A close-up of logos

Description automatically generated

**BITP 3113**

**OBJECT ORIENTED PROGRAMMING**

**LAB:**

**WEEK 03 – USING JAVA**

**LECTURER NAME:**

**DR EMALIANA BINTI KASMURI**

**STUDENT NAME:**

**MUHAMMAD AFIQ MUHAIMIN BIN MOHD ZAINI**

**MATRIC NUMBER:**

**B032410648**

**SEMESTER: 2-2024/25**

## 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](#_bookmark20).

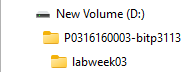


Figure 10: Lab exercise folder structure

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](#_bookmark22).

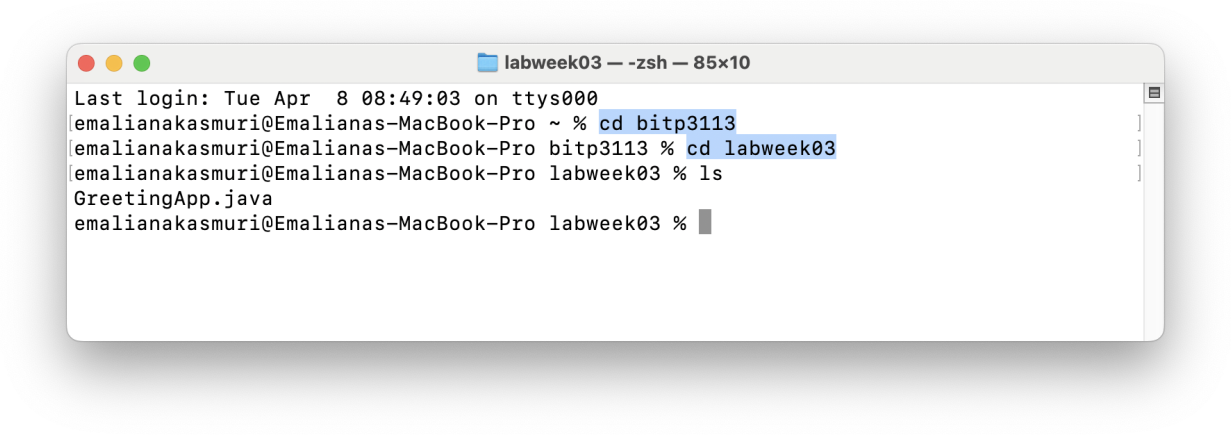
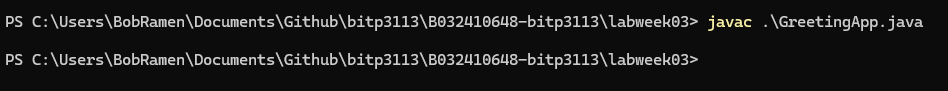


Figure 11: The cd command in Terminal

A screenshot of a computer screen

AI-generated content may be incorrect.

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



1. 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](#_bookmark23).

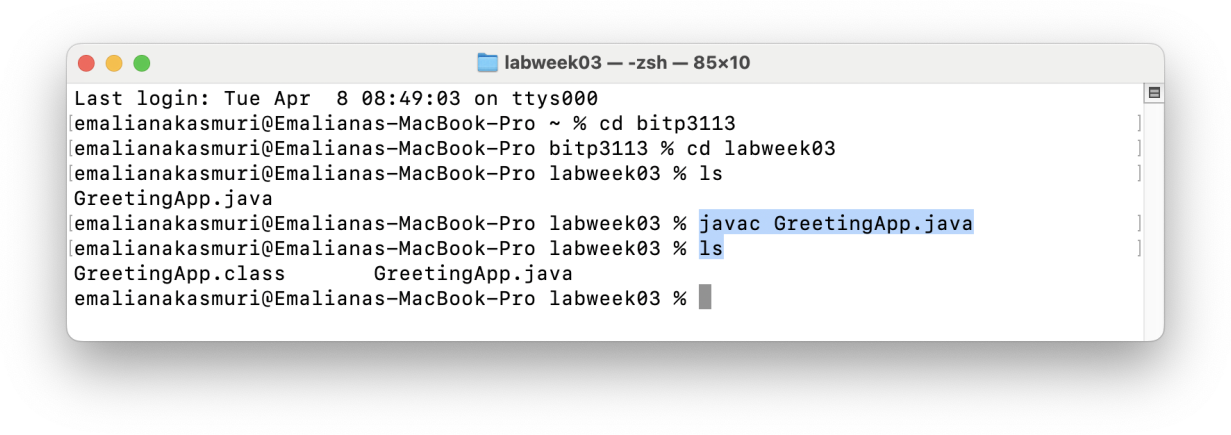


Figure 12: Outcome from javac command on GreetingApp.java

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

A screenshot of a computer screen

AI-generated content may be incorrect.

