Installing and Running Your First Java Program

Estimated time needed: 20 minutes

In this lab you will learn how to install and set up the environment to run Java programs in your system, and also create you first Java program.

You are currently viewing this lab in a Cloud-based Integrated Development Environment (Cloud IDE). It is a fully online, integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

Learning Objectives

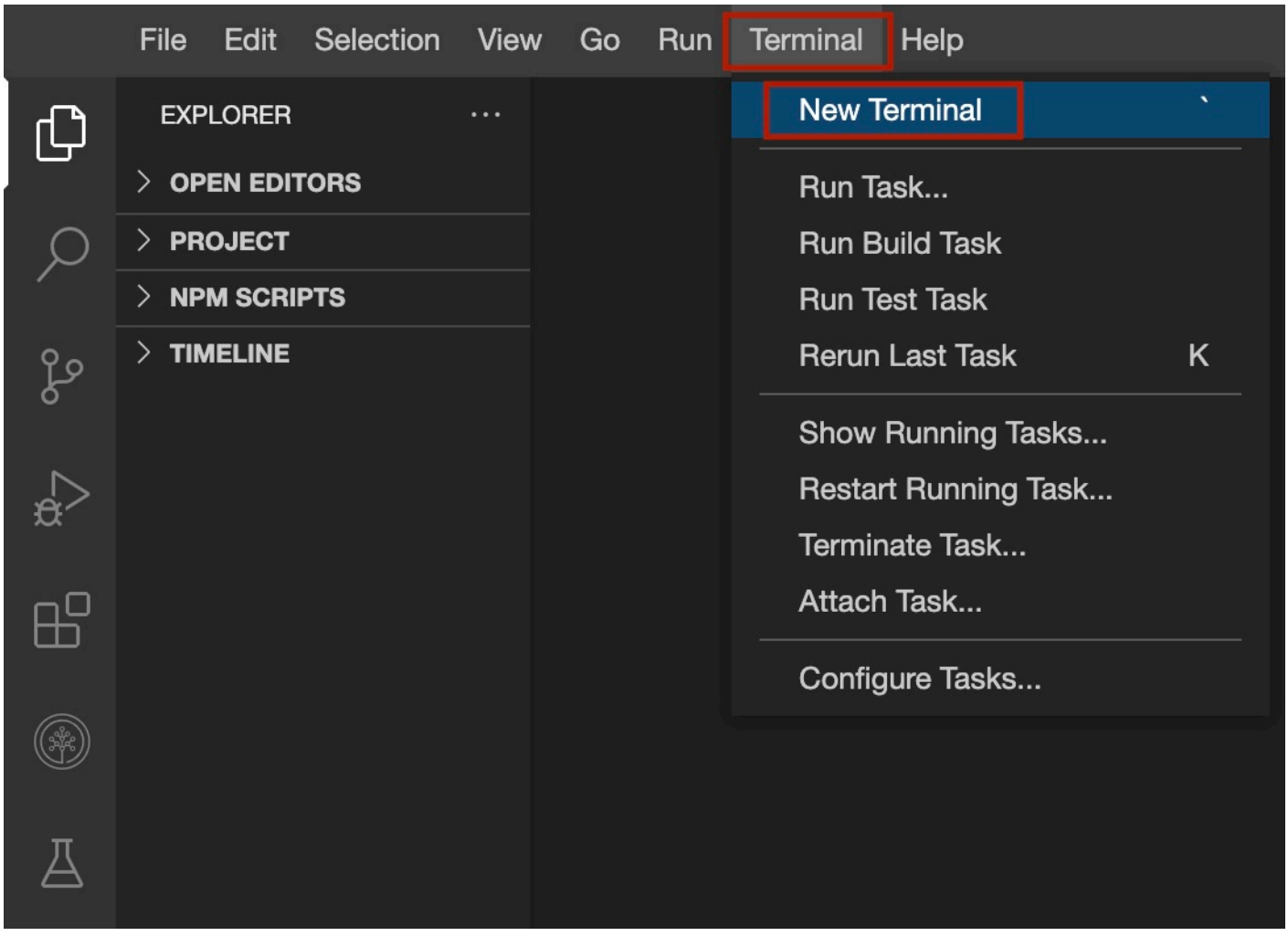
After completing this lab you will be able to:

