

*Lab Exercise for Week 03*

# JAVA DEVELOPMENT SETUP AND BASIC PROGRAMMING

---

*BITP 3113 Object-Oriented Programming*

*Author*

*Emaliana Kasmuri*

*Fakulti Teknologi Maklumat dan Komunikasi*

*Universiti Teknikal Malaysia Melaka*

## Table of Contents

Learning Outcomes .....	1
Tools .....	1
Supporting Materials .....	1
Preparing Java Development Environment .....	2
Exercise 1: Verify JDK Installation .....	2
Exercise 2: Download JDK .....	3
Exercise 3: Install JDK .....	4
Exercise 4: Set Environment Variable (for Windows) .....	5
Exercise 5: Test JDK Installation .....	8
Basic Java Application .....	9
Exercise 6: Preparing Lab Exercise Environment .....	10
Exercise 7: Execute a Java Program .....	10
Exercise 8: Observe the Java Program .....	12
Java Class Name and File .....	13
Exercise 9: Producing Compilation Error .....	13
Exercise 10: Debugging the Compilation Error .....	15
The main() Method .....	15
Exercise 11: Executing a Text Processing App .....	16
Java Executable Statement .....	16
Exercise 12: Executing a Date Manipulation App .....	16
Exercise 13: Executing Other Applications .....	17

## Learning Outcomes

At the end of this lab exercise, the student should be able to:-

1. Install JDK.
2. Compile Java files/classes.
3. Execute Java applications.

## Tools

The exercise for this lab session is using the following tools:-

1. Java SDK
2. Notepad or TextEdit or any text editor
3. Windows Command Prompt or MacOS Terminal

## Supporting Materials

The reference files and supporting materials are available on the ulearn course site.

## Preparing Java Development Environment

The JDK is crucial for Java development as it provides the java compiler for bytecode generation, the JRE for runtime, essential development tools, and extensive libraries. It's the comprehensive toolkit needed to write, compile, and debug Java applications.

### Exercise 1: Verify JDK Installation

1. Switch on the computer.
2. Open **Command Prompt** (for Windows) or **Terminal** (for MacOS) from the computer.
3. Then, type the `java -version` command on the command prompt/terminal. The output shall be similar as shown in Figure 1.

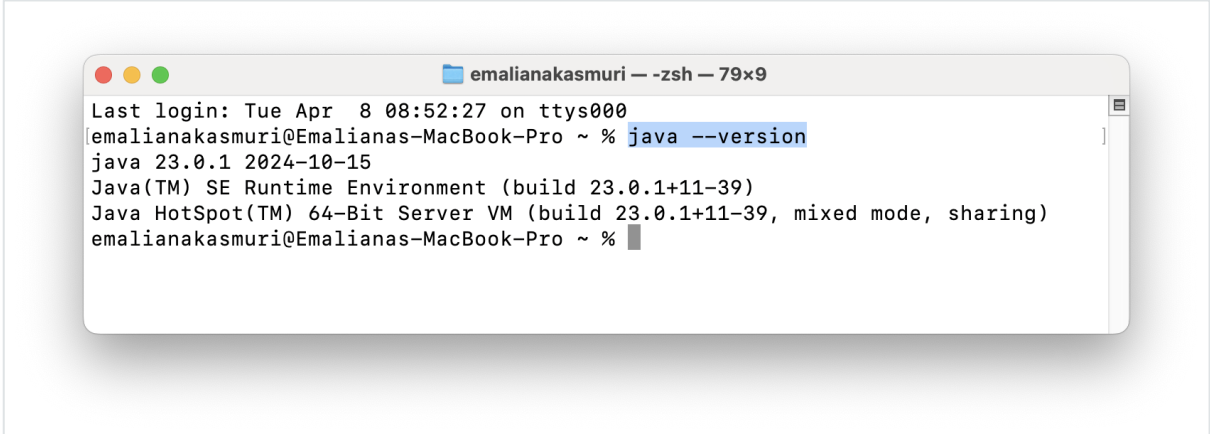
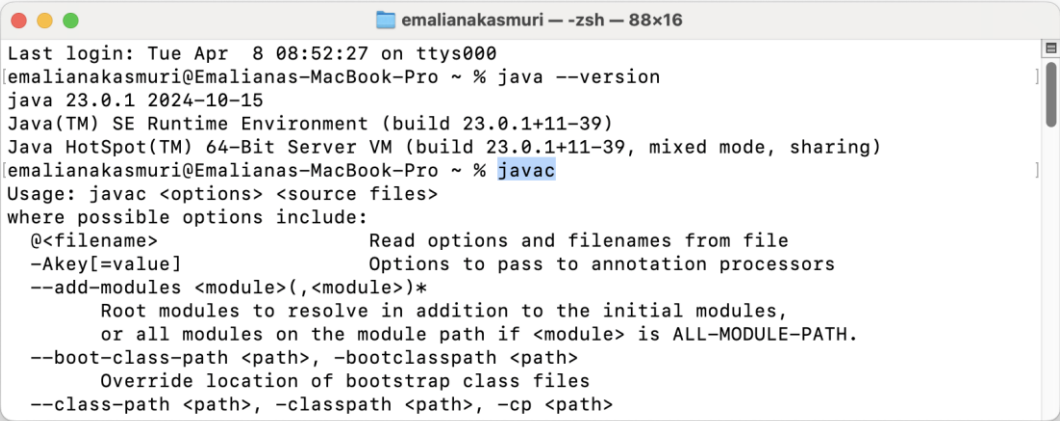
A screenshot of a macOS Terminal window. The title bar shows the user 'emalianakasmuri' and the shell '-zsh' with a window size of '79x9'. The terminal text shows the last login time as 'Tue Apr 8 08:52:27 on ttys000'. The user enters the command 'java --version' (the command is highlighted in blue in the original image). The output is: 'java 23.0.1 2024-10-15', 'Java(TM) SE Runtime Environment (build 23.0.1+11-39)', and 'Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)'. The prompt returns to 'emalianakasmuri@Emalianas-MacBook-Pro ~ %'.

