Java Programming 1

PROFESSOR: CÂI FILIAULT

Topics for this week

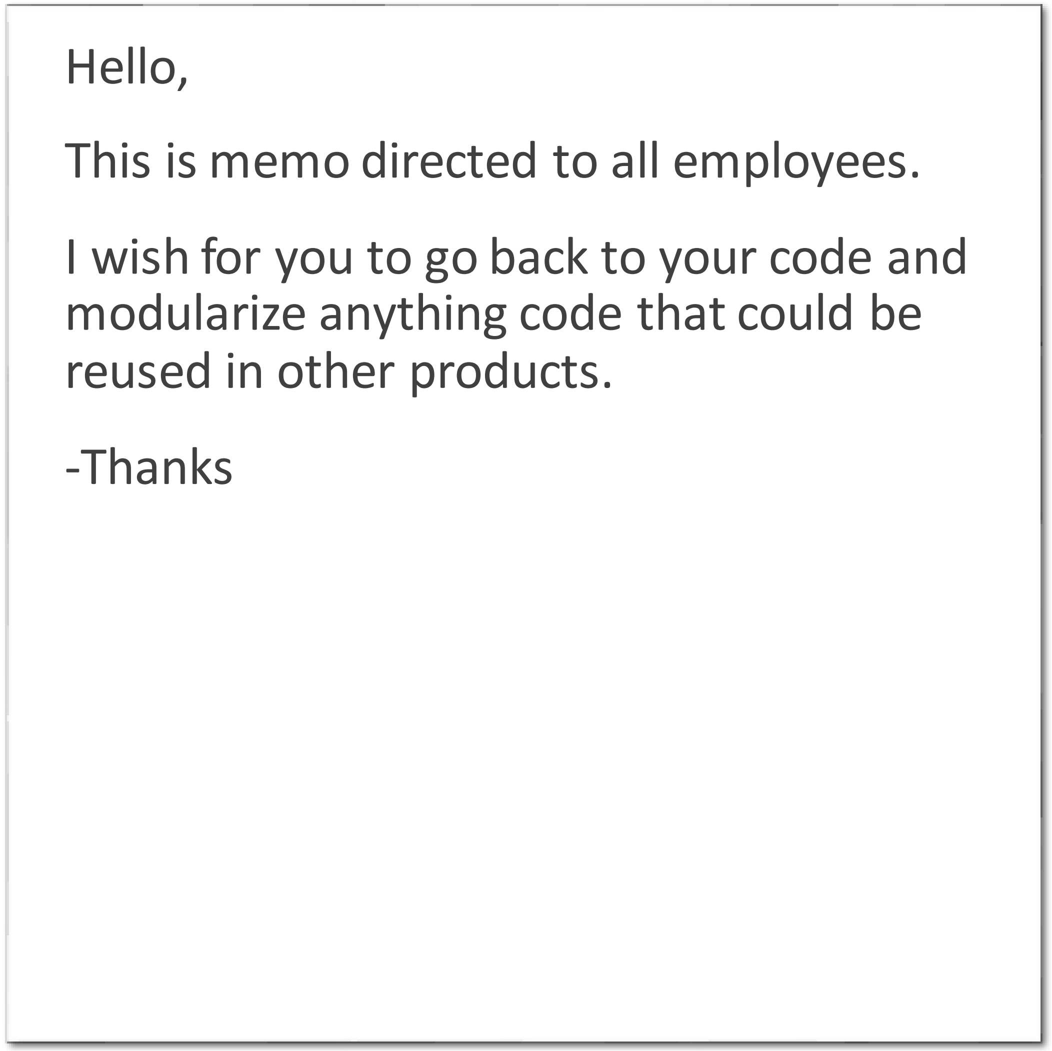
- Functions
- Creating
- Return Types
- Parameters
- Parameter Types
- Scope of functions

Company Email

INTERNAL EMAIL

Email

Company owner



Hello,

This is memo directed to all employees.

I wish for you to go back to your code and modularize anything code that could be reused in other products.

-Thanks

Email

Company Owner

Second Email

Hello,

I would like you to perform a code audit on the Comparison Calculator made for Marty. I noticed that the code could be made a bit more efficient and some of the code could be reused for other projects.

-Thanks

Plan of Action

- We review code in our previous work to ensure that it was done effective and efficiently.
- We also need to externalize any code that could be used in other projects.
- The first step is to learn what a function is and how we can use it to better our software.

Functions

A function is a block of code that is used to complete a task

A function is comprised of many statements that when called, get executed.

Functions are used to:

- Simplify tasks
- Simplify code
- Remove duplicate code
- Minimize the change for errors in code

Java functions are comprised of a few different components:

Functions

Acc. Modifier	Non-Acc. Modifier	Return type	Name	Formal parameters
private	static	int	Any name	Placed inside parentheses ()
public	final	boolean		Must declare data type
protected	abstract	double		Any type in return type except void
default	synchronized	String		May have multiple parameters
		char		
		void		

When building a function typically we follow the chart above:

1. Select an access modifier – in the event none is selected default is used
2. Select a non-access modifier (none can be selected)
3. Select a return type
4. Name the function
5. Formal parameters (if needed)

Access modifiers

Are used to provide the scope for which a function can be seen. It is possible to make your function visible to other software.

Like how we can use Math.random() or Math.ceil(). These functions exist in the Math class and we are accessing them from that program.

The following link provides a table view of where each modifier gives access

https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

Non-access modifier

Provide different features that your method will inherit:

Modifier	Purpose
static	The static modifier is something we will explain more when we explain object-oriented programming.
final	The final modifier ensures the developer cannot make override the method you have created
abstract	The abstract modifier is used in object-oriented programming and will be explained more there. Understand for now that it allows you to create a template for a function that will be created later.
synchronized	Is used in multithreaded programming to ensure only one thread uses a function at a time

Return type

Functions need to provide a return type. This indicates that the function when finished will return something to the place in which it was executed.

An example of this is the Math.sqrt() function:

- When we executed Math.sqrt(25) the program replaced it with the number 5
- https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#sqrt-double-
- If we take a second to look through the oracle documentation for sqrt, we can see that the return type is set to double.

Naming functions

When choosing a function name it has a couple rules that we would like to follow:

- Use **camel case for naming.** That is the first word is in lower case and every word in the name that follows starts with a capital letter.
- Try to **name it something meaningful** so that your code is self-documenting
- Try to **avoid having names start with an _ or $ sign.**
- Names **can contain letters, numbers and characters**
- Visit oracles website to see variable and method naming conventions

Formal parameters

When running a method it will execute a block of code. Sometimes we might want that block of code to run a little differently or possibly give a different result

Formal parameters are used so that we can provide a function with additional data at the time of execution

It's a way we can provide the function with pre-populated variables at the time it runs so that it can run a bit differently

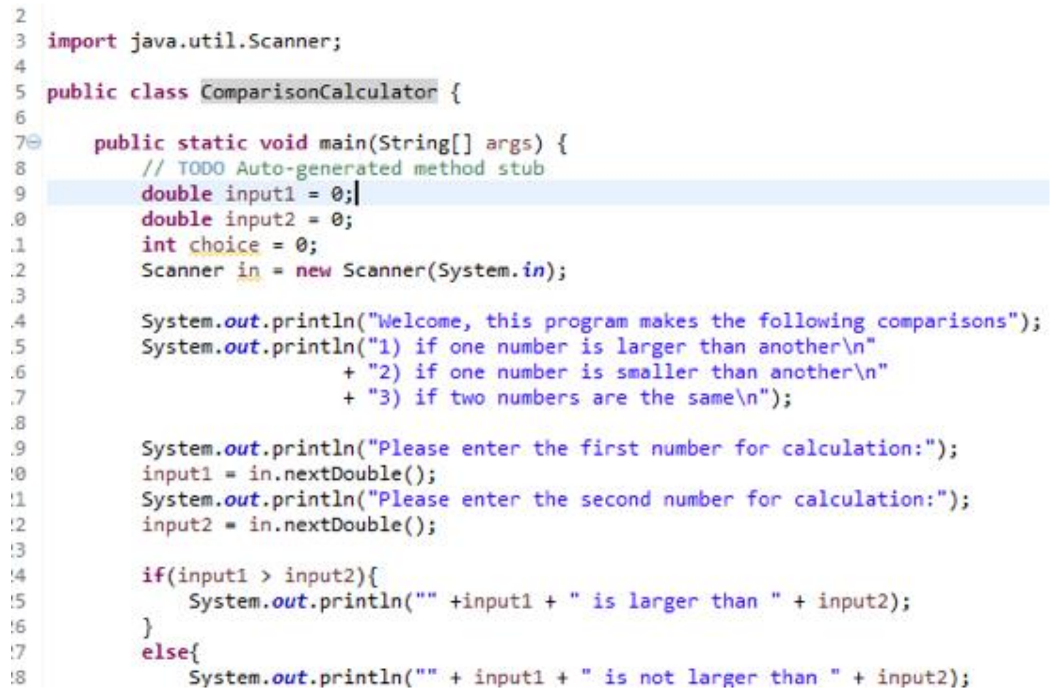
Examples of this are:

- System.out.println();
- Math.sqrt()
- Math.log10()

Code audit

I have uploaded a copy of the comparison calculator to blackboard for you to download if you have lost your copy.

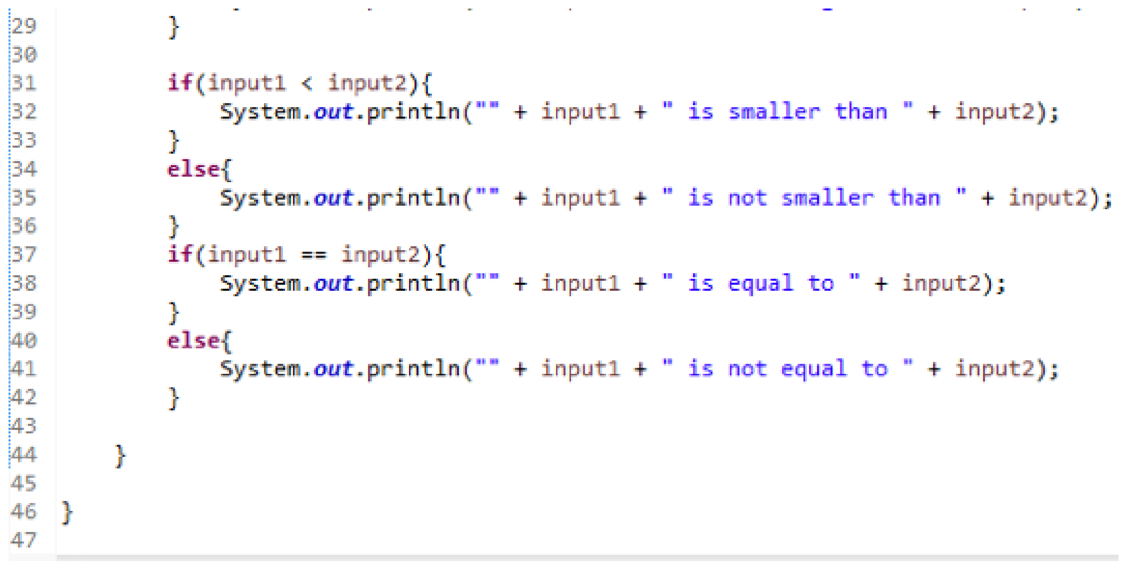
We are going to rewrite this comparison calculator to make a few improvements.



```
import java.util.Scanner;

public class ComparisonCalculator {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        double input1 = 0;
        double input2 = 0;
        int choice = 0;
        Scanner in = new Scanner(System.in);

        System.out.println("Welcome, this program makes the following comparisons");
        System.out.println("1) if one number is larger than another\n"
                         + "2) if one number is smaller than another\n"
                         + "3) if two numbers are the same\n");

        System.out.println("Please enter the first number for calculation:");
        input1 = in.nextDouble();
        System.out.println("Please enter the second number for calculation:");
        input2 = in.nextDouble();

        if(input1 > input2){
            System.out.println("" +input1 + " is larger than " + input2);
        }
        else{
            System.out.println("" + input1 + " is not larger than " + input2);
```

Code Audit

Remaining code



```
        }

        if(input1 < input2){
            System.out.println("" + input1 + " is smaller than " + input2);
        }
        else{
            System.out.println("" + input1 + " is not smaller than " + input2);
        }
        if(input1 == input2){
            System.out.println("" + input1 + " is equal to " + input2);
        }
        else{
            System.out.println("" + input1 + " is not equal to " + input2);
        }

    }

}
```

Code Audit

Currently the comparison calculator is computing the following:

- If one number is larger than another
- If one number is smaller than another
- If two numbers are equal

These are all very useful task that we may want to perform again (in this software or another)

Code Audit

Useful functions that can be derived from the Comparison calculator are the following:

- Min (a function that determines the smallest number given between two numbers)
- Max (a function that determines the largest number given between two numbers)
- Equals (a function that determines if two numbers are the same)

To build a function we need to use the steps outlined before:

- Choose a modifiers
- Choose a return type
- Choose a name
- Choose if we need to have parameters

Code Audit

In Java we only have a few options for our modifier. We will only be concerned with two modifiers:

Public:

- The public modifier lets the program know that the method can be publicly accessed

Static:

- Allows us to access the method in a static context
- We will go into more detail when we cover object-oriented programming

For our functions we will be using both public and static

Code Audit

In Java we have many return types (as listed in previous slides)

The first function we are building is the min function.

- The min function should return the smallest number back to the user.
- If we look at the comparison calculator software, it takes two doubles as input.
- Therefore if we are comparing two numbers the return type of the function should match the data type of the values it is comparing (double)

To perform a comparison between two numbers we need to provide the function with the data it needs to process.

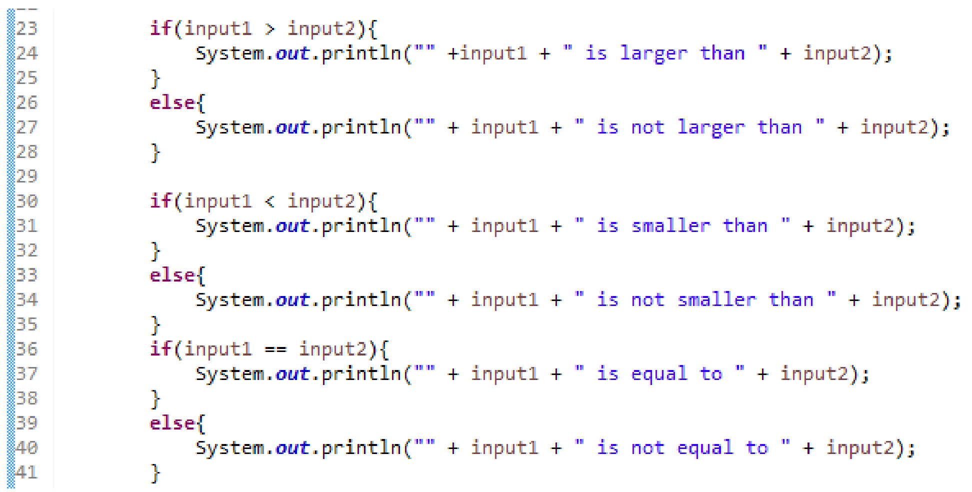
To do this we need to make use of parameters

Code Audit

The following is the piece of code we would like the convert into a series of functions.