- Set the path for Java executables in your system
- Write a Java program to print statement onto the command line
- Set the classpath for your system to look for the classes
- Run a Java program from the command line

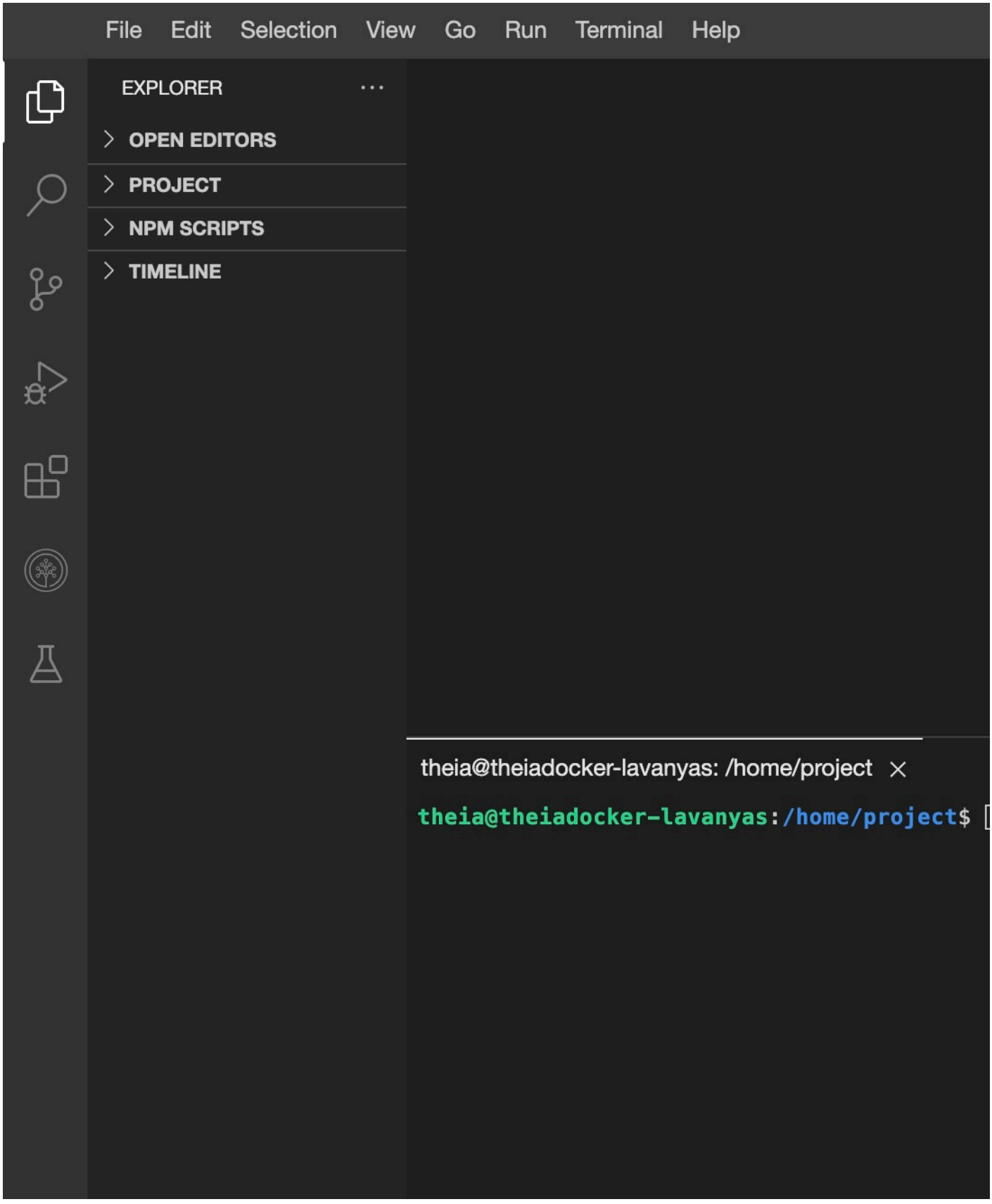
Check for Java installation

To compile and run a Java program, Java should be installed in the system. In the Cloud IDE, Java has been pre-installed for you. For your further practice after completion of the course, it is highly recommended that you install JDK in your system. The guide to installation is covered in a later part of the lab.

1. Click on `Terminal` on the Cloud IDE menu and choose `New Terminal` to open a new terminal.



2. You will see the new terminal as shown in the image, ready to take your commands.



3. Check the version of Java that has been provided in the Cloud IDE.

```
java --version
```

You should see an output such as:

```
java --version
java 21.0.5 2024-10-15 LTS
Java(TM) SE Runtime Environment (build 21.0.5+9-LTS-239)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.5+9-LTS-239, mixed mode, sharing)
```

4. Check JAVA_HOME (recommended).The JAVA_HOME environment variable points to the installation directory of the Java Development Kit (JDK). Many applications, IDEs (Integrated Development Environments), and build tools rely on this variable to locate the Java runtime environment (JRE) and development tools.

   ```
   echo $JAVA_HOME
   ```

You should see an output such as:

```
/usr/lib/jvm/jdk-21.0.5-oracle-x64
```

Create the first java program

In this part of the lab, you will create, compile, and run your first Java program. The first time you are creating a Java program, you can create it in any location. You will now create a Java program in the current directory and run it. A good practice is to keep the source file and the compile class files separate. In the next part of the lab, you will create the same, adhering to the directory structure.

1. At the command prompt, run the following command to create you first Java program.

   ```
   touch MyFirstProgram.java
   ```

2. Click on the button below to open the file that you just created so you can edit it.

