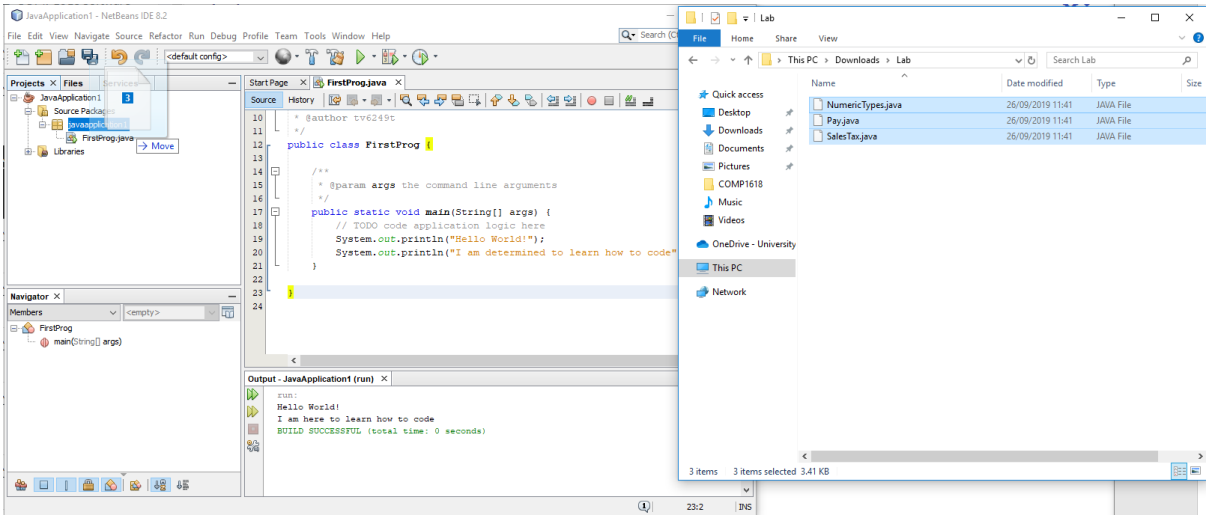
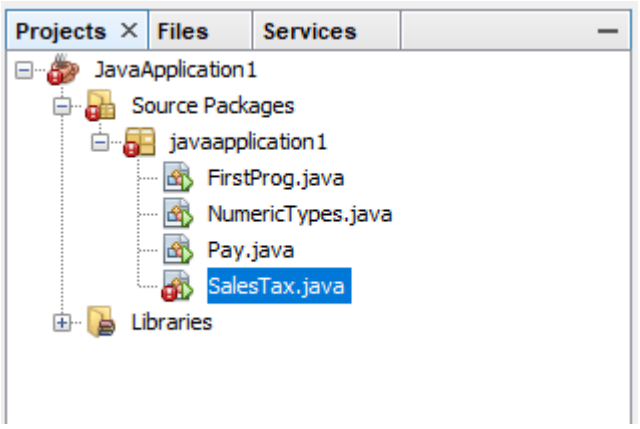
Exercise 2- Java Fundamentals

Part 1: Import Java files

1. Download the **Lab.zip** file from Moodle.
2. **Show in Folder** the file, right click to **Extract all..**
3. Select the 3 java files that were extracted, and drag them to a Java project within NetBeans, and drop the files when you see **-> Move**.



4. Now, the project will have 3 new java files, and one of them will shows a red **!** indicating an error to fix.



Part 2: Modify an existing program

In this part you will be working on two files: SalesTax.java and Pay.java.

SalesTax.java has several errors that you will need to correct for it to compile.

- **Syntax Errors**—errors in the "grammar" of the programming language. These are caught by the compiler and listed out with line number and error found. You will learn how to understand what they tell you with experience. All syntax errors must be corrected before the program will run. If the program runs, this does not mean that it is correct, only that

there are no syntax errors. Examples of syntax errors are **spelling mistakes in variable names, missing semicolon, unpaired curly braces**, etc.

- **Logic Errors**—errors in the logic of the algorithm. These errors emphasize the need for a correct algorithm. If the statements are out of order, if there are errors in a formula, or if there are missing steps, the program can still run and give you output, but it may be the wrong output. Since there is no list of errors for logic errors, you may not realize you have errors unless you check your output. It is very important to know what output you expect. You should test your programs with different inputs, and know what output to expect in each case.
- **Run time errors**—errors that do not occur until the program is run, and then may only occur with some data. These errors emphasize the need for completely testing your program.

So for debugging **SaleTax.java:**

- This file contains a simple Java program that contains errors. Compile the program. You should get a listing of syntax errors. Correct all the syntax errors, you may want to recompile after you fix some of the errors.
- When all syntax errors are corrected, the program should compile. As in the previous exercise, you need to develop some test data. Use the chart below to record your test data and results when calculated by hand.
- Execute the program using your test data and recording the results. If the output of the program is different from what you calculated, this usually indicates a logic error. Examine the program and correct logic error. Compile the program and execute using the test data again. Repeat until all output matches what is expected.

Item	Price	Tax	Total (calculated)	Total (ouput)

Fixing the algorithm for Pay.java:

- Use the file Pay.java. You will need to fill in the identifier declarations and a statement that is needed to enable Java to read from the keyboard. Below is the detailed pseudocode without the calculation part. You need to fill in lines that tell in English what the calculation part of Pay.java is doing.
  Display "How many hours did you work?"
  Input hours
  Display "How much do you get paid per hour?"
  Input rate
  Display the value in the pay variable.
  Your output should be as follows –

```
How many hours did you work? 40
How much do you get paid per hour? 5
You earned £200.0
```

- Compile the Pay.java. You should not receive any error messages.
- When this program is executed, it will ask the user for input. You should calculate several different cases by hand.

Part 3: Correcting Logic Errors

This lab is designed to give you practice with some of the basics in Java. You will continue ideas from above by correcting logic errors while looking at mathematical formulas in Java. You will explore the difference between integer division and division on your calculator as well as reviewing the order of operations.

This lab also introduces communicating with the user. You have already seen how console input and output works. You will now need to learn how to program user input, by investigating the lines of code that you need to add in order to use the Scanner class. You will also learn the method call needed for output.

- Use the file *NumericTypes.java*.
- Compile the source file, run the program, and observe the output. Some of the output is ***incorrect***. You need to **correct logic errors** in the average formula and the temperature conversion formula. The logic errors could be due to conversion between data types, order of operations, or formula problems. The necessary formulas are:

  average = *score*1 + *score*2 / *numberOfScores*
  C = (F – 32) * 5 / 9
- Each time you make changes to the program code, you must compile again for the changes to take effect before running the program again.
- Make sure that the output makes sense before you continue. The average of 95 and 100 should be 97.5 and the temperature that water boils is 100 degrees Celsius.

Documenting a Java Program

- Now look at *NumericTypes.java*, you will see that *NumericTypes.java* has lines which have information about what the program is doing. These lines are called comments and are designated by the // at the beginning of the line. Any comment that starts with /** and ends with */ is considered a documentation comment. These are typically written just before a class header, giving a brief description of the class. They are also used for documenting methods in the same way.
- Write a comment line at the top of the program which indicates the purpose of the program.
- Write a second comment line at the top of the program with your name and today's date.
- Add comment lines after each variable declaration, indicating what each variable represents.
- Add comment lines for each section of the program, indicating what is done in that section.
- Finally add a comment line indicating the purpose of the calculation.

Congratulations! You completed an important Java fundamental exercise.