We will use the same modifiers previously discussed

Each method we create will require two variables/parameters.



```java
        if(input1 > input2){
            System.out.println("" +input1 + " is larger than " + input2);
        }
        else{
            System.out.println("" + input1 + " is not larger than " + input2);
        }

        if(input1 < input2){
            System.out.println("" + input1 + " is smaller than " + input2);
        }
        else{
            System.out.println("" + input1 + " is not smaller than " + input2);
        }
        if(input1 == input2){
            System.out.println("" + input1 + " is equal to " + input2);
        }
        else{
            System.out.println("" + input1 + " is not equal to " + input2);
        }
```

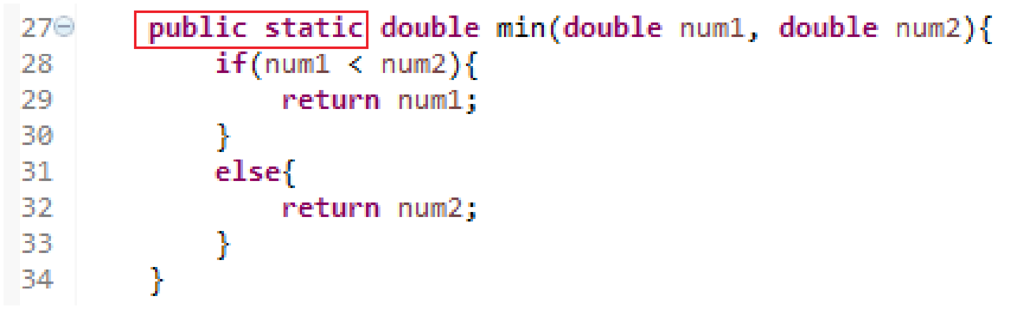
Code Audit

Let's create the first function min our ComparisonCalculator class.

When declaring methods they should be placed outside and below the main method (public static void main(String[] args){})

We start the declaration of the function by giving the modifiers public and static.

This tells the computer this function can be publicly accessed from multiple files.



```
public static double min(double num1, double num2){
    if(num1 < num2){
        return num1;
    }
    else{
        return num2;
    }
}
```

Code Audit

Next, we tell the computer the return type

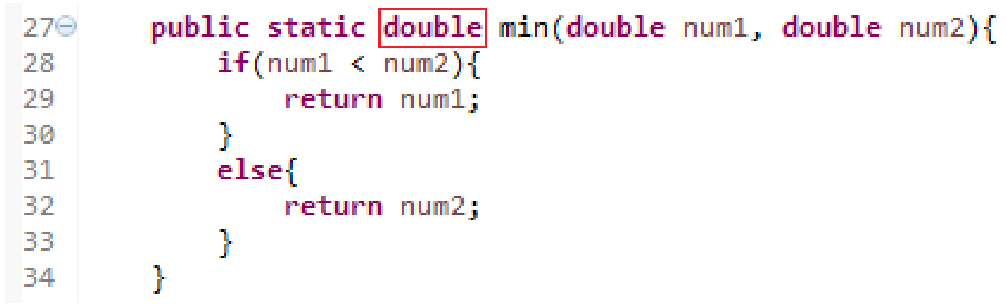
A return type tells the computer when this function is finished executing it will return a value to place in which the function was used.

Recall this return type can be void, String, int, char, Boolean, double etc..

The return can be ANY data type

The only data type that should need explanation is the void type.

A function with the return type void returns no value.



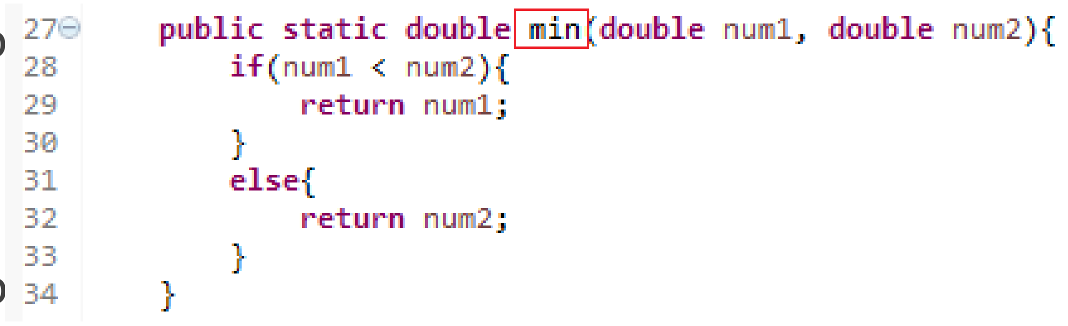
```
    public static double min(double num1, double num2){
        if(num1 < num2){
            return num1;
        }
        else{
            return num2;
        }
    }
```

Code Audit

The next step is to give the function a name, so people know how to call/use it.

We have named out function min.

Keep to the naming conventions provided in yo book and inside this slideshow



```java
public static double min(double num1, double num2){
    if(num1 < num2){
        return num1;
    }
    else{
        return num2;
    }
}
```

Code Audit

Finally we declare the parameters.

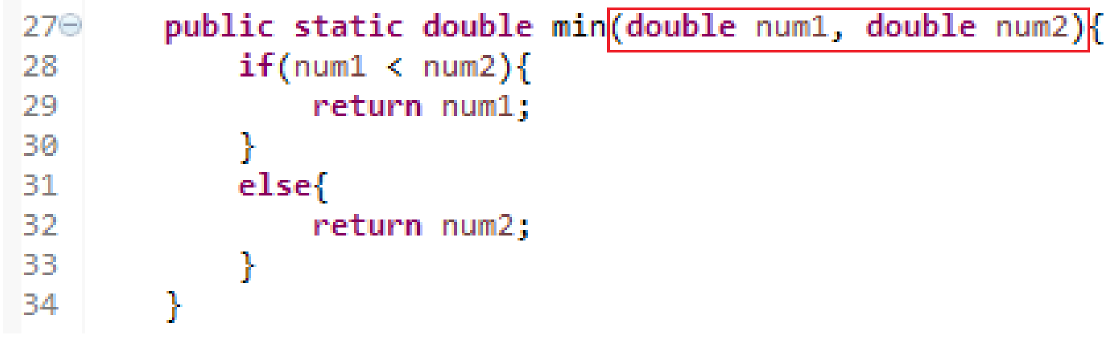
Parameters are used to pass information to the function.

Functions are like mini programs/tasks that we can perform

Parameters are our way of providing this mini program/task with data.

Two variables are declared inside of the parenthesis.

When the user calls the min() function they are expected to provide two pieces of information



```
public static double min(double num1, double num2){
    if(num1 < num2){
        return num1;
    }
    else{
        return num2;
    }
}
```

Code Audit

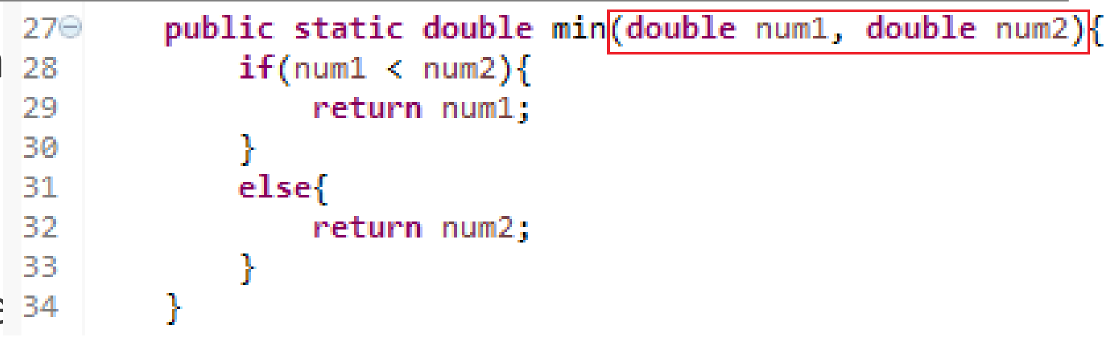
The two pieces of information we specifically wa
them to provide are two doubles.

min(5.0,3.0);

If we perform a trace the program than compare
the two numbers num1 and num2 and replaces
the function call with the value returned.

ex) min(5.0,3.0) + min(5.0,2.0);