Open **MyFirstProgram.java** in IDE

3. Paste the following code in the file.

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

Here's a step-by-step explanation of the Java code:

Class Declaration

```
public class MyFirstProgram {
```

- `public`: is called the `Access modifier` that defines the access level for the class. In this case, the class can be accessed from anywhere as it is `public`.
- `class`: Keyword to declare a new class.
- `MyFirstProgram`: Name of the class. You will notice that the classname is the same as the filename. This is mandatory in Java.

Main Method Declaration

```
public static void main(String s[]) {
```

- public: is called the `Access modifier` that defines the access level for the method. This doesn't have to be the same as the class. In this case, the main method within the class can be accessed from anywhere as it is `public`.
- static: Keyword indicating the method belongs to the class, but you don't have to create an object of the class to access it. As the word suggests, the method doesn't dynamically change for each instance of the class you create. It stays static or unchanged.
- void: Return type, indicating the method doesn't return any value.
- main: Name of the method. the `public static void main` method is the entry point for a Java program.
- (String s[]): Method parameter, an array of strings, which represents the command-line arguments.

Print Statement

```
System.out.println("Hello World!");
```

- System.out: A built-in Java object that represents the standard output stream which is usually the console.
- println: A method of the PrintStream class that prints its argument to the standard output stream, followed by a newline character.
- "Hello World!": The string to be printed.
- All the statements in Java should be completed with a semi-colon at the end, except when you have brackets such as `{` or `}`.

Closing Brackets

```
    }
}
```

- Closing bracket for the main method.
- Closing bracket for the MyFirstProgram class.
- Maintaining the indentation is a good coding practice as it improves the readability of the code.

