

CSE 2102: Introduction to Software Engineering

Lab #1: Jan 25, 2022

Getting Started and Built-in Types

Note: Lab assignments are intended for practice. These will not be graded, and need not be submitted. Students are expected to try these by themselves. If you need help, TAs will be available during the lab times, and office hours, which can be found on HuskyCT.

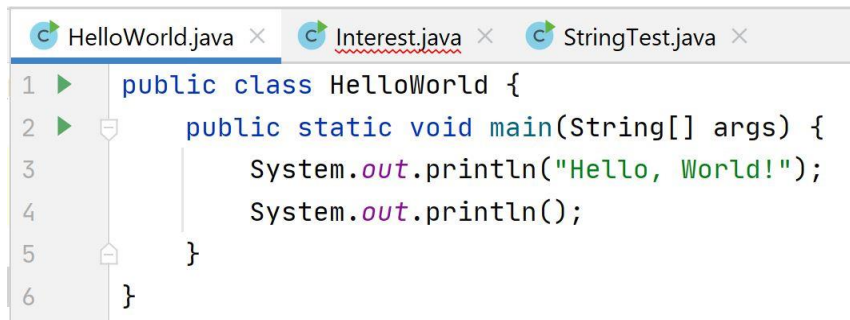
A. Getting Started

The objective of this lab exercise is to guide you through the basic steps required to get a simple program running in Java. It is recommended that you use IntelliJ, specifically, the following installation is used for programming, compiling, and executing Java programs.

<https://www.jetbrains.com/idea/download>

B. Hello World

Since the 1970s, it has been a tradition that when learning a new programming language, the first program should print “Hello, World”. So, first create a new Java project in IntelliJ named `Lab1`, create a new class named `HelloWorld` in the folder named `Lab1/src/` and type the following code:



```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4         System.out.println();
5     }
6 }
```

Compile and run the program in IntelliJ by selecting the green run button or selecting `Run>Run` in the menu bar.

C. Built-in Types (Numbers)

Java has eight primitive types of data, mostly for different kinds of numbers. Consider a simple program that computes the amount of interest that is earned on \$17,000 invested at an interest rate of 0.07 for two years. The interest and the value of the investment after two years are printed to standard output.

This program uses a number of built-in operations on these data types. In addition, it also uses `Math.pow(a, b)` to compute a^b . In general, Java has a large number of built-in functions and APIs available to manipulate these data types. These functions are available in the `Math` class. Functions in this class can compute trigonometric and exponential values and powers among other computations. At this point, it would help to familiarize yourself with the `Math` class.

```
HelloWorld.java x StringTest.java x Interest.java x
1  public class Interest {
2      public static void main (String[] args) {
3          // defining the variables
4          double principal;    // Actual amount invested
5          double rate;         // Annual interest rate
6          double interest;     // Interest earned in two years
7          double amount;       // Total amount, principal plus interest
8          double years;
9
10         // Computations
11         principal = 17000;
12         rate = 0.07;
13         years = 2;
14
15         // compute the compound interest after two years
16         amount = principal * Math.pow((1+ rate), years);
17         interest = amount - principal;
18
19         // print / output the results
20         System.out.println("The interest earned is");
21         System.out.println("$" + interest);
22         System.out.println("The total amount is");
23         System.out.println("$" + amount);
24     } // end of main
25 } // end of class interest
```

D. Input (Command line arguments)/Output

Inadvertently, we have seen how the functions `System.out.print` & `System.out.println` can be used to send the output to the terminal window. It is, however, also necessary for a Java program to take input values from the command line. Accepting command line inputs is a way to interact with the user. This is also necessary to run the program multiple times without having to re-edit and compile. All of our classes have a `main` method that takes a `String` array `args[]` as argument. That array is the sequence of command-line arguments that we type. If we intend for an argument to be a number, we must use a method such as `Integer.parseInt()` to convert it from string to the appropriate type. Consider the modified version of the interest calculation program that now accepts three inputs from the user – principal, rate, and number of years.