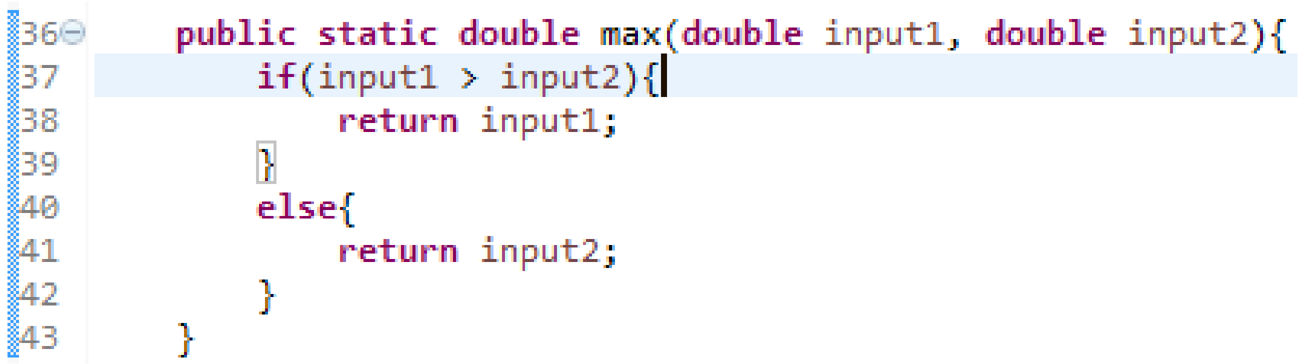
3 + 2



```
public static double min(double num1, double num2){
    if(num1 < num2){
        return num1;
    }
    else{
        return num2;
    }
}
```

Code Audit

Create the max function

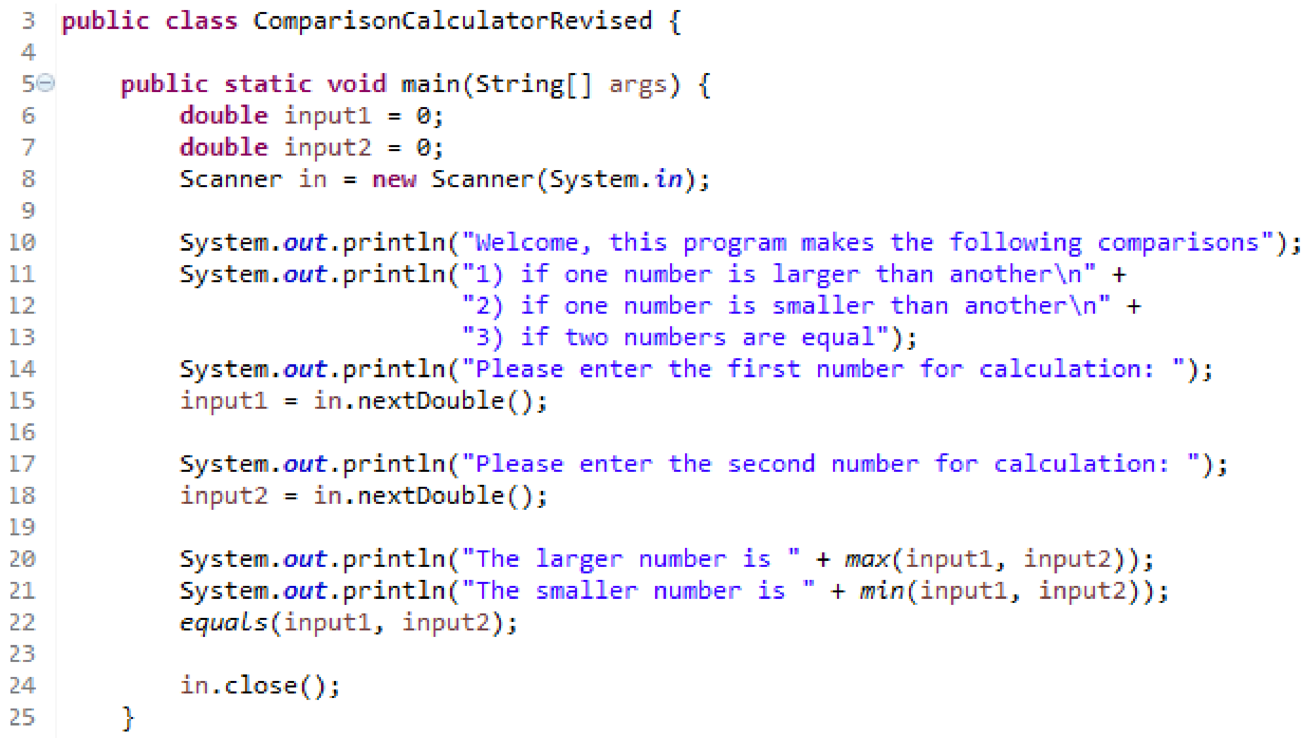


```
    public static double max(double input1, double input2){
        if(input1 > input2){
            return input1;
        }
        else{
            return input2;
        }
    }
```

Code Audit

We can now delete the old comparison from out software and utilize the new functions we created

Compile and run your software.



```java
public class ComparisonCalculatorRevised {

    public static void main(String[] args) {
        double input1 = 0;
        double input2 = 0;
        Scanner in = new Scanner(System.in);

        System.out.println("Welcome, this program makes the following comparisons");
        System.out.println("1) if one number is larger than another\n" +
                           "2) if one number is smaller than another\n" +
                           "3) if two numbers are equal");
        System.out.println("Please enter the first number for calculation: ");
        input1 = in.nextDouble();

        System.out.println("Please enter the second number for calculation: ");
        input2 = in.nextDouble();

        System.out.println("The larger number is " + max(input1, input2));
        System.out.println("The smaller number is " + min(input1, input2));
        equals(input1, input2);

        in.close();
    }
```

Marty A. Theodore

NEW EMAIL

Email

Marty A. Theodore

Grade one teacher

Hello,

I am helping my grade ones with their counting. I do so with the following activity:

I walk around the room speaking to each student individually. I ask the student to count as high as they can. I then record the students result. Each time I calculate if the student counted higher than all the other students. If they did, I record their score as the new highest score. I then continue to the next student until I have made my way around the room.

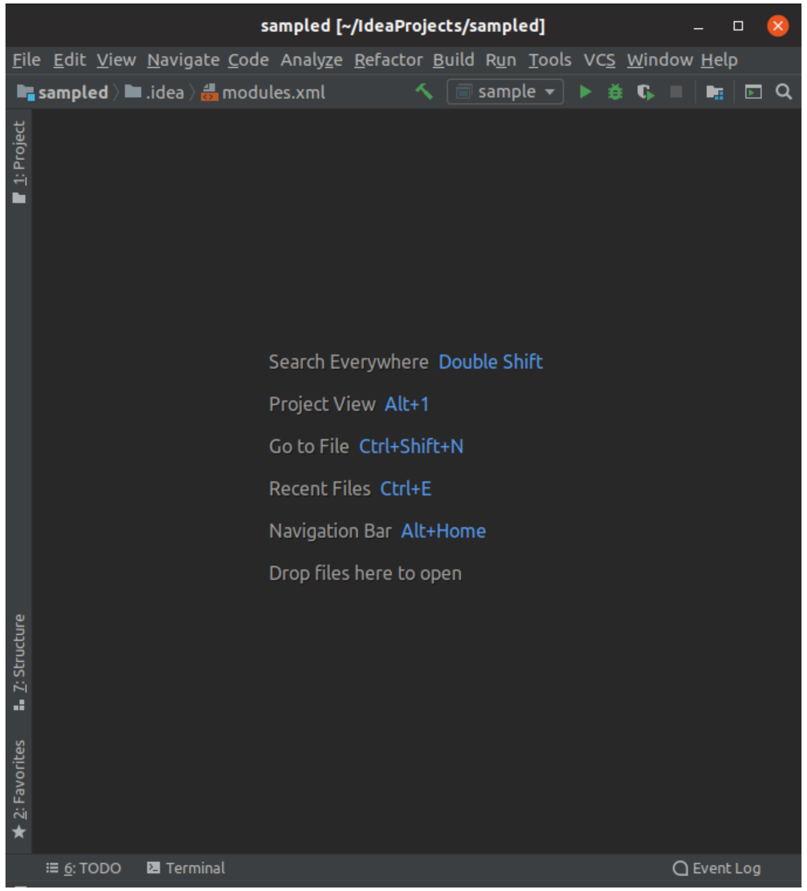
Can you build a program that is able to do the following:

- Allow me to enter a number
- Have the program keep running until I enter the number -1
- When I enter a number it will continually record if the number is higher than any number previously entered.