Figure 1: Results from the `java --version` command

- After that, type the `java -version` command on the terminal. The output shall be similar as shown in Figure 2.



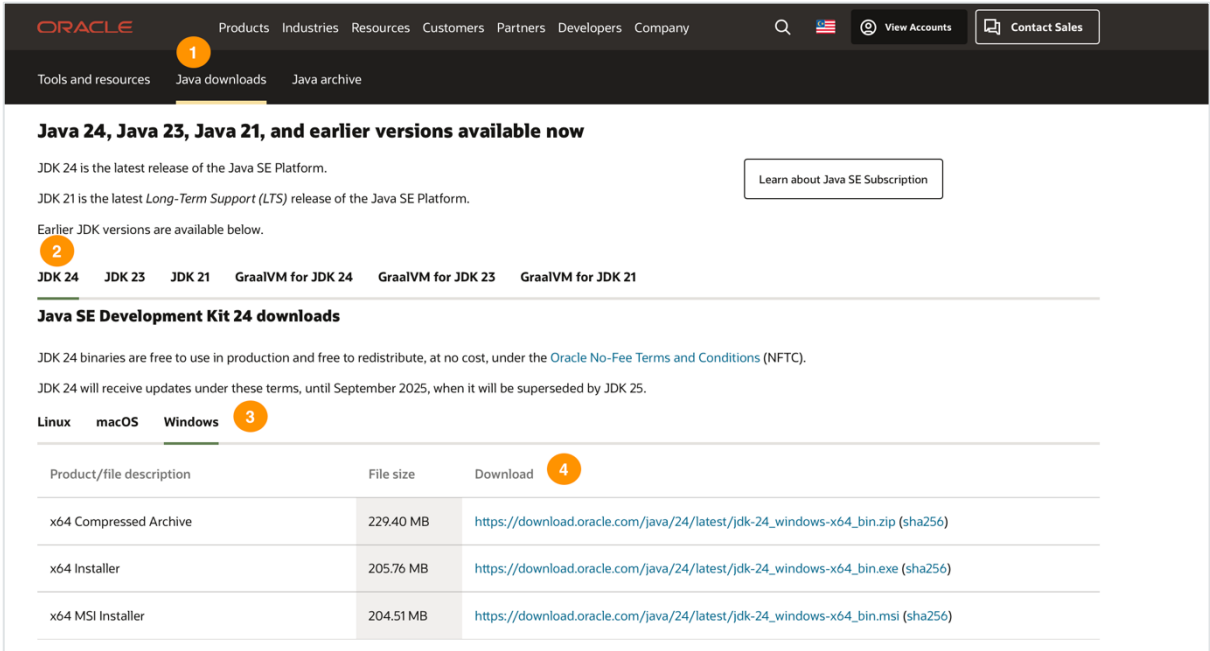
```
emalianakasmuri — -zsh — 88x16
Last login: Tue Apr  8 08:52:27 on ttys000
emalianakasmuri@Emalianas-MacBook-Pro ~ % java --version
java 23.0.1 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)
emalianakasmuri@Emalianas-MacBook-Pro ~ % javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>           Read options and filenames from file
  -Akey[=value]         Options to pass to annotation processors
  --add-modules <module>(<module>)*
                        Root modules to resolve in addition to the initial modules,
                        or all modules on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
                        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
```

Figure 2: Results from `javac` command

- The results indicate the JDK has been installed in the computer. Proceed to [Exercise 6](#).
- Proceed to [Exercise 2](#) if the result is not similar as shown in Figure 1 and/or Figure 2.

## Exercise 2: Download JDK

- Open <https://www.oracle.com/my/java/technologies/downloads/> in a browser. It will display a web page as shown in Figure 3.



The screenshot shows the Oracle Java Downloads page for JDK 24. The page is titled "Java 24, Java 23, Java 21, and earlier versions available now". It includes a "Learn about Java SE Subscription" button. Below the title, it states "JDK 24 is the latest release of the Java SE Platform." and "JDK 21 is the latest Long-Term Support (LTS) release of the Java SE Platform." It also mentions "Earlier JDK versions are available below." and lists "JDK 24", "JDK 23", "JDK 21", "GraalVM for JDK 24", "GraalVM for JDK 23", and "GraalVM for JDK 21". The "JDK 24" link is highlighted with a red circle labeled "2". Below this, it says "Java SE Development Kit 24 downloads" and "JDK 24 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC)." It also states "JDK 24 will receive updates under these terms, until September 2025, when it will be superseded by JDK 25." The "Linux" tab is selected, and the "Windows" tab is highlighted with a red circle labeled "3". Below the tabs, there is a table with columns "Product/file description", "File size", and "Download". The "Download" column has a red circle labeled "4" next to the first row. The table lists three download options for Windows: "x64 Compressed Archive" (229.40 MB), "x64 Installer" (205.76 MB), and "x64 MSI Installer" (204.51 MB).

Product/file description	File size	Download
x64 Compressed Archive	229.40 MB	<a href="https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.zip">https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.zip</a> (sha256)
x64 Installer	205.76 MB	<a href="https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.exe">https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.exe</a> (sha256)
x64 MSI Installer	204.51 MB	<a href="https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.msi">https://download.oracle.com/java/24/latest/jdk-24_windows-x64_bin.msi</a> (sha256)

Figure 3: Webpage to download the Java SDK

2. Figure 3 are marked with numbers to indicate the important part for the downloads. Select **Java downloads** tab (no 1) from the page.
3. Then, select **JDK 24** tab (no 2).
4. After that, select the operating system of the computer (no 3).
5. Finally, click the link from the **Download** column (no 4) to download the JDK. The file will be downloaded into the computer. It might take a while depending on the Internet connection speed.

### Exercise 3: Install JDK

1. The JDK should be downloaded into the computer's Download folder. It should be similar as shown in Figure 4. Double-click the file.