E. Command-line argument (Instruction to use cmd arguments)

```
public class Test {  
    public static void main(String[] args)  
    {  
        double number_1 = Double.parseDouble(args[0]);  
        double number_2 = Double.parseDouble(args[1]);  
    }  
}
```

Example of a java command-line for running the Test.java file:

```
java Test 1.2 2.3
```

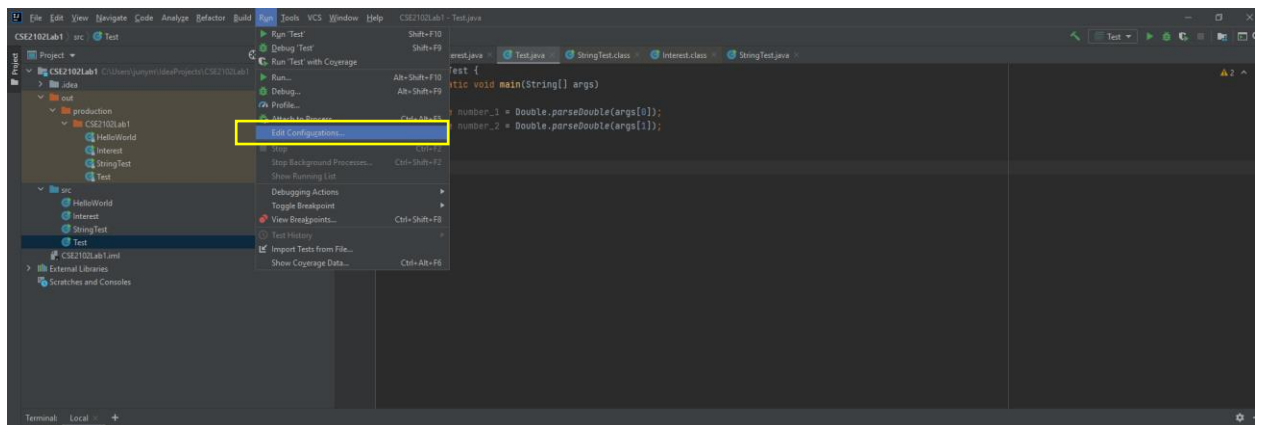
args[0] args[1]

2. Run command-line arguments

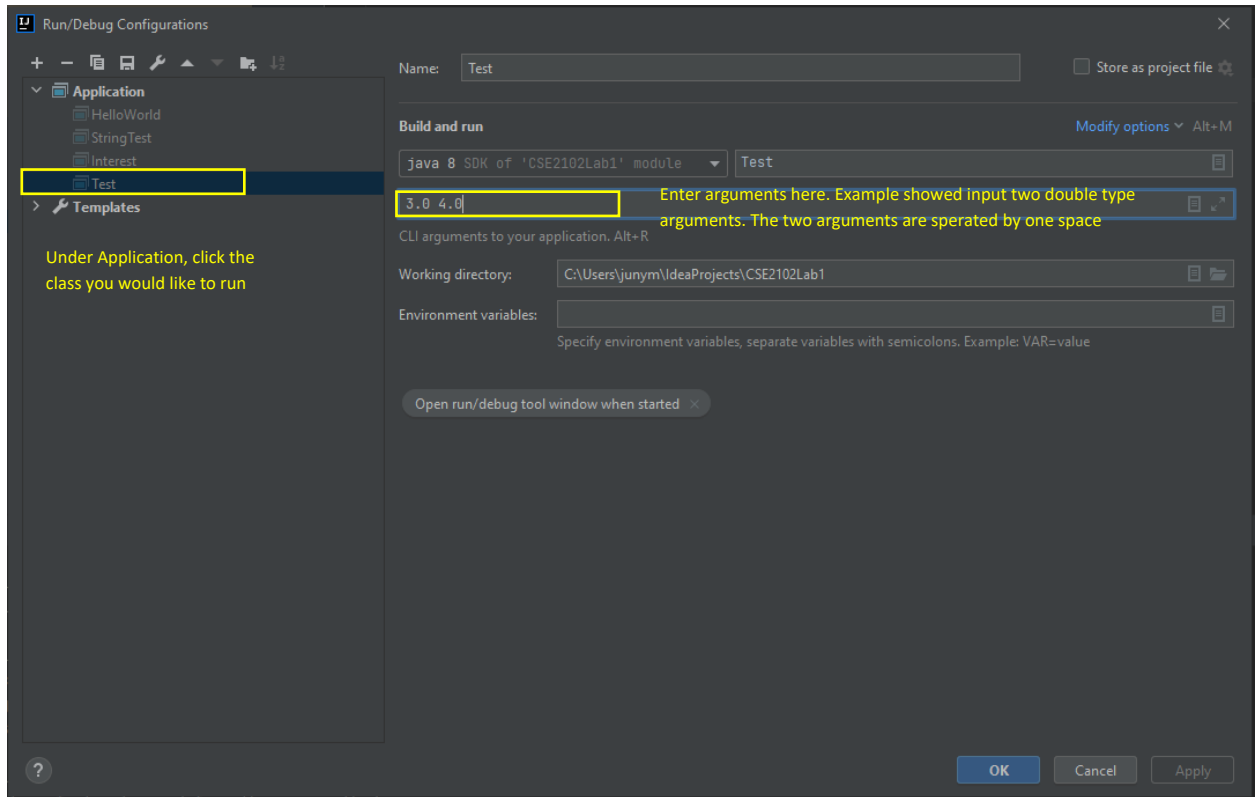
Below we demonstrate two ways to run command-line argument (these instructions are based on IntelliJ IDEA):