Creating a new project

Start by opening IntelliJ and selecting the following:

File> New> Project



sampled [~/IdeaProjects/sampled]
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
sampled
.idea
modules.xml
sample
1: Project
7: Structure
2: Favorites
6: TODO
Terminal
Event Log

Search Everywhere Double Shift

Project View Alt+1

Go to File Ctrl+Shift+N

Recent Files Ctrl+E

Navigation Bar Alt+Home

Drop files here to open

New Project

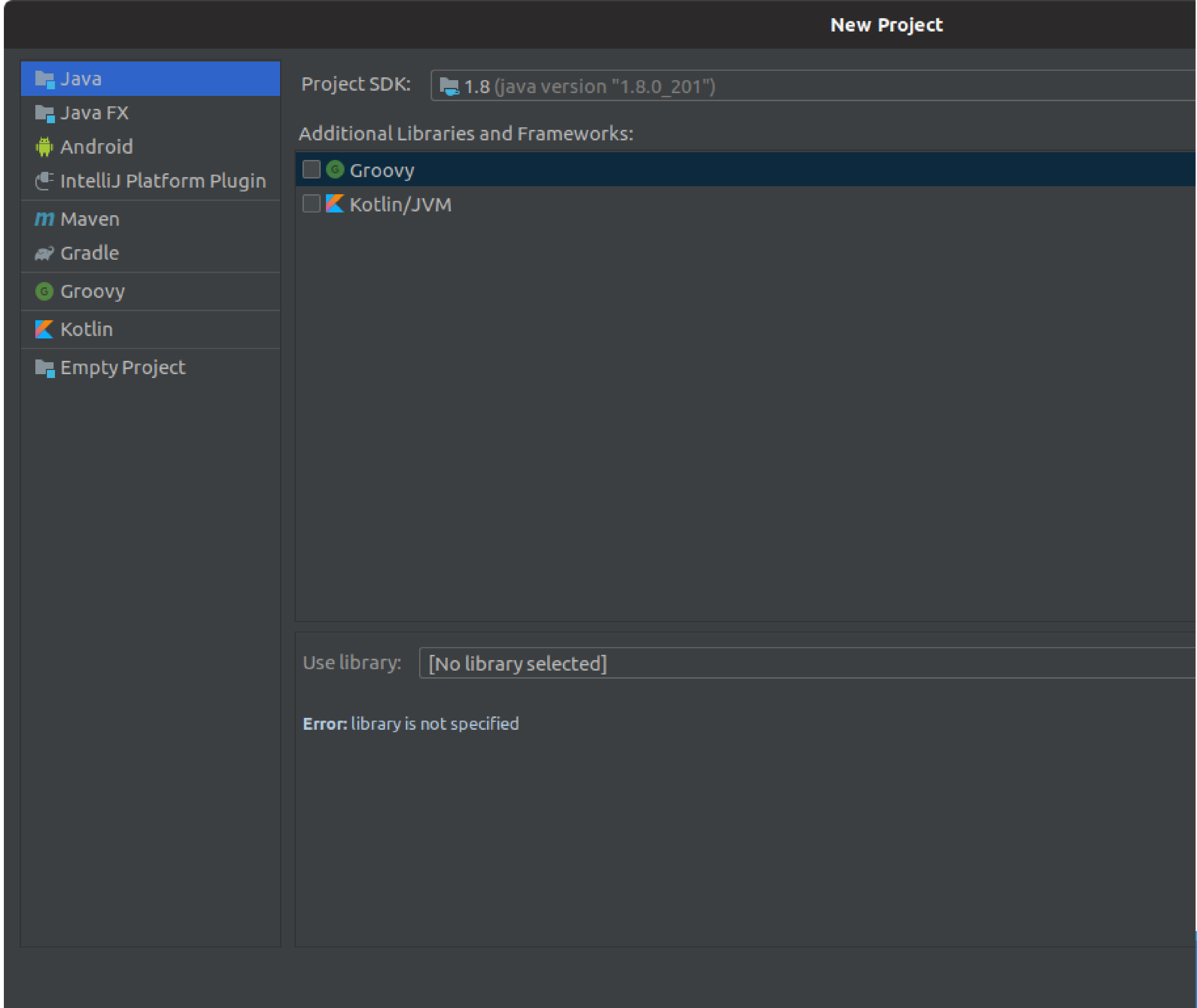
You will now see a dialog box appear *(It may be a little different than the one that I have shown)*

Starting on the left we will select a new Java Project

We will now want to select the project SDK of 1.8

SDK stands for Software Development Kit and it dictates which version of Java we are using.

At this point we can hit the next button twice


New Project

Java
Java FX
Android
IntelliJ Platform Plugin
Maven
Gradle
Groovy
Kotlin
Empty Project

Project SDK: 1.8 (java version "1.8.0_201")

Additional Libraries and Frameworks:
Groovy
Kotlin/JVM

Use library: [No library selected]

Error: library is not specified

New Project

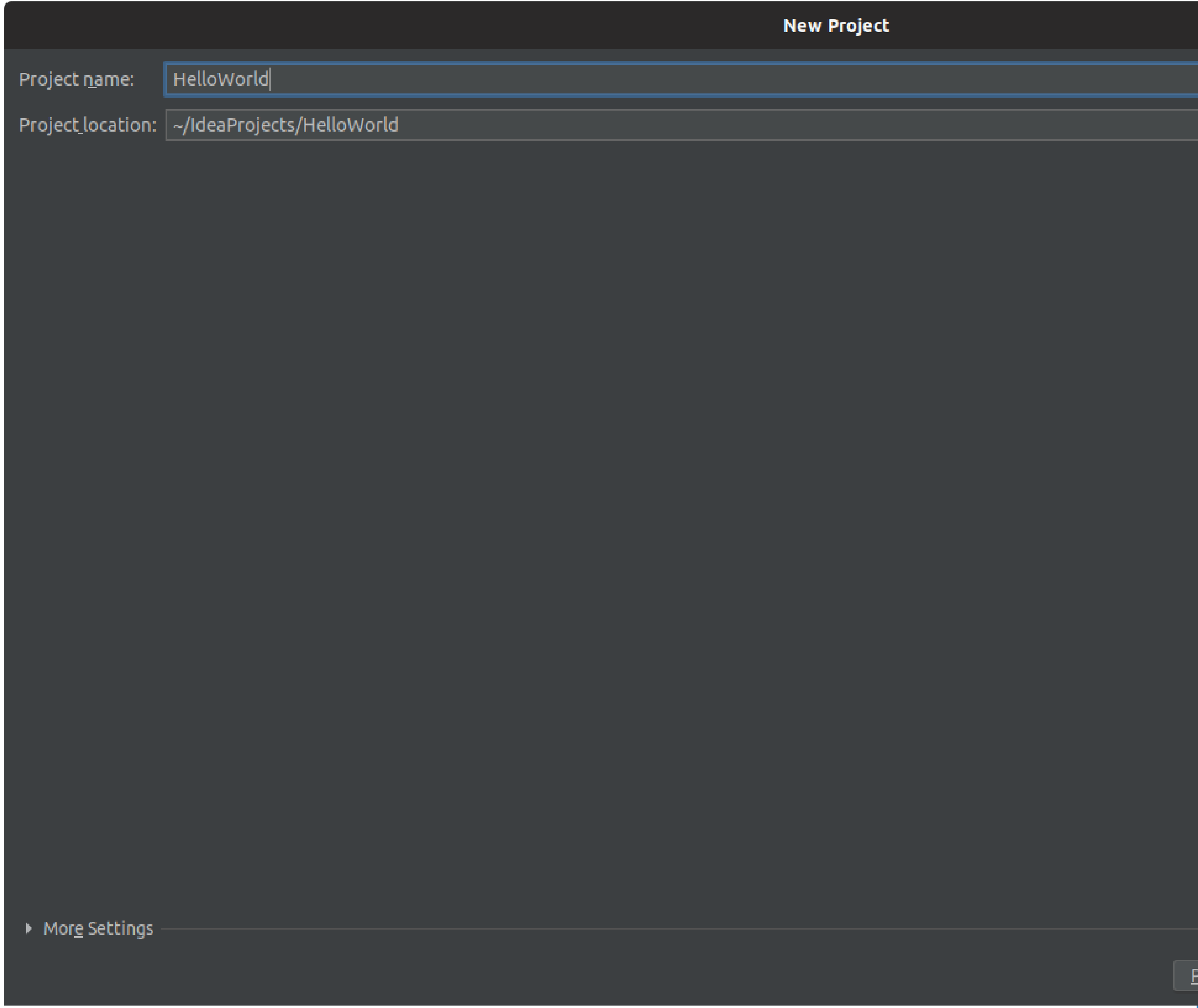
Give the project a name of:

HighLow

At this time you can change the project location to any convenient place you would like

Tip:

Keeping your programming projects on a flash/jump drive is perfectly fine. However you will want to avoid from directly working on the drive.