1. After that, type the java Greeting command in the terminal. The outcome should be similar as shown in [Figure 13](#_bookmark24).

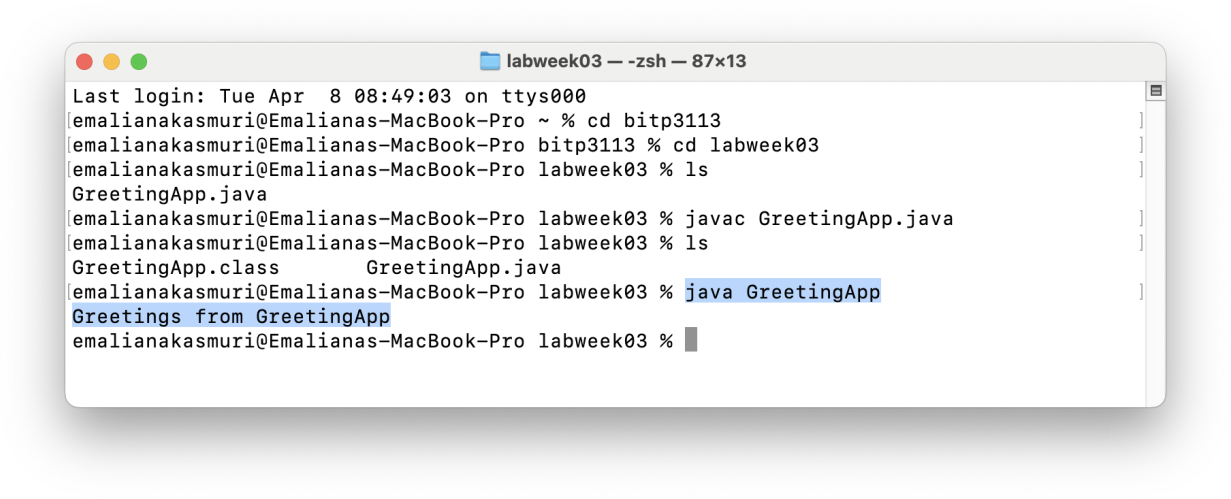


Figure 13: The outcome from java command on GreetingApp

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



## 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.**
   1. public = Line 10
   2. class = Line 10



* 1. static = Line 17
  2. void = Line 17
  3. String = Line 17
  4. Main = Line 17



1. **Identify the curly brackets { } for the class and the main() method.**

A screenshot of a computer program

AI-generated content may be incorrect.

1. **Identify the method to display the greeting message on the console.**

Line 21



1. **Identify the comment block.**

Line 2 – 9 and Line 12 – 16

A screenshot of a computer program

AI-generated content may be incorrect.

1. **Observe the class and the file name.**

The class name and the file name is the same which is GreetingApp

A screenshot of a computer program

AI-generated content may be incorrect.

1. 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](#_bookmark28).