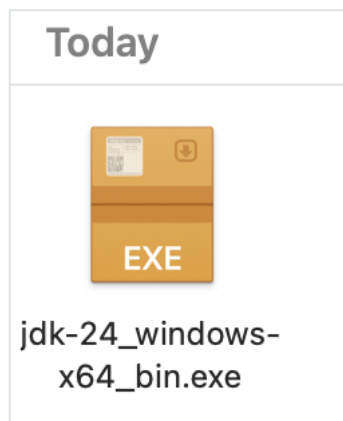


Figure 4: Downloaded JDK

2. Follow the instruction on the screen until the end.
3. Observe the installation path. It is needed for the next exercise.

## Exercise 4: Set Environment Variable (for Windows)

Commonly, Java is installed in **C:\Program Files\Java** (might be otherwise for some computers). The following are step to configure JDK using Windows environment variable.

1. Copy the Java installation path.
2. Click **Windows** menu from the bottom toolbar.
3. Type **Environment variable** from the search box as shown in Figure 5.

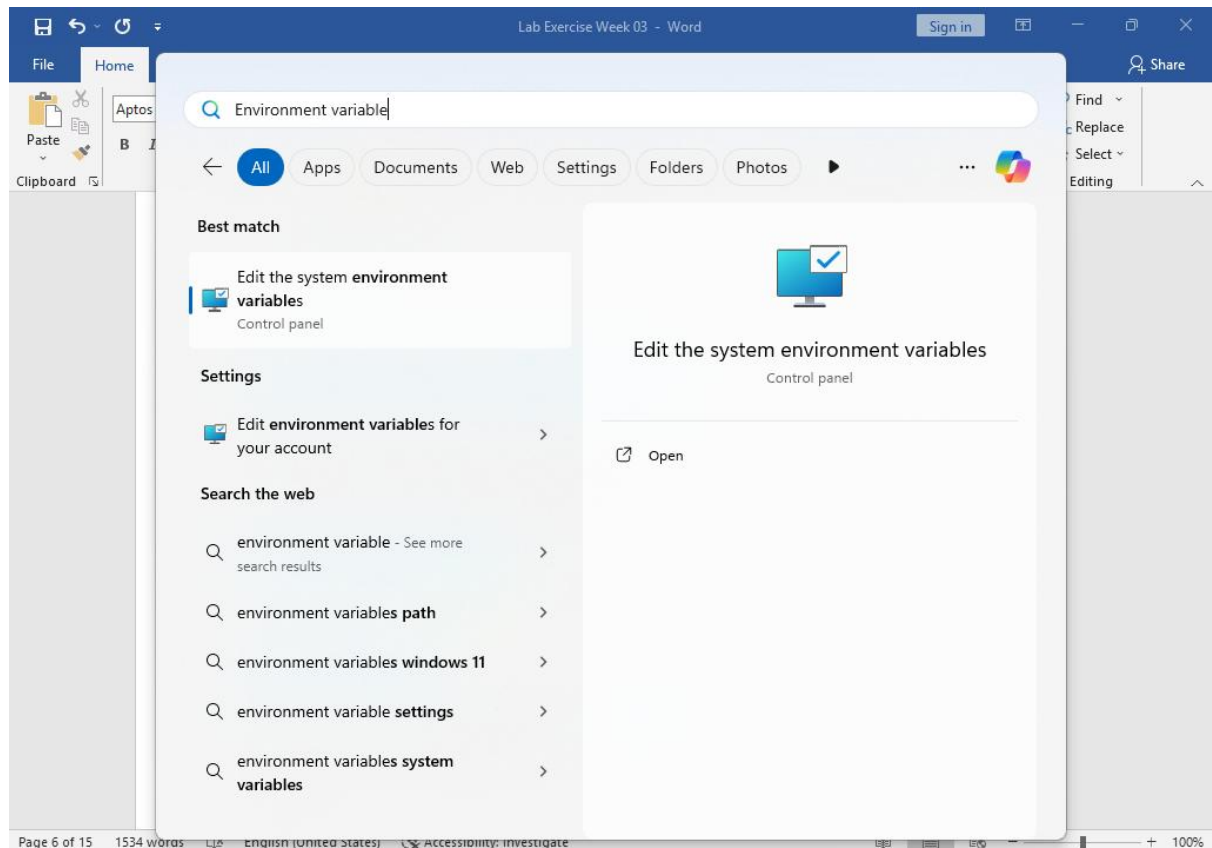


Figure 5: Result of environment variable

4. The result shall not be off as shown in Figure 5. Click **Edit the system environment variables** from the result. A window named System Properties as shown in Figure 6 will be displayed.