New Project

Project name: HelloWorld

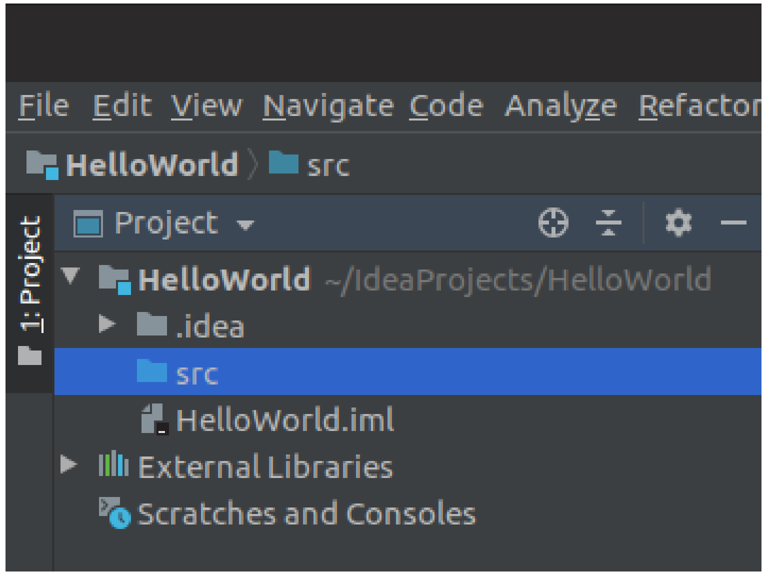
Project location: ~/IdeaProjects/HelloWorld

Creating a new project

You will now see in the top left-hand corner of your IDE a package explorer.

Within the package explorer you should see the **HighLow** project.

Expand the **HighLow** project to see a folder named **src**.



Creating a new class file

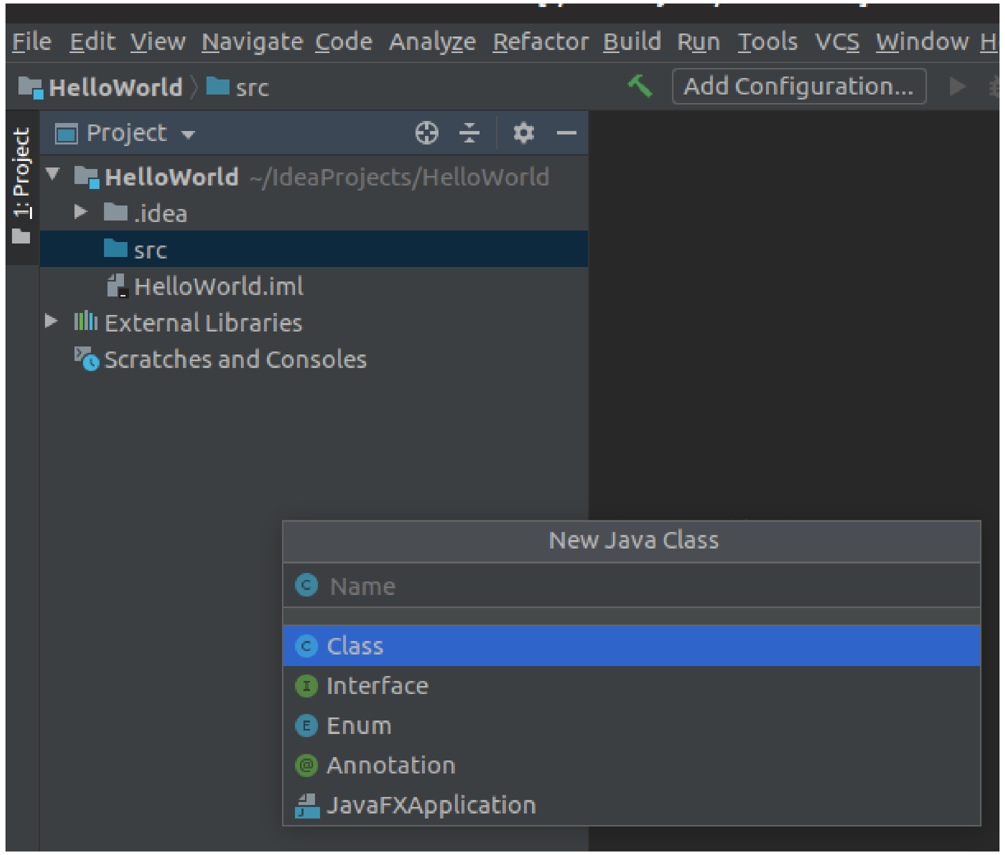
Next, we will need to create a new class file.

Right click on the **src** folder and select:

New> Java Class

Creating a new class is like creating a new program

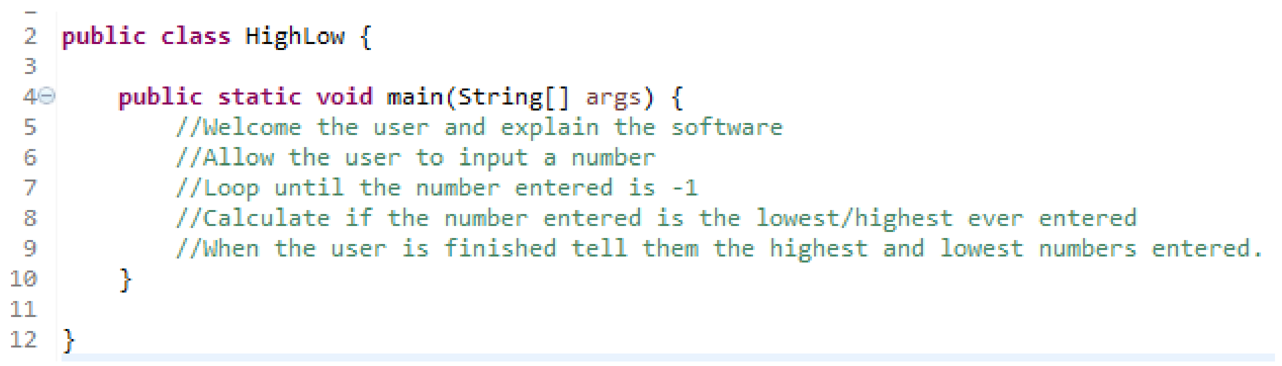
Name the class **HighLow** and hit **enter**



Writing the program

You should now see a HighLow.java file opened on the screen.

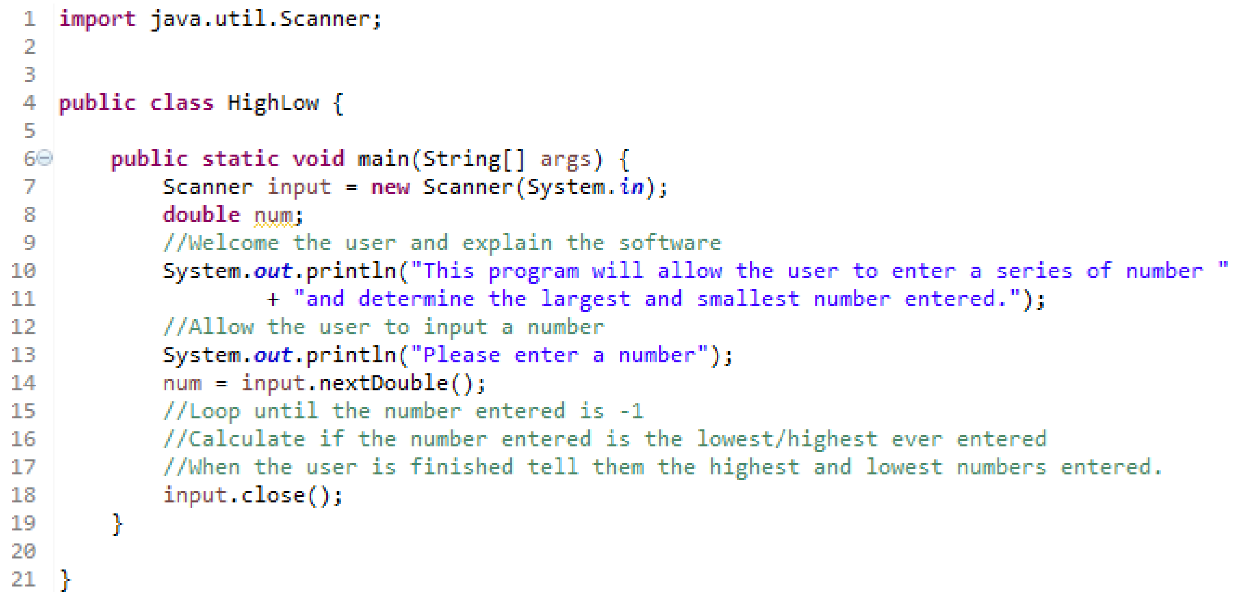
- Refer to the image on the right to see what your file should look like
- The first thing we are going to do is type "psvm" and hit the enter key
- This is what's known as "auto-completion"
- Add the comments to match ours shown



```java
public class HighLow {

    public static void main(String[] args) {
        //Welcome the user and explain the software
        //Allow the user to input a number
        //Loop until the number entered is -1
        //Calculate if the number entered is the lowest/highest ever entered
        //When the user is finished tell them the highest and lowest numbers entered.
    }

}
```

Writing the program

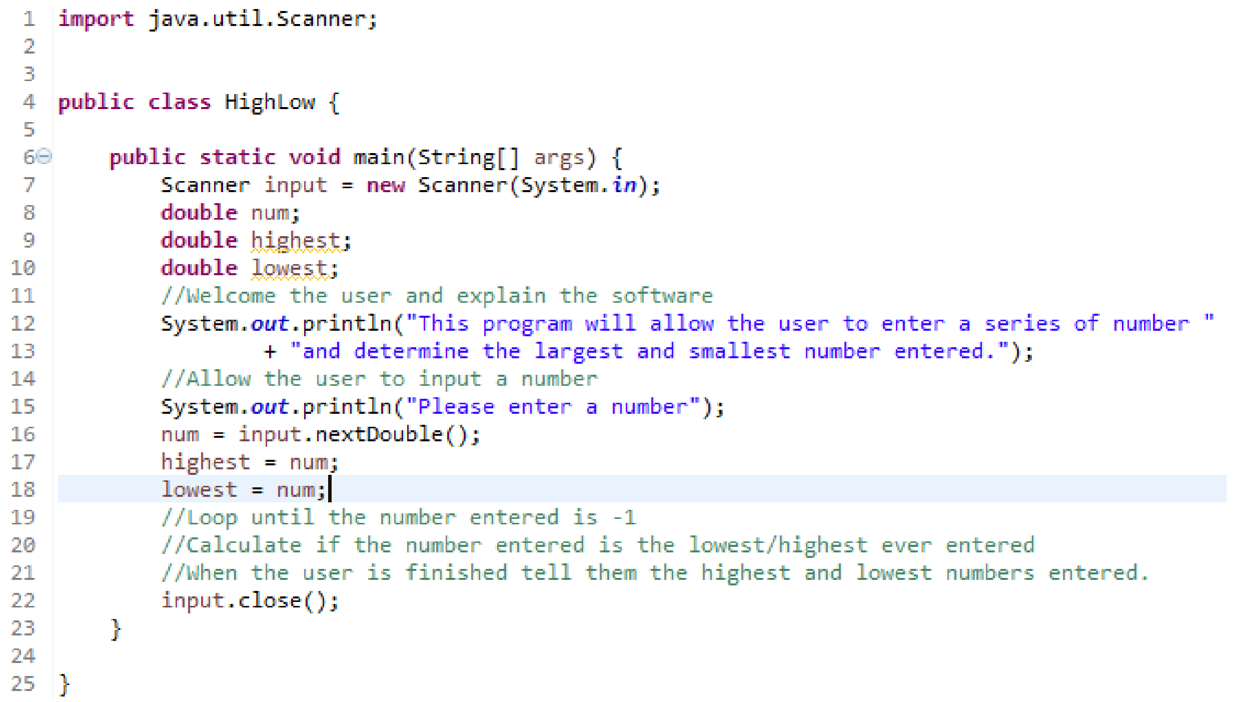
- We start by welcoming the user to the program (lines 10 and 11)
- We then open a channel of communication between the keyboard and the programmer with a new Scanner object.
- Finally we create a local variable named num (which stores the number entered by the user)



```
 1 import java.util.Scanner;
 2 
 3 
 4 public class HighLow {
 5 
 6     public static void main(String[] args) {
 7         Scanner input = new Scanner(System.in);
 8         double num;
 9         //Welcome the user and explain the software
10         System.out.println("This program will allow the user to enter a series of number "
11                 + "and determine the largest and smallest number entered.");
12         //Allow the user to input a number
13         System.out.println("Please enter a number");
14         num = input.nextDouble();
15         //Loop until the number entered is -1
16         //Calculate if the number entered is the lowest/highest ever entered
17         //When the user is finished tell them the highest and lowest numbers entered.
18         input.close();
19     }
20 
21 }
```

Writing the program

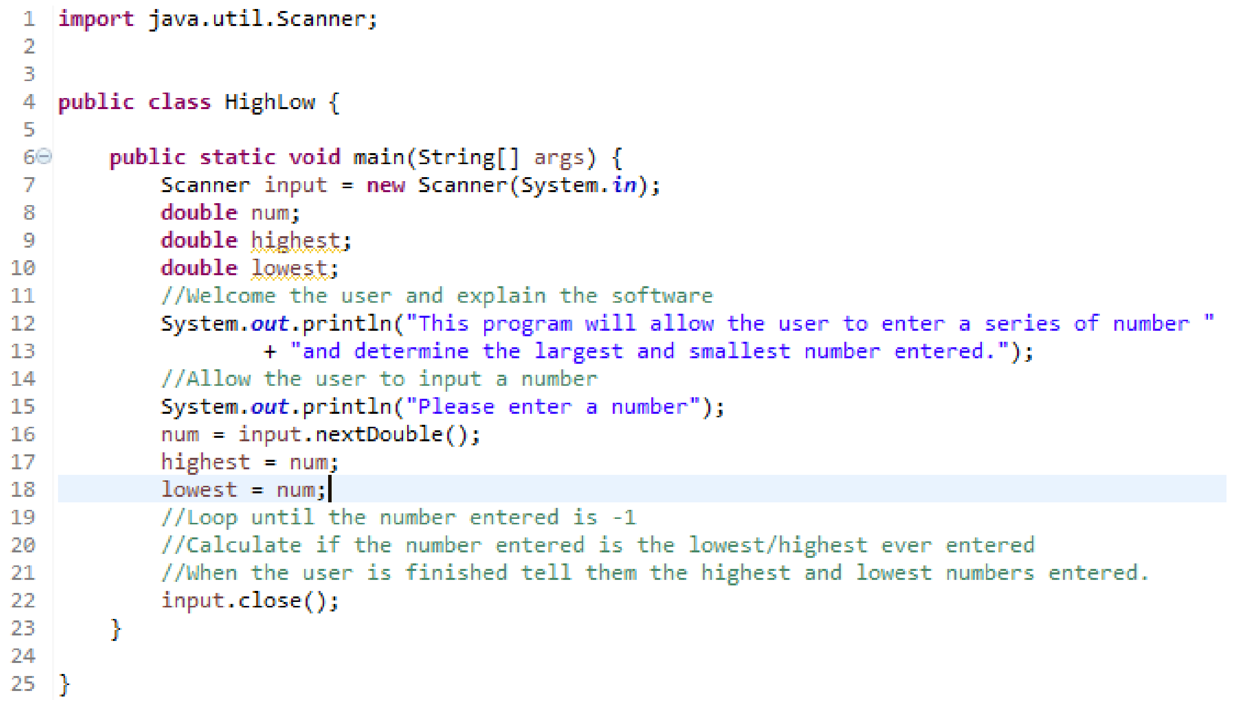
- Next, we create a variable highest and lowest.
- We will use these variables to keep track of the highest and lowest number entered by the user.
- Initially highest and lowest need to be set to the first number entered.
- If we initialize highest and lowest to 0 then we are assuming the numbers entered are going to be higher or lower than zero.



```java
import java.util.Scanner;


public class HighLow {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double num;
        double highest;
        double lowest;
        //Welcome the user and explain the software
        System.out.println("This program will allow the user to enter a series of number "
                + "and determine the largest and smallest number entered.");
        //Allow the user to input a number
        System.out.println("Please enter a number");
        num = input.nextDouble();
        highest = num;
        lowest = num;
        //Loop until the number entered is -1
        //Calculate if the number entered is the lowest/highest ever entered
        //When the user is finished tell them the highest and lowest numbers entered.
        input.close();
    }

}
```

Writing the program

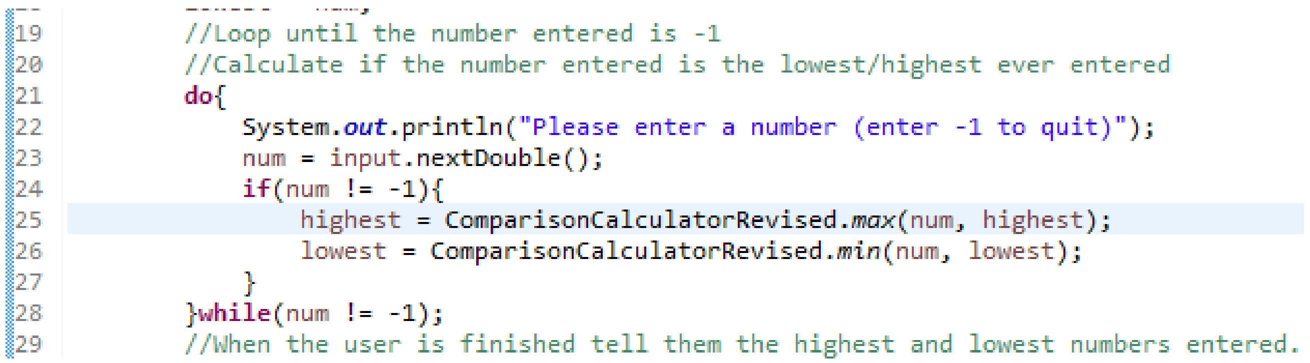
- If the user runs the program and enters 5, 4, 3, 2
- The lowest number entered would remain 0 because the user did not enter a lower number.
- To ensure our program is correctly and accurately recording the results we need to use the first number entered as a base.



```java
import java.util.Scanner;


public class HighLow {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double num;
        double highest;
        double lowest;
        //Welcome the user and explain the software
        System.out.println("This program will allow the user to enter a series of number "
                + "and determine the largest and smallest number entered.");
        //Allow the user to input a number
        System.out.println("Please enter a number");
        num = input.nextDouble();
        highest = num;
        lowest = num;
        //Loop until the number entered is -1
        //Calculate if the number entered is the lowest/highest ever entered
        //When the user is finished tell them the highest and lowest numbers entered.
        input.close();
    }

}
```

Writing the program

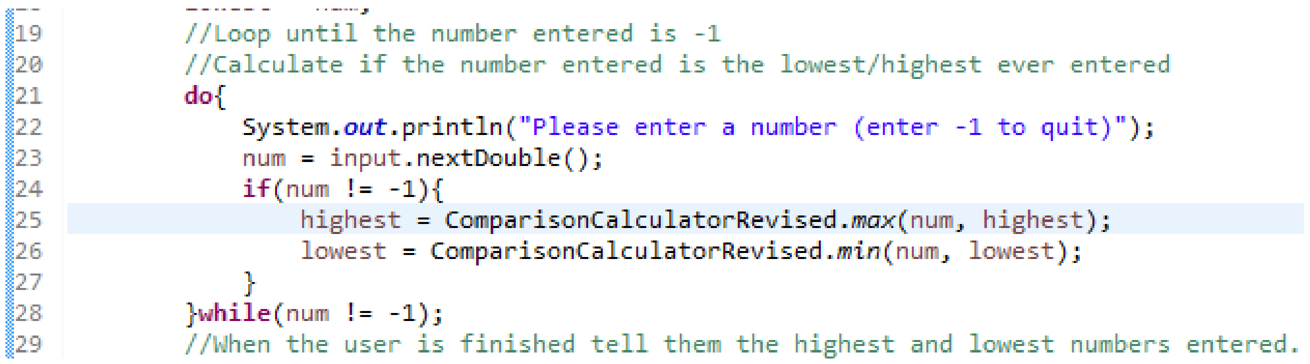
- Next, we need to create a loop that will allow the user to enter a number a correctly determine if it is the highest or lowest number.
- At this time we can use the functions we created in the previous exercise.
- Note we can ComparisonCalculatorRevised.min()
- and ComparisonCalculatorRevised.max()



```
//Loop until the number entered is -1
//Calculate if the number entered is the lowest/highest ever entered
do{
    System.out.println("Please enter a number (enter -1 to quit)");
    num = input.nextDouble();
    if(num != -1){
        highest = ComparisonCalculatorRevised.max(num, highest);
        lowest = ComparisonCalculatorRevised.min(num, lowest);
    }
}while(num != -1);
//When the user is finished tell them the highest and lowest numbers entered.
```

Writing the program

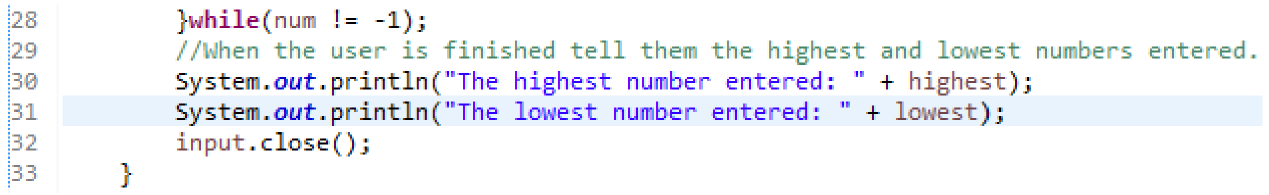
- We cannot begin to see full circle how math functions, system functions and all other functions work.
- We can have files that contain within them functions and we can reference those functions in other files.
- Math.Random() is in the Math library and is a function named Random().



```
        //Loop until the number entered is -1
        //Calculate if the number entered is the lowest/highest ever entered
        do{
            System.out.println("Please enter a number (enter -1 to quit)");
            num = input.nextDouble();
            if(num != -1){
                highest = ComparisonCalculatorRevised.max(num, highest);
                lowest = ComparisonCalculatorRevised.min(num, lowest);
            }
        }while(num != -1);
        //When the user is finished tell them the highest and lowest numbers entered.
```

Writing the program

- Finally we can output the highest and lowest numbers to the screen.



```
        }while(num != -1);
        //When the user is finished tell them the highest and lowest numbers entered.
        System.out.println("The highest number entered: " + highest);
        System.out.println("The lowest number entered: " + lowest);
        input.close();
    }
```

Test your software

Take a moment to test your software

Homework

Read Chapter 6 of your textbook

Next Week

- Test