Compiling the Java program

1. To compile a Java program, run the `javac` command followed by the name of the file you need to compile.

   ```
   javac MyFirstProgram.java
   ```

2. If all the steps were done correctly, this command will not produce any console output. It will create `MyFirstProgram.class` in the current directory. The file is in unreadable bytecode format.

```
ls *.class
```

You should see `MyFirstProgram.class` listed.

3. Run the class by executing the following command.

```
java MyFirstProgram
```

Please note that you don't have to specify the `.class` extension. You just need to specify the name of the class and the system by default will look for the class in the current directory.

The output would appear as given below.

```
Hello World!
```

4. Now change the code to print `Welcome to Java Programming!` in addition to `Hello World!`. Add the following code below the previous print statement.

```
System.out.println("Welcome to Java Programming!");
```

▶ Click here to view the code

5. Compile the updated program. The Java program needs to be recompiled every time a change is made, and the old `.class` file will be overwritten with the updated one.

```
javac MyFirstProgram.java
```

6. Run the class by executing the following command.

```
java MyFirstProgram
```

The output would appear as given below.

```
Hello World!
Welcome to Java Programming!
```

Creating a project and organising in a directory

Though what you have started off with is a simple program, Java is used for most advanced projects and becomes manageable when the code is organized within a project structure. It is a good habit to start with that right at the outset.

1. Create a project directory by running the following command.

   ```
   mkdir my_first_proj
   ```

2. Within the project directory, you segragate the source file, the output files, and the test files when you eventually start creating those. Run the following code to create the directory structure.

   ```
   mkdir -p my_first_proj/src
   mkdir -p my_first_proj/classes
   mkdir -p my_first_proj/test
   cd my_first_proj
   ```

3. Now move the `MyFirstProgram.java` from the default directory where you created it, to the project's `src` directory.

   ```
   mv /home/project/MyFirstProgram.java src
   ```

4. Remove the old class file that was created by compiling the Java code.

   ```
   rm /home/project/MyFirstProgram.class
   ```

5. Compile the Java program, this time explicitly specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/MyFirstProgram.java
   ```

6. Try to run the Java program as done before.

```
java MyFirstProgram
```

This will throw ClassNotFoundException.

This is because the class file is not in the default directory. You need to tell the system where to look for the class file. The path where `java` looks for classes is called the classpath.

7. Set the `CLASSPATH` variable.

