

## Lab 1: Intro to Java

### Part I: Java Basics (5 points)

In this part, we will do some java coding exercise in an online coding environment, i.e., LearnJavaOnline.org (<https://www.learnjavaonline.org/>). Specifically, you need to complete the exercise at the end of the three pages:

- [Hello, World!](#)
- [Variables and Types](#)
- [Conditionals](#)

(1 Extra Point)

- [Loops](#)

### Part II: Install JDK (5 points)

The Java Development Kit (JDK), officially named "Java Platform Standard Edition" or "Java SE", is needed for writing Java programs. The JDK is freely available from Sun Microsystems (now part of Oracle). The mother site for JDK (Java SE) is <http://www.oracle.com/technetwork/java/javase/overview/index.html>.

"JDK" or "JRE"?

JRE (Java Runtime) is needed for *running* Java programs. JDK (Java Development Kit), which includes JRE plus the development tools (such as compiler and debugger), is need for *writing* as well as *running* Java programs. In other words, JRE is a *subset* of JDK. Since you are supposed to write Java Programs, you should install JDK, which includes JRE.

**JDK Versions**

**Reference:** "Java Version History" @ [https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history).

1. JDK Alpha and Beta (1995): Sun Microsystem announced Java in September 23, 1995.
2. JDK 1.0 (January 1996): Originally called *Oak* (named after the oak tree outside James Gosling's office). Renamed to Java 1 in JDK 1.0.2.
3. JDK 1.1 (February 1997): Introduced AWT event model, inner class, JavaBean, JDBC, and RMI.
4. J2SE 1.2 (JDK 1.2) (December 1998): Re-branded as "Java 2" and renamed JDK to J2SE (Java 2 Standard Edition). Also released J2EE (Java 2 Enterprise Edition) and J2ME (Java 2 Micro Edition). Included JFC (Java Foundation Classes - Swing, Accessibility API, Java 2D, Pluggable Look & Feel, and Drag & Drop). Also introduced Collection Framework and JIT compiler.
5. J2SE 1.3 (JDK 1.3) (May 2000): Introduced Hotspot JVM.

6. J2SE 1.4 (JDK 1.4) (February 2002): Introduced `assert` statement, non-blocking IO (`nio`), logging API, image IO, Java webstart, regular expression (`regex`) support.
  7. J2SE 5.0 (JDK 5) (September 2004): Officially called 5.0 instead of 1.5 (by dropping the 1.). Introduced generics, autoboxing/unboxing, annotation, enum, varargs, for-each loop, static import. See "[JDK 5 New Features](#)".
  8. Java SE 6 (JDK 6) (December 2006): Renamed J2SE to Java SE (Java Platform Standard Edition). No new language features. See "[JDK 6 New Features](#)".
  9. Java SE 7 (JDK 7) (July 2011): First version after Oracle purchased Sun Microsystems - also called OracleJDK. Introduced Strings in `switch` statement, Binary integer literals, allowing underscores in numeric literals, improved type inference for generic instance creation (or diamond operator `<>`), Catching multiple exception types and rethrowing exceptions with improved type checking. See "[JDK 7 New Features](#)".
  10. Java SE 8 (JDK 8) (LTS) (March 2014): Included support for Lambda expressions, default and static methods in interfaces, improved collection, and JavaScript runtime. Also integrated JavaFX graphics subsystem. See "[JDK 8 New Features](#)".
  11. Java SE 9 (JDK 9) (September 21, 2017): Introduced modularization of the JDK (module) under project Jigsaw, the Java Shell (`jshell`), and more. See "[JDK 9 New Features](#)".
  12. Java SE 10 (18.3) (JDK 10) (March 2018): Introduced `var` for type inference local variable (similar to JavaScript). Introduced time-based release versioning with two releases each year, in March and September, denoted as `YY.M`. Removed native-header generation tool `javah`. See "[JDK 10 New Features](#)".
  13. Java SE 11 (18.9) (LTS) (JDK 11) (September 2018): Extended `var` to lambda expression. Standardize HTTP client in `java.net.http`. Support TLS 1.3. Clean up the JDK and the installation package (removed JavaFX, JavaEE, CORBA modules, deprecated Nashorn JavaScript engine). OracleJDK is no longer free for commercial use, but OpenJDK is still free. See "[JDK 11 New Features](#)".
  14. Java SE 12 (19.3) (JDK 12) (March 2019): Switch Expression (preview). See "[JDK 12 New Features](#)".
  15. Java SE 13 (19.9) (JDK 13) (September 2019): Switch Expression (preview), Multi-line Text Block (preview). See "[JDK 13 New Features](#)".
  16. Java SE 14 (20.3) (JDK 14) (March 2020): Records (preview). See "[JDK 14 New Features](#)".
  17. Java SE 16.0.2 is the latest release for the Java SE Platform, as of 08/2021.
- 