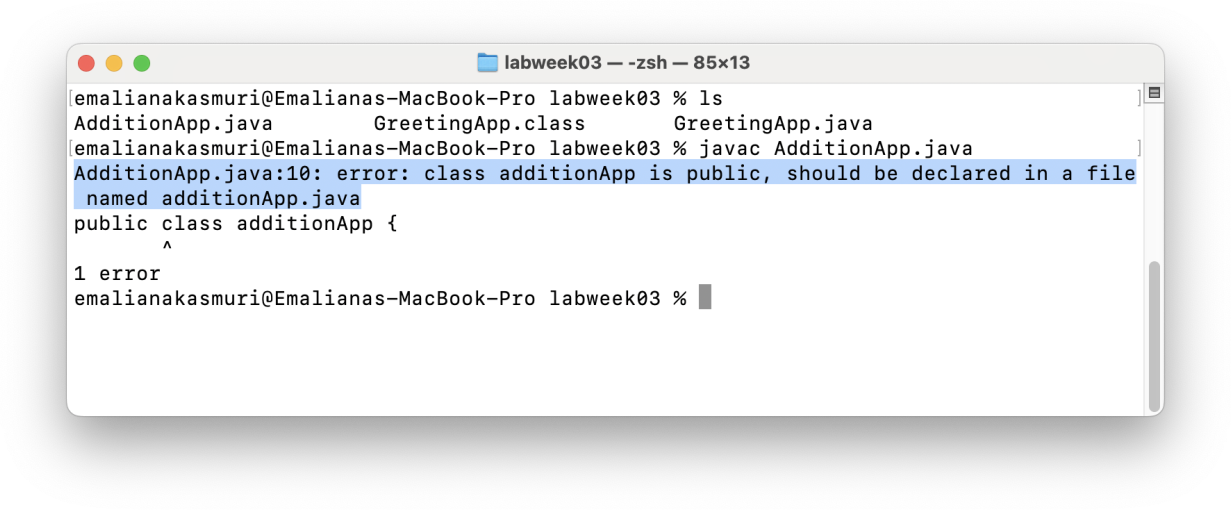


Figure 14: Outcome from AdditionApp.java compilation

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

1. The compilation shall produce 1 error message as highlighted in [Figure 14](#_bookmark28). Observe the error message anatomy in [Figure 15](#_bookmark29).

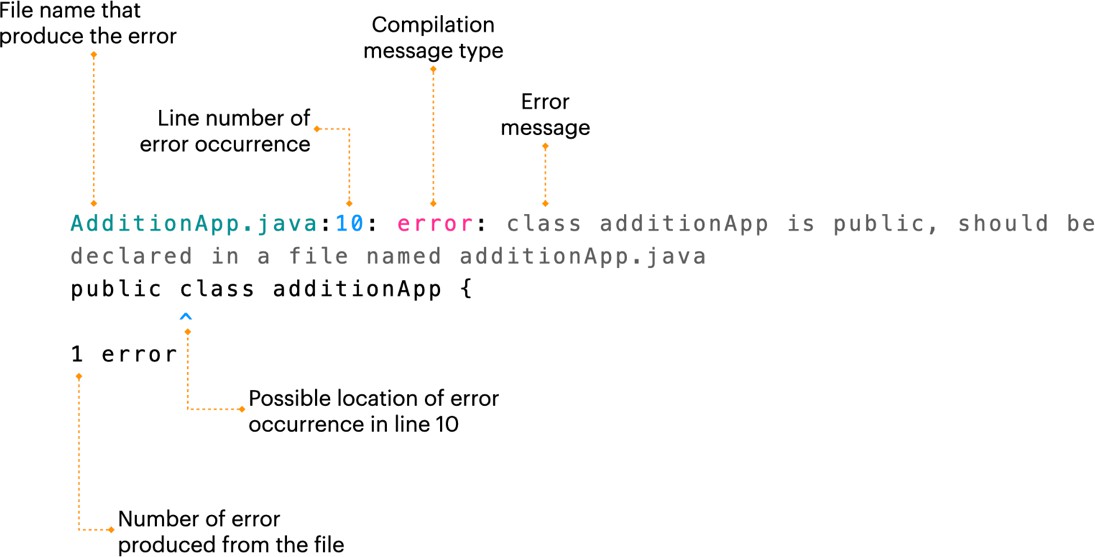


Figure 15: Anatomy of Java compilation error

## 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.**

Change the classname from additionApp to AdditionApp

Before



After



1. Save the file.
2. Compile the file.
3. Execute the application. The outcome shall be similar as shown in [Figure 16](#_bookmark31).