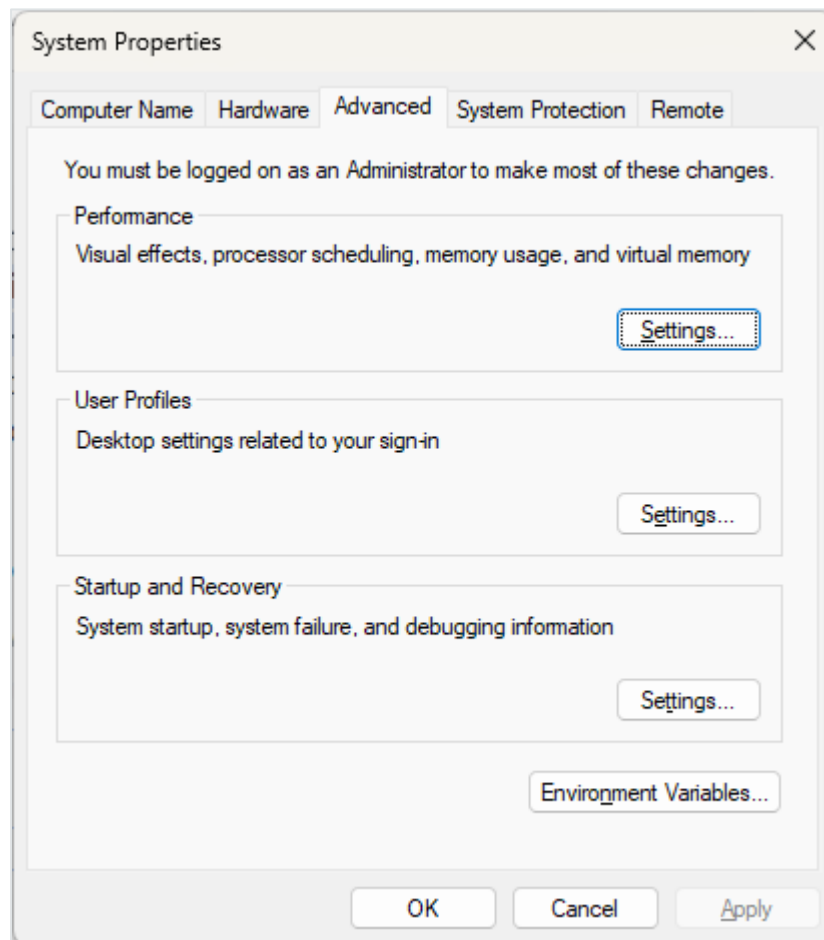


Figure 6: System Properties window



5. Click the **Environment Variables** button from the window. A window named Environment Variables as shown in Figure 7 will be displayed.

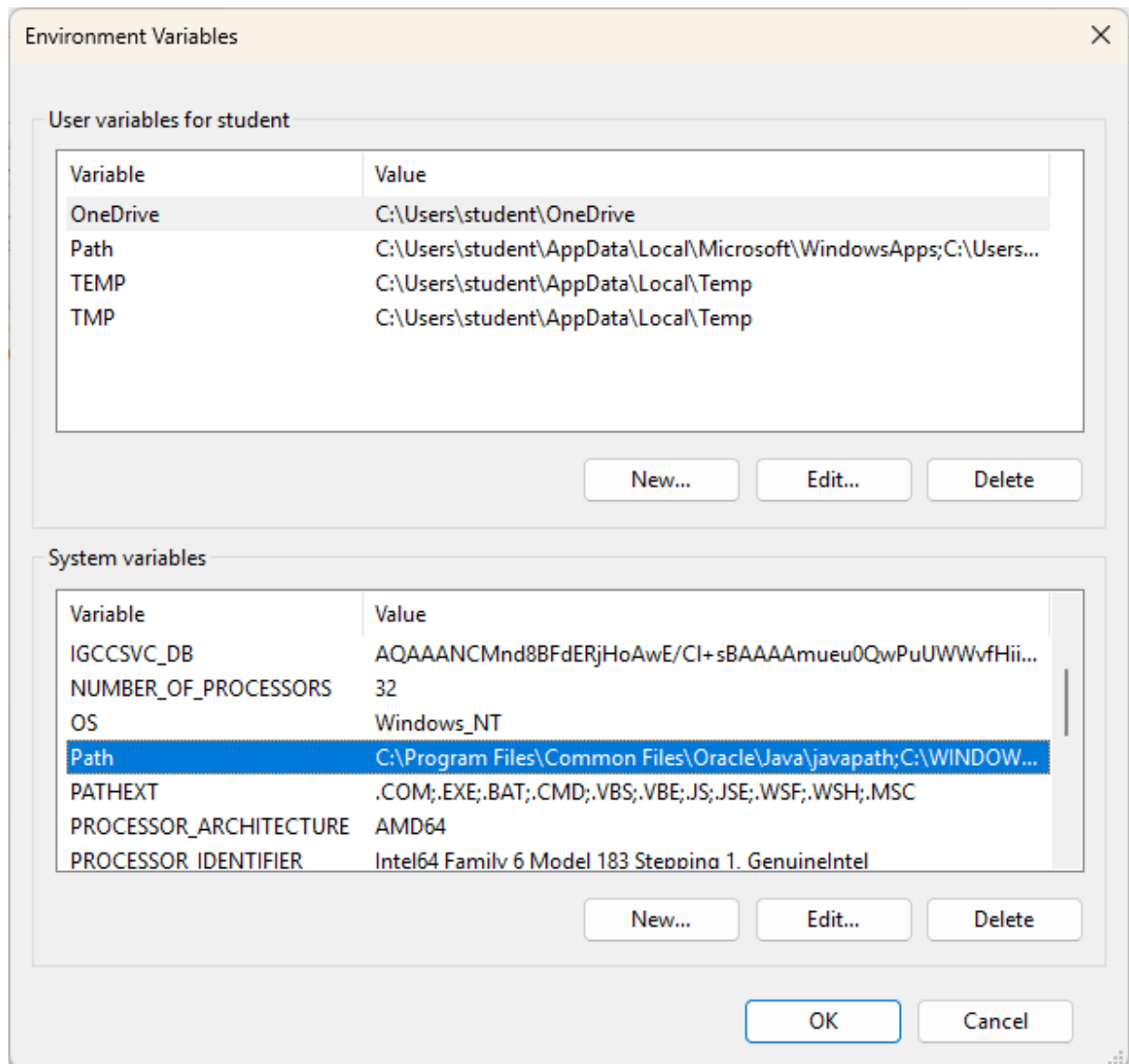


Figure 7: Environment variables window

6. After that, select the **Path** variable from the **System variables** panel.

7. Then, click the **Edit** button from the window. A window named Edit environment variable as shown in Figure 8 will be displayed.

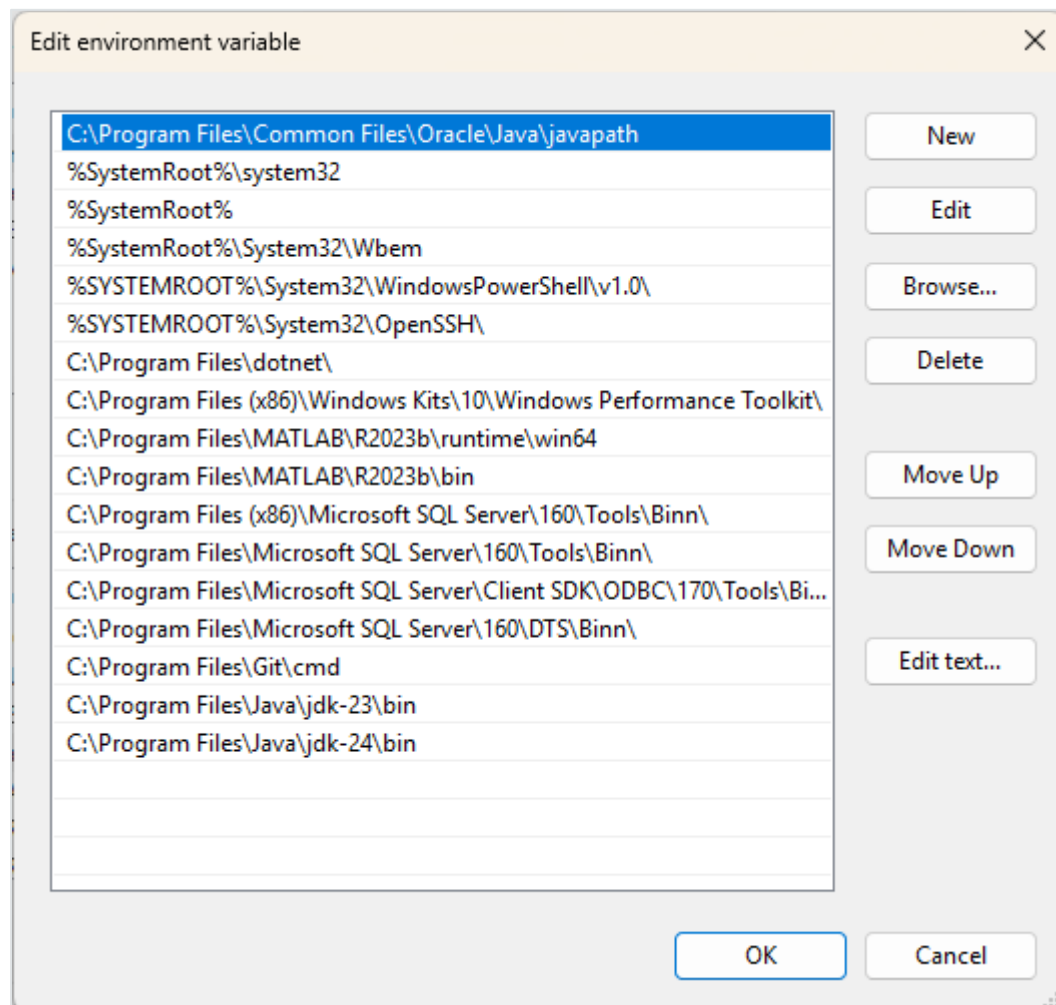


Figure 8: Edit environment variable window

Figure 8 shows some Java path in the environment variables list. Proceed to the next step to add new JDK into the environment variables.

8. Click the **New** button to add a new environment variable.
9. A new entry will be created in the list. Paste the path copied from step no 1 of this exercise.
10. Then, click the **OK** button.
11. Click all OK buttons to close the setting.

### Exercise 5: Test JDK Installation

1. Repeat the steps in [Exercise 1](#) to test the installation.

## Basic Java Application

A basic Java application is made of a class with a `main()` method, as shown in Figure 9. The program is saved as **GreetingApp.java**, retaining the same name as the class with **.java** extension.

```
1
2  /**
3   * BITP 3113 Object Oriented Programming
4   *
5   * This class demonstrate the first Java application to be compiled
6   * and execute.
7   *
8   * @author Emaliana Kasmuri, FTMK, UTeM
9   */
10 public class GreetingApp {
11
12     /**
13      * The main entry point to the application.
14      *
15      * @param args
16      */
17     public static void main (String args[]) {
18
19
20         // Display a greeting message
21         System.out.println(x:"Greetings from GreetingApp");
22     }
23 }
24
```

Figure 9: GreetingApp class with the `main()` method

The class should be compiled and execute using Java command on console. The java command to compile the application is `javac` followed by the file name. For example, `javac GreetingApp.java`. While the `java` command followed by the class name is to execute the application. For example, `java Greeting`.

## Exercise 6: Preparing Lab Exercise Environment

1. Create a folder named **bitp3113** on your computer. Preferably in the **C:** or **D:** drive.
2. Prefix the folder name with your matric number. For example, P0316160003-bitp3113.
3. Create a subfolder named **labweek03**. The structure should be similar as shown in Figure 10.

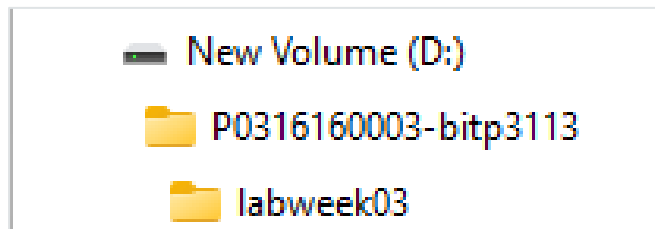


Figure 10: Lab exercise folder structure

This folder will store all Java codes related to lab exercises in week 03.

## Exercise 7: Execute a Java Program

1. Download the **GreetingApp.java** from ulearn.
2. Move the file into folder named **labweek03**.
3. Open **MS Prompt** (for Windows) or **Terminal** (for MacOS) from the computer.
4. Change the directory to **labweek03** using the `cd` command. The outcome should be similar as shown in Figure 11.