2.1 Input command-line arguments using run configuration

2.1.1 Go to “Run” tab => Edit Configuration.. as the image is shown below



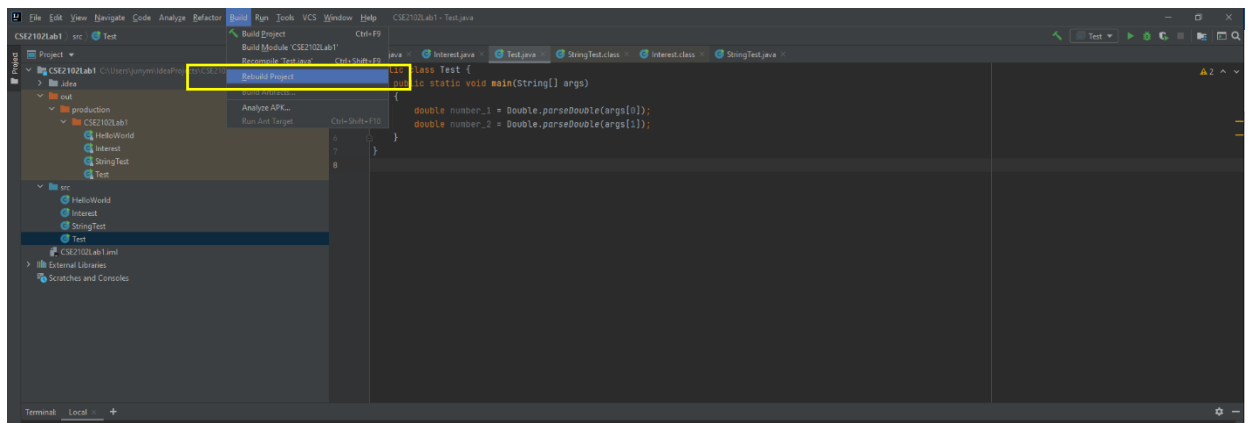
2.1.2 In the newly opened window, under “Application”, click the class you would like to run. Under “Build and run”, enter arguments you would like to input as command line arguments (The arguments should be separated by one space. For example, if you would like to input two arguments of type double, you would type 3.0 4.0 on the arguments bar). Example shown below:



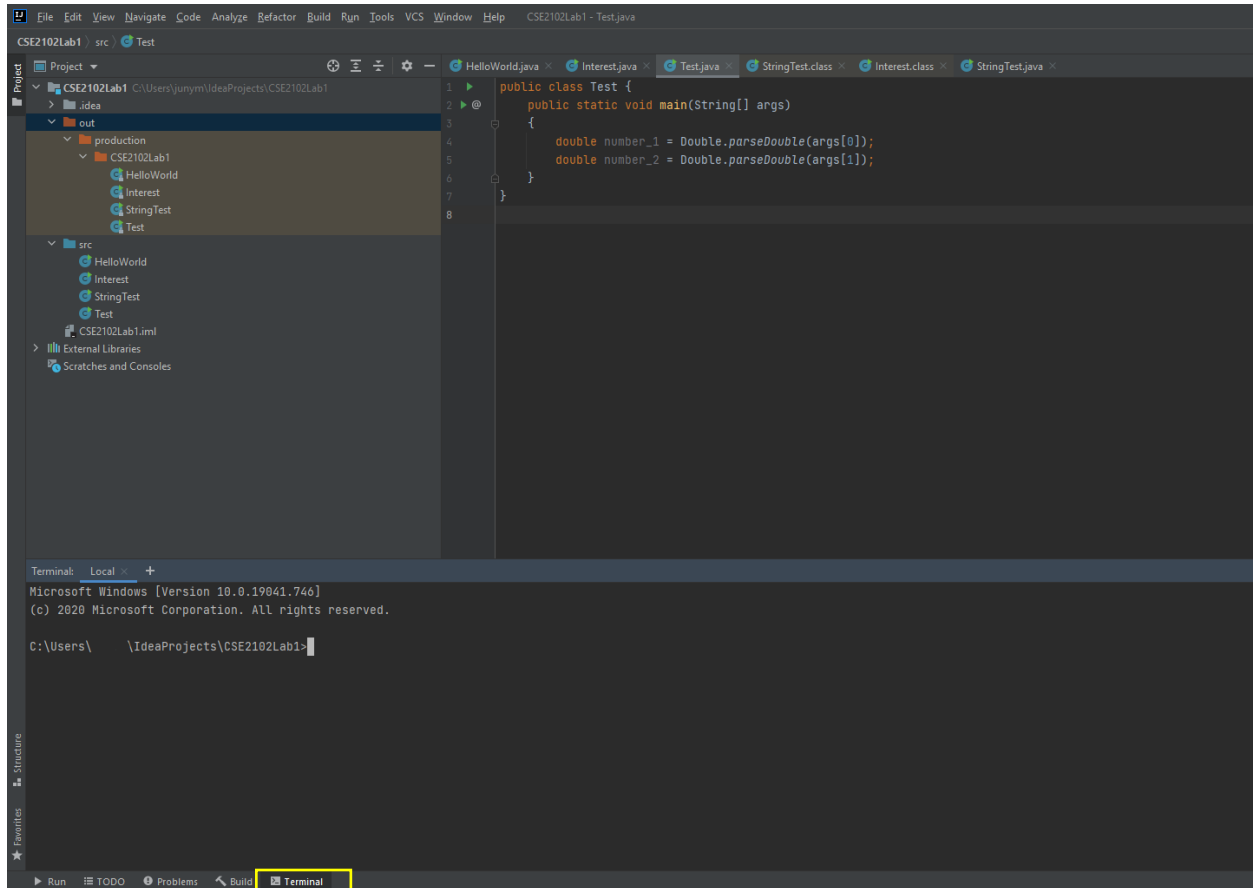
2.1.3 Click “Apply” and “Ok”. Now you can run your code by clicking the “Run” button. If you still get an error running your code, please double check if the number of arguments you input matches with the number of arguments accepted in your code. Please double check if the arguments you entered in the run configuration have been saved.