## 1. How To Install JDK on Windows

### Step 0: Un-Install Older Version(s) of JDK/JRE

It is recommended that you install only the *latest* JDK. Although you can install multiple versions of JDK/JRE concurrently, it is messy. If you have previously installed older version(s) of JDK/JRE, un-install ALL of them. Goto "Control Panel" ⇒ (optional) "Programs" ⇒ "Programs and Features" ⇒ Un-install ALL programs begin with "Java", such as "Java SE Development Kit ...", "Java SE Runtime ...", "Java X Update ...", and etc.

## Step 1: Download JDK

1. Go to Java SE download site @ <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
2. Under "Java Platform, Standard Edition" ⇒ "Java SE 14.0.{x}", where {x} denotes a fast running security-update number" ⇒ Click the "Oracle JDK Download" link.
3. Under "Java SE Development Kit 14.0. {x}" ⇒ Check "Accept License Agreement".
4. Choose the JDK for your operating system, i.e., "Windows". Download the "exe" installer (e.g., "jdk-14.0.{x}\_windows-x64\_bin.exe" - about 162MB).

## Step 2: Install JDK

Run the downloaded installer (e.g., "jdk-14.0.{x}\_windows-x64\_bin.exe"), which installs both the JDK and JRE.

By default, JDK is installed in directory "C:\Program Files\Java\jdk-14.0.{x}", where {x} denotes the update number. Accept the defaults and follow the screen instructions to install JDK.

Use your "File Explorer", navigate to "C:\Program Files\Java" to inspect the sub-directories. Take note of your **JDK installed directory** jdk-14.0.{x}, in particular, the update number {x}, which you will need in the next step.

Let's refer to the **JDK installed directory** as <JAVA\_HOME>, hereafter, in this article.

## Step 3: Include JDK's "bin" Directory in the PATH

Windows' Command Prompt (CMD) searches the current directory and the directories listed in the *PATH environment variable* (or *system variable*) for executable programs. JDK's programs (such as Java compiler "javac.exe" and Java runtime "java.exe") reside in the *sub-directory* "bin" of the JDK installed directory. You need to include JDK's "bin" in the *PATH* to run the JDK programs.

To edit the *PATH* environment variable in Windows 10:

1. Launch "Control Panel" ⇒ (Optional) "System and Security" ⇒ "System" ⇒ Click "Advanced system settings" on the left pane.
2. Switch to "Advanced" tab ⇒ Click "Environment Variables" button.
3. Under "System Variables" (the bottom pane), scroll down to select variable "Path" ⇒ Click "Edit...".
4. **For Newer Windows 10:**  
You shall see a **TABLE** listing all the existing *PATH* entries (if not, goto next step). Click "New" ⇒ Click "Browse" and navigate to your JDK's "bin" directory, i.e., "c:\Program Files\Java\jdk-14.0.{x}\bin", where {x} is your installation update number ⇒ Select "Move Up" to move this entry all the way to the TOP.
5. **For Older Windows 10** (Time to change your computer!):  
**(CAUTION: Read this paragraph 3 times before doing this step! Don't push**

**"Apply" or "OK" until you are 101% sure. There is no UNDO!!!)**  
(To be SAFE, copy the content of the "Variable value" to Notepad before changing it!!!)  
In "Variable value" field, APPEND "c:\Program Files\Java\jdk-14.0.{x}\bin"  
(where {x} is your installation update number) IN FRONT of all the existing directories,  
followed by a semi-colon (;) to separate the JDK's bin directory from the rest of the  
existing directories. DO NOT DELETE any existing entries; otherwise, some existing  
applications may not run.

Variable name : **PATH**  
Variable value : **c:\Program Files\Java\jdk-14.0.{x}\bin;***[do not delete  
existing entries...]*

Note: If you have started CMD, you need to re-start for the new environment settings to take effect.

#### Step 4: Verify the JDK Installation

Launch a CMD via one of the following means:

1. Click "Search" button ⇒ Type "cmd" ⇒ Choose "Command Prompt", or
2. Right-click "Start" button ⇒ run... ⇒ enter "cmd", or
3. Click "Start" button ⇒ Windows System ⇒ Command Prompt

Issue the following commands to verify your JDK installation:

1. Issue "path" command to list the contents of the PATH environment variable. Check to make sure that your JDK's "bin" is listed in the PATH.

**path**  
PATH=c:\Program Files\Java\jdk-14.0.{x}\bin;other entries...

2. Issue the following commands to verify that JDK/JRE are properly installed and display their version:

```
// Display the JDK version
javac -version
javac 14

// Display the JRE version
java -version
java version "14" 2020-03-17
Java(TM) SE Runtime Environment (build 14+36-1461)
Java HotSpot(TM) 64-Bit Server VM (build 14+36-1461, mixed mode,
sharing)
```

#### Step 5: Write a Hello-World Java Program

1. Create a directory to keep your works, e.g., "d:\myProject" or "c:\myProject". Do NOT save your works in "Desktop" or "Documents" as they are hard to locate. The

directory name shall not contain *blank* or special characters. Use meaningful but short name as it is easier to type.

2. Launch a programming text editor (such as Sublime Text, Atom, TextPad, NotePad++, etc.). Begin with a *new file* and enter the following source code. Save the file as "Hello.java", under your work directory (e.g., d:\myProject).

```
/*
 * First Java program to say Hello
 */
public class Hello {    // Save as "Hello.java" under "d:\myProject"
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

#### Step 6: Compile and Run the Hello-World Java Program

1. Start a CMD Shell (Search ⇒ enter "cmd" ⇒ select "Command Prompt").
2. Set the *Current Drive* to the drive where you saved your source file "Hello.java". If you use drive "c", skip this step. Else if you use drive "d", enter "d:" as follow:

```
d:
D:\xxx>
```

3. Set the *Current Working Directory* to the directory that you saved your source file via the **cd** (*Change Directory*) command. For example, suppose that your source file is saved in directory "myProject".

```
cd \myProject
D:\myProject>
```

4. Issue a **dir** (*List Directory*) command to confirm that your source file is present in the *current directory*.

```
dir
.....
xx-xxx-xx  xx:xx PM                277 Hello.java
.....
```

5. Invoke the JDK compiler "**javac**" to compile the source code "Hello.java".
6. **javac Hello.java**  
// If error message appears, correct your source code and re-compile

The compilation is successful if the command prompt returns. Otherwise, error messages would be shown. Correct the errors in your source file and re-compile.

7. The output of the compilation is a Java class called "Hello.class". Issue a **dir** (*List Directory*) command again to check for the output.

```

dir
.....
xx-xxx-xx  xx:xx PM                416 Hello.class
xx-xxx-xx  xx:xx PM                277 Hello.java
.....

```

To run the program, invoke the Java Runtime "java":

```

java Hello
Hello, world!

```

---

## 2. How to Install JDK on macOS

### Step 1: Check if JDK has been Pre-Installed

To check if JDK has been installed, open a "Terminal" (Search "Terminal"; or Finder ⇒ Go ⇒ Utilities ⇒ Terminal) and issue this command:

```
javac -version
```

- If a JDK version number is returned (e.g., JDK x.x.x), then JDK has already been installed. If the JDK version is prior to 1.8, proceed to Step 2 to install the latest JDK; otherwise, proceed to "Step 3: Write a Hello-world Java program".
- If message "command not found" appears, JDK is NOT installed. Proceed to the "Step 2: Install JDK".
- If message "To open javac, you need a Java runtime" appears, select "Install" and follow the instructions to install JDK. Then, proceed to "Step 3: Write a Hello-world Java program".

### Step 2: Download JDK

1. Goto Java SE download site @ <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
2. Under "Java Platform, Standard Edition" ⇒ "Java SE 14.0.{x}", where {x} denotes a fast running security-update number ⇒ Click the "Oracle JDK" "Download" button.
3. Under "Java SE Development Kit 14.0.{x}" ⇒ Check "Accept License Agreement".
4. Choose the JDK for your operating platform, i.e., macOS. Download the DMG installer (e.g, jdk-14.0.{x}\_osx-x64\_bin.dmg - about 176MB).

### Step 3: Install JDK/JRE

1. Double-click the downloaded Disk Image (DMG) file. Follow the screen instructions to install JDK/JRE.
2. Eject the DMG file.

3. To verify your installation, open a "Terminal" and issue these commands.

```
// Display the JDK version
javac -version
javac 14

// Display the JRE version
java -version
java version "14" 2020-03-17
.....