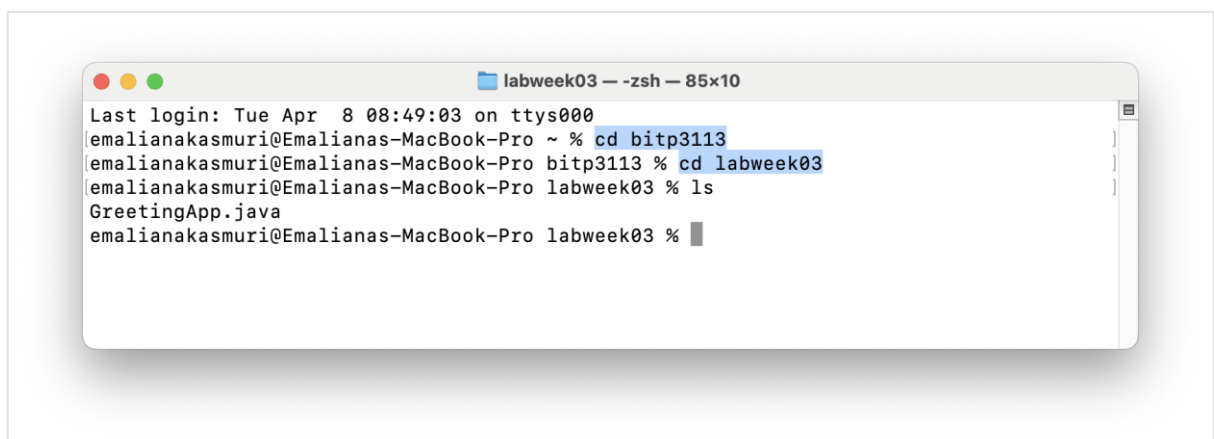
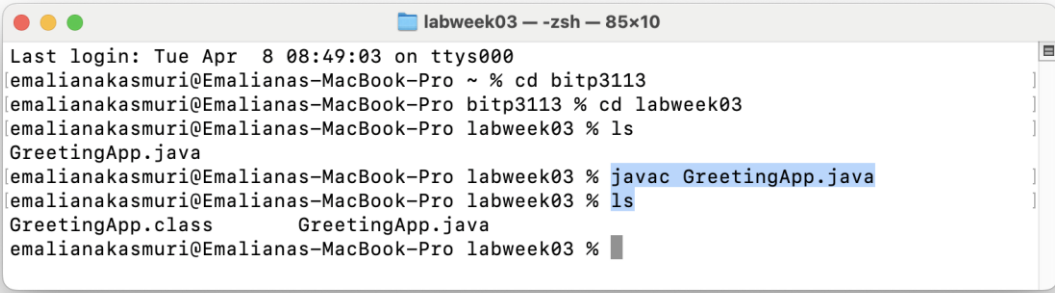


Figure 11: The `cd` command in Terminal

5. Type the `javac GreetingApp.java` command on the terminal to compile the Java class.

6. Then, type the `ls` command on the terminal to view list of files in the directory. The outcome should be similar as shown in Figure 12.

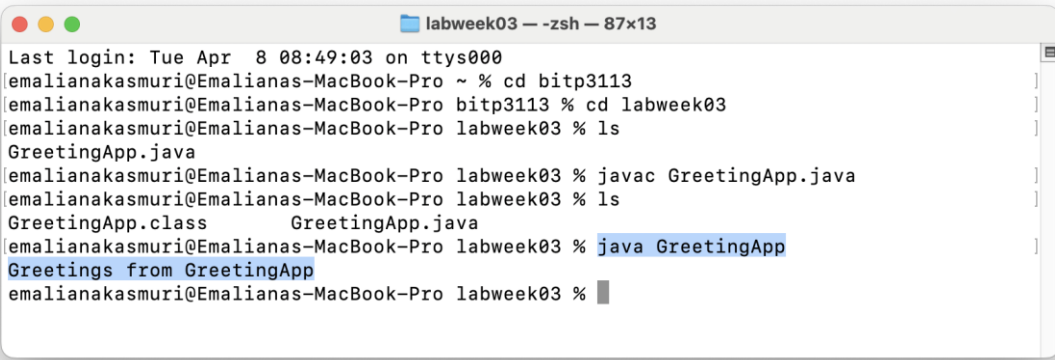
A terminal window titled 'labweek03 - zsh - 85x10' showing the following commands and output:

```
Last login: Tue Apr 8 08:49:03 on ttys000
emalianakasmuri@Emalianas-MacBook-Pro ~ % cd bitp3113
emalianakasmuri@Emalianas-MacBook-Pro bitp3113 % cd labweek03
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % ls
GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % javac GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % ls
GreetingApp.class      GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 %
```

Figure 12: Outcome from `javac` command on `GreetingApp.java`

The `.class` file existence indicate the Java file is successfully compiled.

7. After that, type the `java Greeting` command in the terminal. The outcome should be similar as shown in Figure 13.

A terminal window titled 'labweek03 - zsh - 87x13' showing the following commands and output:

```
Last login: Tue Apr 8 08:49:03 on ttys000
emalianakasmuri@Emalianas-MacBook-Pro ~ % cd bitp3113
emalianakasmuri@Emalianas-MacBook-Pro bitp3113 % cd labweek03
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % ls
GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % javac GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % ls
GreetingApp.class      GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % java GreetingApp
Greetings from GreetingApp
emalianakasmuri@Emalianas-MacBook-Pro labweek03 %
```

Figure 13: The outcome from `java` command on `GreetingApp`

The greeting statement from the **GreetingApp** marks your first successful Java application execution. Congratulation!

## Exercise 8: Observe the Java Program

Java used the same syntax as C/C++. The program is written using a combination of Java keywords and method calling. The following are the steps to observe a program.