2.2 Use terminal

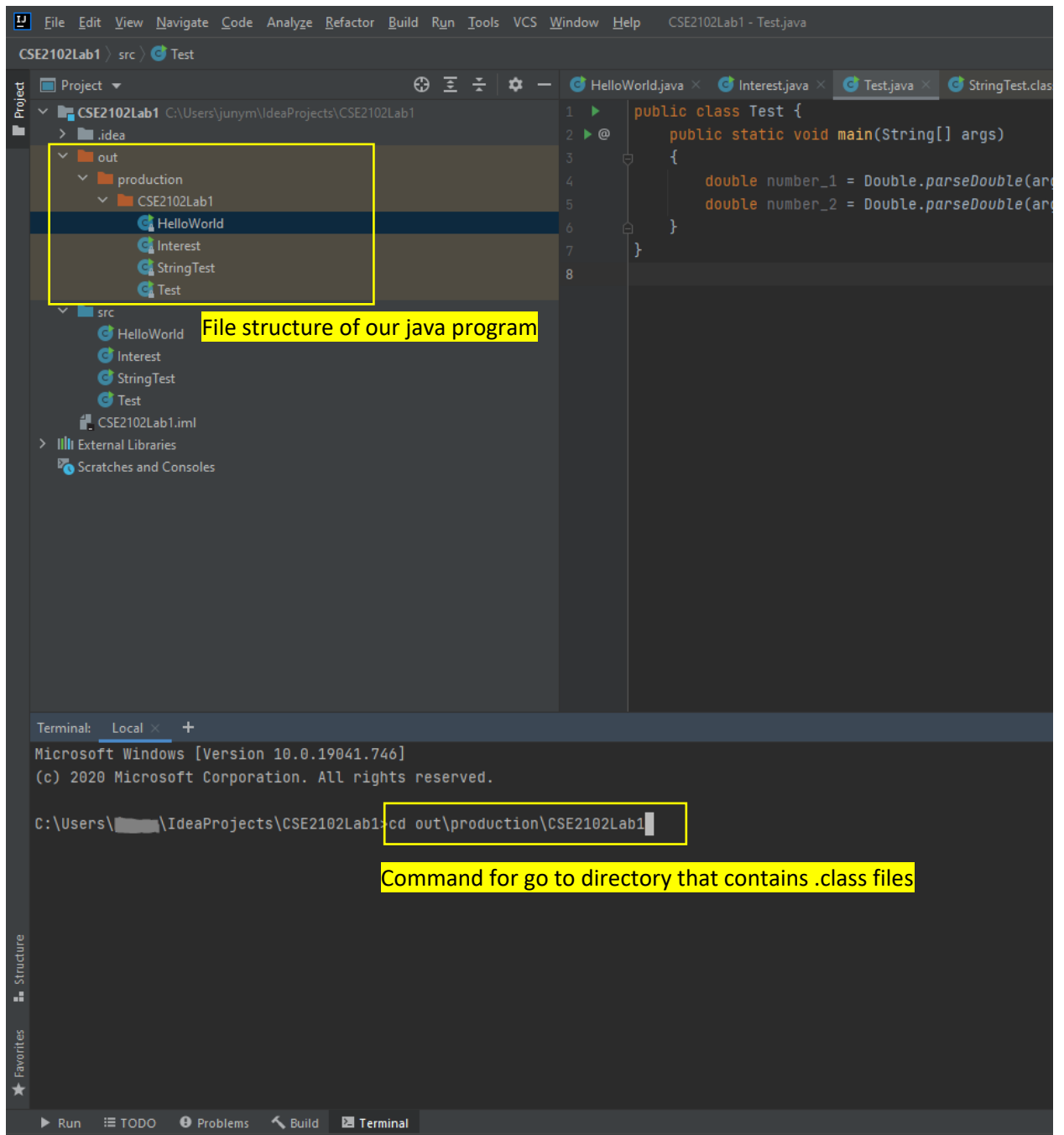
2.2.1 Go to “Build” tab, click “Build Project” as shown in the image below:



2.2.2 Go to the bottom of IntelliJ, you will find “Terminal” tab as shown in the image below. Click on the Terminal tab.



2.2.3 Go to the directory that contains your .class files. In IntelliJ, the .class files are under “out => production => <YOUR_PROJECT_NAME>”. The command for changing the directory is “cd”. Example of change directory is shown below:



2.2.4 Run your code using command line in the terminal as shown in the image below:

