click ***New***
- Variable name = `JAVA_HOME`, value = path you pasted to jdk unzip, then click ***OK***
- Update existing variable- `PATH`
  - Find PATH in list, click edit
  - Click ***New***
  - Paste path to jdk such as `C:\Program Files\jdk-17.0.1`
  - Add `\bin`

Mac Users

- Follow the instructions for Amazon Corretto.
- Follow the Mac + Linux to add the following to your ~./bash_profile

Linux Users

- Same general steps
- Put jdk download in ***/opt***
- Alternative– install using ***apt-get***
- If you chose to use Amazon Corretto there are instructions for the various distributions of linux available through Amazon Web Services documentation.

Mac + Linux

In ***~/.bash_profile***

```
export PATH=/Users/your-username/Library/Java/JavaVirtualMachines/jdk-16.jdk/bin:$PATH

export JAVA_HOME=/Users/your-username/Library/Java/JavaVirtualMachines/jdk-16.jdk/bin
```

Note: Linux users

Update the filepaths to be the correct location of the JDK download on your machine.

and then…

```
$ source .bash_profile
```

Verify Successful Install

```
$ java --version
```

Part 2: Hello World in Java

1. Create a new file (anywhere on your machine) called ***HelloWorld.java***. Include the following code. We'll learn more about what this code means later. For now, don't worry too much about the syntax, but do go ahead and read the code carefully to appreciate the high-level structure.

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

2. Compile your Java code!

Unlike Python, Java needs to compile before it can be executed. So, before we **run** Java, we need to **compile** Java. When Java code is compiled, the human-readable Java code is converted into **bytecode** which is optimized for and readable by computers.

Note: Does Python get compiled?

A lot of your knowledge about how code is evaluated at this point comes from your experience with Python. In order to compare/contrast Java with Python in terms of compilation, click here to read about whether Python is interpreted (not needing compilation) or compiled.

```
$ javac HelloWorld.java
```

After using `javac`, your new bytecode is in the new file ***HelloWorld.class***. Go ahead and run this command, and open up both files and compare them.

1. Run the following command, which executes the bytecode contained in the new file. When you do this, you should see a message get printed to the console.

```
$ java HelloWorld
```

What is the JDK?

- Stands for **Java Development Kit**
- Includes ***javac***, ***java*** commands
- Includes Java Virtual Machine (JVM)
- Includes Java Standard Edition (SE) APIs
  - `System.out.println("Hello world"");`
  - ***System*** is only available through SE APIs

Part 3: Your Second Program

Let's get some practice with the workflow of compiling and running Java code.

1. Create a new file called ***Color.java*** and include the following Java code in the file.

```
import java.util.Scanner;

class Color {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("What is your favorite color?");

    String color = input.nextLine();  // Get user input
    System.out.println("Nice, I like " + color + " too");
  }
}
```

2. Compile and run ***Color.java*** using the same steps that you used above.
3. Now that you have a sense of the utility of the ***Color*** program, go back to the code and read it carefully. Using the internet and also talking with your pairing partner, see if you can determine what each line of the program is going. What did you notice?

Hint: Read below to see a list of the important things to notice

▼ *Click to hide*

- There are type declarations before each new variable (`String`, `Scanner`, etc.)
- This is an object, and it has a lot of similarities to Python classes
- There are a bunch of words (`public static void`) before the name of the method (`main()`)
- In the parameter space, it seems you also have to declare the type of the parameters
- Lots of semicolons to finish all lines
- Lots of curly braces
- Import statements also look similar to Python
- String concatenation is similar to (one way to do it) in Python
- Many other cool things!

Scanner in Java

As you likely noticed, we are instantiating the ***Scanner*** class in order to obtain user input.

```
Scanner input = new Scanner(System.in);
```

To get a line of user input and save it in the variable, you call the method ***nextLine***, like so:

```
String name = input.nextLine();
```

Part 4: Mars Adventure Game

Believe it or not, you have the makings of a nifty text-based adventure game in Java at your disposal. Create a program that takes the user on a semi-interactive Mars Adventure. The goal is to ask the user at least 5 questions, and generate a response based on their input. While we don't know how to do if-statements, for loops, etc. yet, you can ask the user questions and issue responses that incorpoate the user's answers. Pretty cool!

Some ideas for questions:

- What is their name?
- What color will their spacesuit be?
- What is their pet's name?

Note: Build "Muscle Memory"

The goal here is to build some syntax muscle memory for Java code. While you might not understand every aspect of the syntax, typing out the pattern you saw in the ***Color*** program a few times is going to be useful in the future.

Make sure you compile and run your code often as you add new questions to your game. If you run into an error, see if you can figure it out with your debugging skills from Python!

© 2021 Devmountain