// Display the location of Java Compiler
which javac
/usr/bin/javac

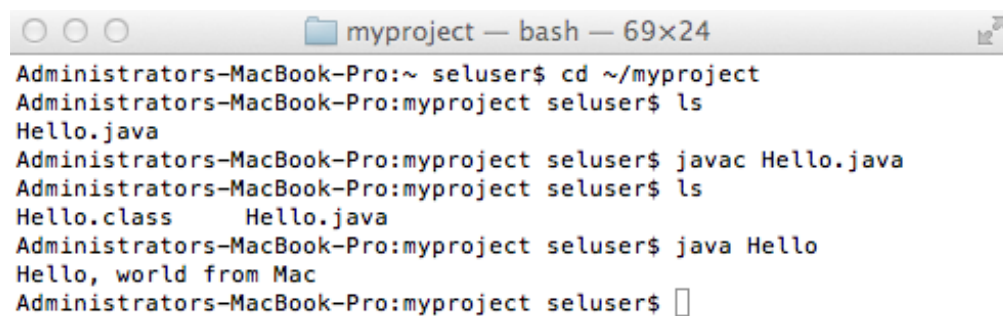
// Display the location of Java Runtime
which java
/usr/bin/java
```

### Step 3: Write a Hello-World Java Program

1. Create a directory called "myProject" under your *home* directory (Launch "Finder" ⇒ "Go" ⇒ "Home"; Select "File" ⇒ "New Folder" ⇒ "myProject").  
In macOS, the home directory of the current user can be referenced as "~". Hence, this new directory can be referenced as "~/myProject".
2. Use a programming text editor (such as Sublime Text or Atom) to input the following source code and save as "Hello.java" under the directory "~/myProject".  
(If you use macOS's default text editor "TextEdit" (NOT recommended), you need to open a new file ⇒ choose "Format" ⇒ "Make Plain Text" ⇒ Enter the source code ⇒ Save as "Hello.java".)

```
/*
 * My First Java program to say Hello
 */
public class Hello {    // Save as "Hello.java" under "~/myProject"
    public static void main(String[] args) {
        System.out.println("Hello, world from Mac!");
    }
}
```

### Step 4: Compile and Run the Hello-World Java Program



```
Administrators-MacBook-Pro:~ seluser$ cd ~/myproject
Administrators-MacBook-Pro:myproject seluser$ ls
Hello.java
Administrators-MacBook-Pro:myproject seluser$ javac Hello.java
Administrators-MacBook-Pro:myproject seluser$ ls
Hello.class  Hello.java
Administrators-MacBook-Pro:myproject seluser$ java Hello
Hello, world from Mac
Administrators-MacBook-Pro:myproject seluser$
```

1. To compile the source code "Hello.java", open a new "Terminal" ("Go" ⇒ "Utilities" ⇒ "Terminal") and issue these commands (as illustrated):

```
// Change Directory (cd) to where "Hello.java" resides
cd ~/myProject

// Check if "Hello.java" exists using list (ls) command
ls
Hello.java      .....

// Compile "Hello.java" using JDK compiler "javac"
javac Hello.java
// If error message appears, correct your source code and re-compile

// Check for the compiled output "Hello.class"
ls
Hello.class     Hello.java      .....
```

2. To run the Hello-world, invoke the Java Runtime "java" as follows:

```
java Hello
Hello, world from Mac!
```

---

### 3. How to Install JDK on Linux

There are several JDK implementations available for Linux, such as Oracle JDK, OpenJDK, Sun JDK, IBM JDK and GNU Java Compiler. We shall choose the Oracle JDK 8. Ubuntu chooses OpenJDK as its default JDK, which is not 100% compatible with Oracle JDK.

#### Step 0: Check if JDK has already been Installed

Open a Terminal and issue this command:

```
$ javac -version
```

If a JDK version number (e.g., "javac x.x.x") appears, JDK has already been installed. You can skip the installation and goto step 2.

To remove OpenJDK, issue command:

```
$ sudo apt-get purge openjdk-*
```

#### Step 1: Download and Install JDK

1. Goto JDK (Java SE) download site @ <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Under "Java Platform, Standard Edition" ⇒ "Java SE 11.0.{x}" ⇒ Click JDK's "Download" ⇒ Under "Java SE Development Kit 11.0.{x}" ⇒ Check "Accept License Agreement" ⇒ Select



"Linux", "tar.gz" package, (e.g., "jdk-13.0.{x}-linux-x64\_bin.tar.gz" - 171MB). The tarball will be downloaded in directory "~/Downloads", by default.

2. We shall install JDK under "/usr/local/java" (or Ubuntu's default JDK directory /usr/lib/jvm; or /opt/java). First, create a directory "java" under "/usr/local". Open a Terminal and issue these commands:

```
$ cd /usr/local
$ sudo mkdir java
```

Extract the downloaded package (Check your downloaded filename!)