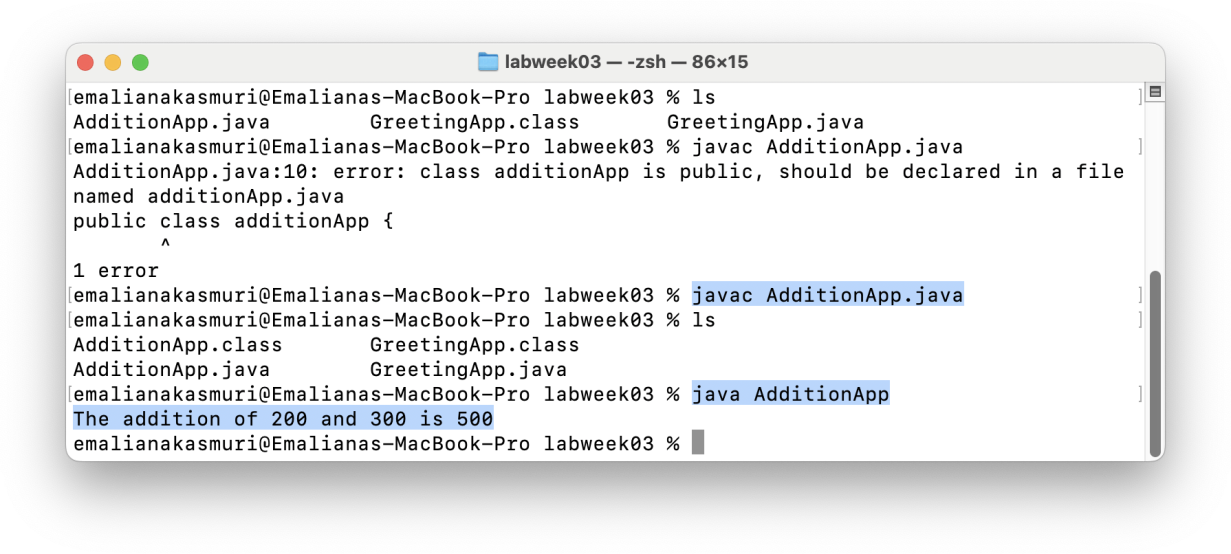


Figure 16: Outcome from AdditionApp execution



# 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](#_bookmark33). It may receive none or any number of execution variables.

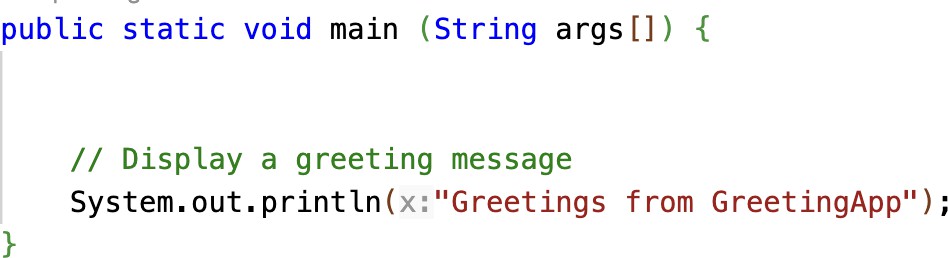


Figure 17: The main( ) method definition

## 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.**

**The original file is not saved in .java, so we First, the file needed to be renamed to change the extension.**

Before

A screen shot of a computer

AI-generated content may be incorrect.

After

A screenshot of a computer program

AI-generated content may be incorrect.

**Next, we need to change the class name to be the same as the file name, in this case the file name is StringManipApp but the class name is defined as TextManipApp**

Before



After



**Lastly, the main method is missing a keyword which is static, we need to change that, and the “Main” keyword is case sensitive so change it to “main”**

Before



After



1. Execute the application only when the compilation is free from any error. The application shall produce an output similar as shown in [Figure 18](#_bookmark35).



Figure 18: Output from StringManipApp execution

A screenshot of a computer screen

AI-generated content may be incorrect.

# 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**.

A screen shot of a computer

AI-generated content may be incorrect.

1. Compile the file.
2. **Using the knowledge comprehended until this point, fix the errors produced from the file.**

**First, we need to change the class name to the file name from DateManipulationApp to DateFormattingApp**

Before



After



**Fix all the semicolon**



A black background with white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.



A screenshot of a computer program

AI-generated content may be incorrect.



**Add the missing curly bracket**

A screen shot of a computer program

AI-generated content may be incorrect.

1. Execute the application only when the compilation is free from any error. The application shall produce an output similar as shown in [Figure 19](#_bookmark38).