```
export CLASSPATH=$CLASSPATH:/home/project/my_first_proj/classes
```

8. Now when you run the Java program, it will run seamlessly as expected.

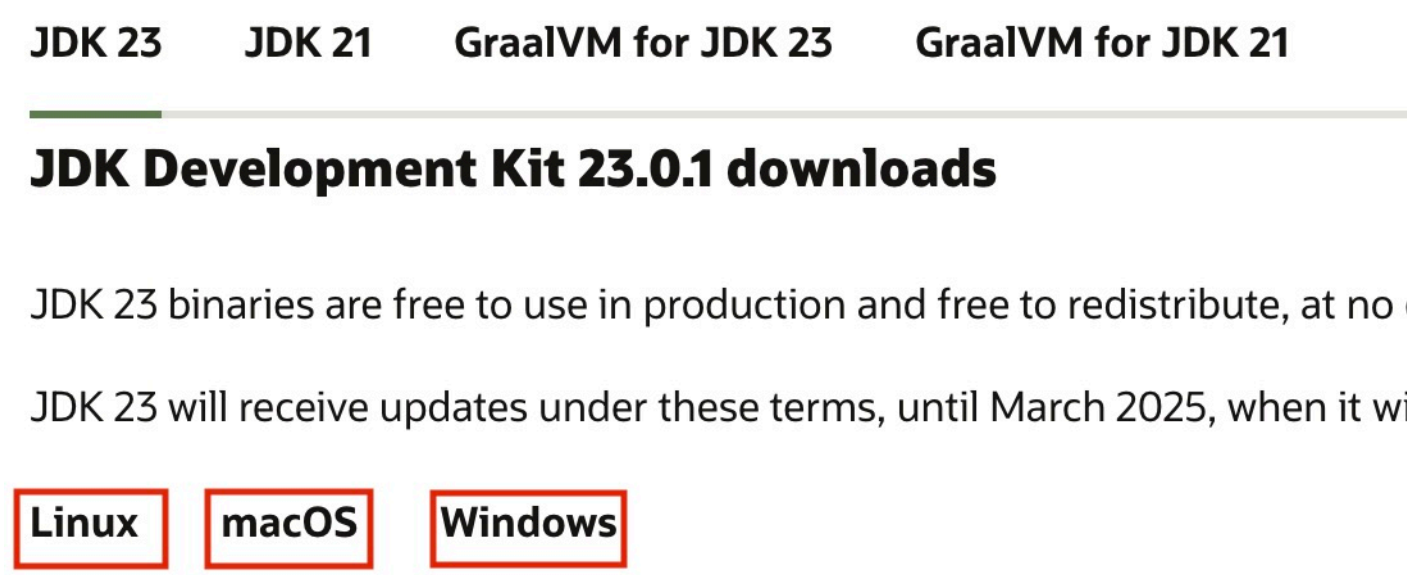
```
java MyFirstProgram
```

Practice Exercise

1. Create a Java project directory named `practice_proj` with src and classes directory in it.
2. Create a file in the `src` folder that will print the message `Now I can write a Java program on my own!.`
3. Compile it in such a way that the class file is output in the classes directory.
4. Set the classapath and run the file.

Installing Java on your system

1. Go to https://www.oracle.com/au/java/technologies/downloads/ to find the latest JDK downloads. At the time of creating this lab, the latest is version 23. From March 2025, it will be superseded by version 24.



2. Choose the operating system of you computer and choose the installable that is appropriate to your system.

3. Follow the JDKinstallation instructions provided in this link.

4. Once you complete the installation, open the `Command Prompt`, follow the steps below depending on which OS you are using.

    - On Windows:
        - Click on the `Windows` button on the keyboard and typing `cmd` and press `Enter`.
        - When the command prompt opens, type `java -version` to see if Java has been installed and ensure the right version is displayed.
    - On macOs:
        - Press `command` and space bar and type `Terminal` in the spotlight.
        - In the terminal, type `java -version`. If the installation has been done successfully, you will see the java version displayed.

```
(base) lavanyas-mbp-2:~ lavanyas$ java --version
java 23.0.1 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, s
```

5. Set the JAVA_HOME environment variable to point to the JDK installation directory.

- Windows

    1. Right click on the windows key on the keyboard and choose `Settings`.
    2. In the settings window search for `Environment Variables`. You will see the option to `Edit environment variables for your account`. Choose that option, especially if you are using a work PC or more than one user uses the PC where it is being installed.
    3. Under `User Variables`, click on the New button.
    4. In the `Variable name` field, enter JAVA_HOME.
    5. In the `Variable value` field, enter the path to your JDK installation directory (e.g., C:\Program Files\Java\jdk-23). Please ensure that the path you set matches the installation location on your system.
    6. Click OK to close all the windows.

- macOS (via Terminal)

    1. Open the Terminal application.
    2. Run the command export JAVA_HOME=$(/usr/libexec/java_home). Please ensure that the path you set matches the installation location on your system. This will set the JAVA_HOME for that Terminal instance only for as long as the terminal is open.
    3. If you want to permanently set JAVA_HOME, search and open `.bashrc` file and add the following statement.

        ```
        export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home
        ```

    Please ensure that the path you set matches the installation location on your system.

- Linux (via Terminal)

    1. Open the Terminal application.
    2. Run the command export JAVA_HOME=/usr/lib/jvm/java-23-openjdk-amd64. Please ensure that the path you set matches the installation location on your system.

6. Create and run the Java program.

Conclusion

In this lab, you learned how to create Java program inside an organized program structure and run it.

Author(s)

Lavanya