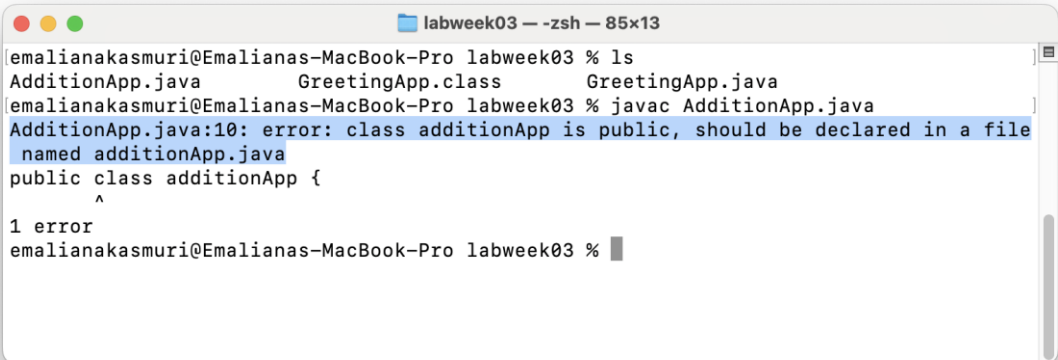
1. Open **GreetingApp.java** in Notepad or TextEditor.
2. Turn on the line number from the editor.
3. Locate the following keyword from the source code.
  - a. `public`
  - b. `class`
  - c. `static`
  - d. `void`
  - e. `String`
  - f. `main`
4. Identify the curly brackets `{ }` for the class and the `main()` method.
5. Identify the method to display the greeting message on the console.
6. Identify the comment block.
7. Observe the class and the file name.
8. Get yourself familiar with the source code and the programming style.

## Java Class Name and File

The Java class name must be the same as file name. The class must be saved with `.java` extension, as shown in the previous example. A good class name shall always start with an upper-case letter, for example `GreetingApp`.

### Exercise 9: Producing Compilation Error

1. Download a file named **AdditionApp.java**.
2. Move the file into folder named **labweek03**.
3. Compile the file using `javac` command. The outcome shall be similar as shown in Figure 14.



```
labweek03 — -zsh — 85x13
emalianakasmuri@Emalias-MacBook-Pro labweek03 % ls
AdditionApp.java      GreetingApp.class    GreetingApp.java
emalianakasmuri@Emalias-MacBook-Pro labweek03 % javac AdditionApp.java
AdditionApp.java:10: error: class additionApp is public, should be declared in a file
named additionApp.java
public class additionApp {
      ^
1 error
emalianakasmuri@Emalias-MacBook-Pro labweek03 %
```

Figure 14: Outcome from AdditionApp.java compilation

4. The compilation shall produce 1 error message as highlighted in Figure 14. Observe the error message anatomy in Figure 15.

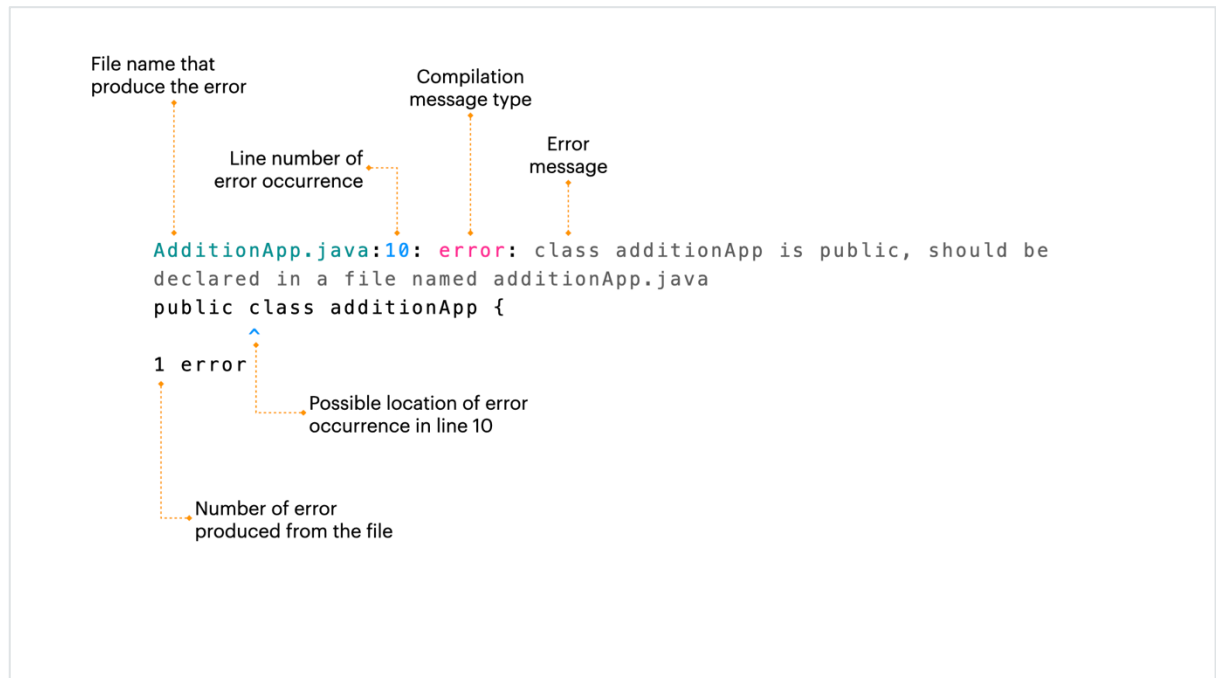
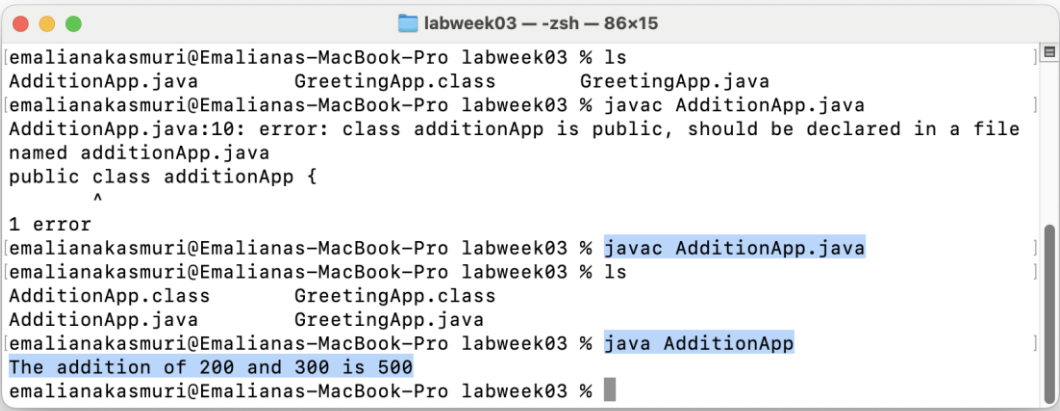