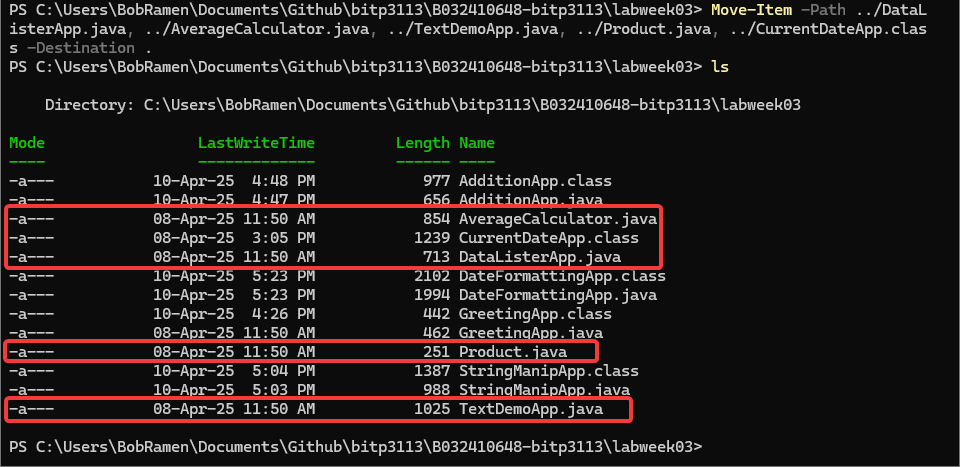
Figure 19: Output from DateManipulationApp execution

A screenshot of a computer

AI-generated content may be incorrect.

# Exercise 13: Executing Other Applications

1. Download other the following files from ulearn.
   1. DataListerApp.java
   2. AverageCalculator.java
   3. TextDemoApp.java
   4. Product.java
   5. CurrentDateApp.class
2. Move the file into folder named **labweek03**.



1. **Using the comprehended knowledge, compile the files.**
2. **Fix any errors produced from the compilation.**

**DataListerApp.java**

**Class name is not the same as filename.**

Before



After



Main method is written incorrectly

Before



After



**Semicolon is not added**





**Method to display text is written incorrectly**

Before



After



**Add all missing brackets**

A screenshot of a computer program

AI-generated content may be incorrect.

**AverageCalculator.java**

**Class is missing**

Before



After



**Main method is written incorrectly**

Before



After



**Add all semicolon**

A screenshot of a computer

AI-generated content may be incorrect.







**TextDemoApp.java**

**Main method is written incorrectly and there are 2 main method, there can be only one main method**

Before



After



**Insert all missing Semicolon**





**Method to display text is wrongly written**

Before

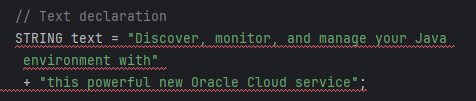


After



**STRING is an incorrect keyword, it is case sensitive**

Before



After

A screen shot of a computer monitor

AI-generated content may be incorrect.

**2 variables with the same name are defined, change to other variables**

Before

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

After

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**The second print statement is using the wrong variable**

Before

A screenshot of a computer

AI-generated content may be incorrect.

After

A black background with text

AI-generated content may be incorrect.

**Product.java**

**Class name is not the same as the file name**

Before



After



**Curly bracket does not tally**

A screen shot of a computer

AI-generated content may be incorrect.

**Add missing semicolon**

Before  


After



1. Execute the application.

**AverageCalculator.java**

A computer screen shot of a computer screen

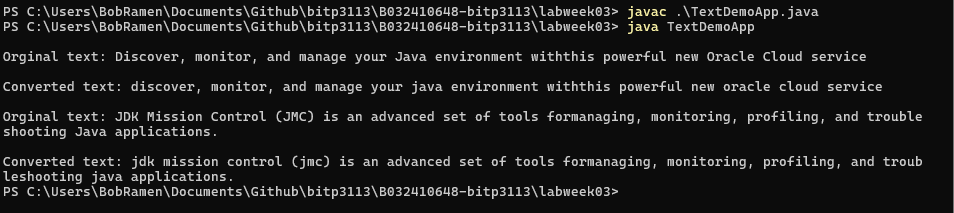
AI-generated content may be incorrect.

**DataListerApp.java**

A computer screen with white text

AI-generated content may be incorrect.

**TextDemoApp.java**



1. **One of the files is not executable even though the file passed compilation phase. Using the comprehended knowledge, record your analysis of this failure in ulearn.**

The file in question is Product.java

Reason:

* Java didn’t require a main method in each class to pass compilation
* If the syntax is correct, it can be compiled even though the code is technically not runnable/usable.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

1. **One of the files is executable even without .java. Using the comprehended knowledge, record your analysis on this success in ulearn.**

The file in question is CurrentDateApp.class

Reason:

* The .class file means that the code has already been compiled and has been converted to machine code so any device that can run Java can run it without any problem.

### 