```
$ cd /usr/local/java
$ sudo tar xzvf ~/Downloads/jdk-13.0.{x}-linux-x64_bin.tar.gz
    // x: extract, z: for unzipping gz, v: verbose, f: filename
```

JDK shall be extracted in a folder "/usr/local/java/jdk-13.0.{x}", where {x} is the update number.

3. Inform the linux to use this JDK/JRE:

```
// Setup the location of java, javac and javaws
$ sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/local/java/jdk-13.0.{x}/bin/java" 1
    // --install symlink name path priority
$ sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/local/java/jdk-13.0.{x}/bin/javac" 1
$ sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"/usr/local/java/jdk-13.0.{x}/bin/javaws" 1

// Use this Oracle JDK/JRE as the default
$ sudo update-alternatives --set java /usr/local/java/jdk-
13.0.{x}/bin/java
    // --set name path
$ sudo update-alternatives --set javac /usr/local/java/jdk-
13.0.{x}/bin/javac
$ sudo update-alternatives --set javaws /usr/local/java/jdk-
13.0.{x}/bin/javaws
```

The above steps set up symlinks java, javac, javaws at /usr/bin (which is in the PATH), that link to /etc/alternatives and then to JDK bin directory.

The "alternatives" system aims to resolve the situation where several programs fulfilling the same function (e.g., different version of JDKs). It sets up symlinks thru /etc/alternatives to refer to the actual programs to be used.

```
$ ls -ld /usr/bin/java*
lrwxrwxrwx 1 root root xx xxx xx xx:xx /usr/bin/java ->
/etc/alternatives/java
lrwxrwxrwx 1 root root xx xxx xx xx:xx /usr/bin/javac ->
/etc/alternatives/javac
lrwxrwxrwx 1 root root xx xxx xx xx:xx /usr/bin/javaws ->
/etc/alternatives/javaws
```

```
$ ls -ld /etc/alternatives/java*
lrwxrwxrwx 1 root root xx xxx xx xx:xx /etc/alternatives/java ->
/usr/local/java/jdk-13.0.{x}/bin/java
lrwxrwxrwx 1 root root xx xxx xx xx:xx /etc/alternatives/javac ->
/usr/local/java/jdk-13.0.{x}/bin/javac
lrwxrwxrwx 1 root root xx xxx xx xx:xx /etc/alternatives/javaws ->
/usr/local/java/jdk-13.0.{x}/bin/javaws
```

Alternatively, you can include the JDK's bin and JRE's bin into the PATH directly.

4. To verify the JDK installation, issue these commands:

```
// Show the Java Compiler (javac) version
$ javac -version
javac 11.0.{x}

// Show the Java Runtime (java) version
$ java -version
java version "11.0.{x}"
.....

// Show the location of javac and java
$ which javac
/usr/bin/javac

$ which java
/usr/bin/java
```

5. [Don't Do this step - It is taken care by "alternative" in Step 3. Keep here to show you how to set PATH.]

Add JDK's binary directory ("bin") to the "PATH" by editing "/etc/profile":

```
$ cd /etc
$ gksudo gedit profile // OR "sudo nano profile" to use the console-
based nano editor
```

Add these lines at the *end* of the file "/etc/profile", replace "{x}" with the actual number:

```
export JAVA_HOME=/usr/local/java/jdk-13.0.{x}
export PATH=$JAVA_HOME/bin:$PATH
```

Rerun the configuration file by:

```
// Refresh
$ source /etc/profile

// Check the new settings for JAVA_HOME and PATH
$ echo $JAVA_HOME
/usr/local/java/jdk-13.0.{x}

$ echo $PATH
```

```
.....:/usr/local/java/jdk-13.0.{x}/bin
```

## Step 2: Compile and Run a Hello-world Java Program

1. File Explorer ⇒ Home ⇒ Create a new folder called "myProject" to keep our works.
2. Open "Text Editor" (gedit). Enter the following source code and save as "Hello.java" under the "~/myProject" directory created earlier.

```
public class Hello {    // To save as "Hello.java" under "~/myProject"
    public static void main(String[] args) {
        System.out.println("Hello, world from Ubuntu!");
    }
}
```

3. To compile the Hello-world Java program, launch a Terminal and issue these commands:

```
// Change directory to where the source code resides
$ cd ~/myProject

// List the contents of current directory. Check for "Hello.java"
$ ls
..... Hello.java .....

// Compile "Hello.java" into "Hello.class"
$ javac Hello.java

// Check for "Hello.class"
$ ls
..... Hello.class .....
```

4. Run the Hello-world Java program:

```
// Run "Hello.class"
$ java Hello
Hello, world from Ubuntu!
```