Figure 15: Anatomy of Java compilation error



## Exercise 10: Debugging the Compilation Error

1. Open **AdditionApp.java** using the previous text editor.
2. Turn on the line number.
3. Bring the cursor the line where the error has occurred.
4. Apply the knowledge comprehended on Java class and file name to fix the error.
5. Save the file.
6. Compile the file.
7. Execute the application. The outcome shall be similar as shown in Figure 16.



```
labweek03 — zsh — 86x15
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % ls
AdditionApp.java      GreetingApp.class    GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % javac AdditionApp.java
AdditionApp.java:10: error: class additionApp is public, should be declared in a file
named additionApp.java
public class additionApp {
      ^
1 error
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % javac AdditionApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % ls
AdditionApp.class      GreetingApp.class
AdditionApp.java       GreetingApp.java
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % java AdditionApp
The addition of 200 and 300 is 500
emalianakasmuri@Emalianas-MacBook-Pro labweek03 %
```

Figure 16: Outcome from AdditionApp execution

## The main() Method

The `main( )` method is the entry point to any executed Java application. The method is declared within a Java class. The method is declared using a combination of reserved word – `public`, `static`, `void` and `main`, as shown in Figure 17. It may receive none or any number of execution variables.

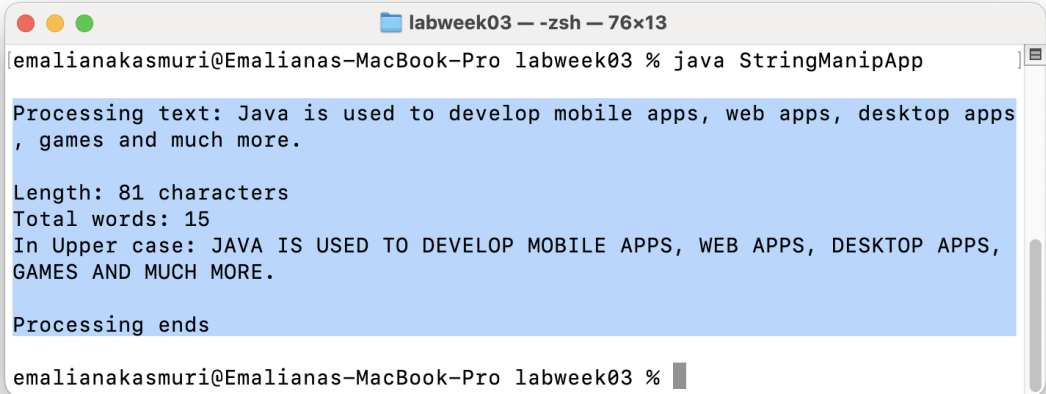
```
public static void main (String args[]) {

    // Display a greeting message
    System.out.println(x:"Greetings from GreetingApp");
}
```

Figure 17: The `main( )` method definition

### Exercise 11: Executing a Text Processing App

1. Download **StringManip** file from ulearn.
2. Move the file into folder named **labweek03**.
3. Compile the file.
4. Using the knowledge comprehended until this point, fix the errors produced from the file.
5. Execute the application only when the compilation is free from any error. The application shall produce an output similar as shown in Figure 18.



```
labweek03 — -zsh — 76x13
emalianakasmuri@Emalianas-MacBook-Pro labweek03 % java StringManipApp

Processing text: Java is used to develop mobile apps, web apps, desktop apps
, games and much more.

Length: 81 characters
Total words: 15
In Upper case: JAVA IS USED TO DEVELOP MOBILE APPS, WEB APPS, DESKTOP APPS,
GAMES AND MUCH MORE.

Processing ends

emalianakasmuri@Emalianas-MacBook-Pro labweek03 %
```


Figure 18: Output from StringManipApp execution

### Java Executable Statement

Each Java executable statement must be terminated with a semi-colon `;`. Most Java statements are written within the class or method block. A block is marked with curly brackets.

### Exercise 12: Executing a Date Manipulation App

1. Download **DateFormattingApp.java** file from ulearn.
2. Move the file into folder named **labweek03**.
3. Compile the file.
4. Using the knowledge comprehended until this point, fix the errors produced from the file.
5. Execute the application only when the compilation is free from any error. The application shall produce an output similar as shown in Figure 19.



```
labweek03 -- zsh -- 99x17

Current Date and Time: 08-Apr-2025 10:35:48

Extracted Details
Year: 2025
Month: 4
Day: 8
Day of week : TUESDAY

Date manipulation
Yesterday: 07-Apr-2025 10:35:48
Two Week from now: 22-Apr-2025 10:35:48
Three hours ago: 08-Apr-2025 07:35:48

Program ends

emalianakasmuri@Emalianas-MacBook-Pro labweek03 %
```

Figure 19: Output from DateManipulationApp execution

### Exercise 13: Executing Other Applications

1. Download other the following files from ulearn.
  - a. DataListerApp.java
  - b. AverageCalculator.java
  - c. TextDemoApp.java
  - d. Product.java
  - e. CurrentDateApp.class
2. Move the file into folder named **labweek03**.
3. Using the comprehended knowledge, compile the files.
4. Fix any errors produced from the compilation.
5. Execute the application.
6. One of the files is not executable even though the file passed compilation phase. Using the comprehended knowledge, record your analysis on this failure in ulearn.
7. One of the files is executable even without **.java**. Using the comprehended knowledge, record your analysis on this success in ulearn.

End of Document

