

Java Programming 1

PROFESSOR: CÂI FILIAULT

Topics for this week

Variables

Console

Assignment
Statements

Naming
conventions

Data Types

Operators

Numeric
Type Casting

Truss Goodman

OUR FIRST CLIENT – FOLLOW UP

Email

Truss Goodman

Local business owner

Owns a roofing business

Truss Goodman – Roofing Inc.

Hello,

I would like the following changes made:

- I need a way I can change the shingle cost, customer roof size and installation cost
- Please make sure the program is also calculating taxes
- Could the program output the total before and after taxes

-Thanks

Final Product

At the end of the class we plan to have a final product that looks like the following:

```
Enter the cost of the shingles (per square foot):  
500  
Enter the size of the roof (in square feet):  
200  
Enter the cost of installation (per square foot):  
5  
Sub Total: $500000.0  
Total: 565000.0
```

Steps:

Let's build a list of what we need to complete:

1. Allow the user to input the shingle cost
2. Allow the user to input the size of the roof in square feet
3. Allow the user to input the installation cost per square feet
4. Inform the program about relevant taxes
5. Calculate two totals (one before and one after taxes)
6. Output the two totals on the console

Current Problem:

Currently we must alter the variables in our code to change the inputs for:

- Shingle cost
- Customer roof size
- Installation cost.

After editing these variables we need to recompile our code and execute to see the result.

Truss wants to be able to enter the values at run time and have the program compute those values.

Let's look at how we can make this possible:

Step 1-3) Scanner Class

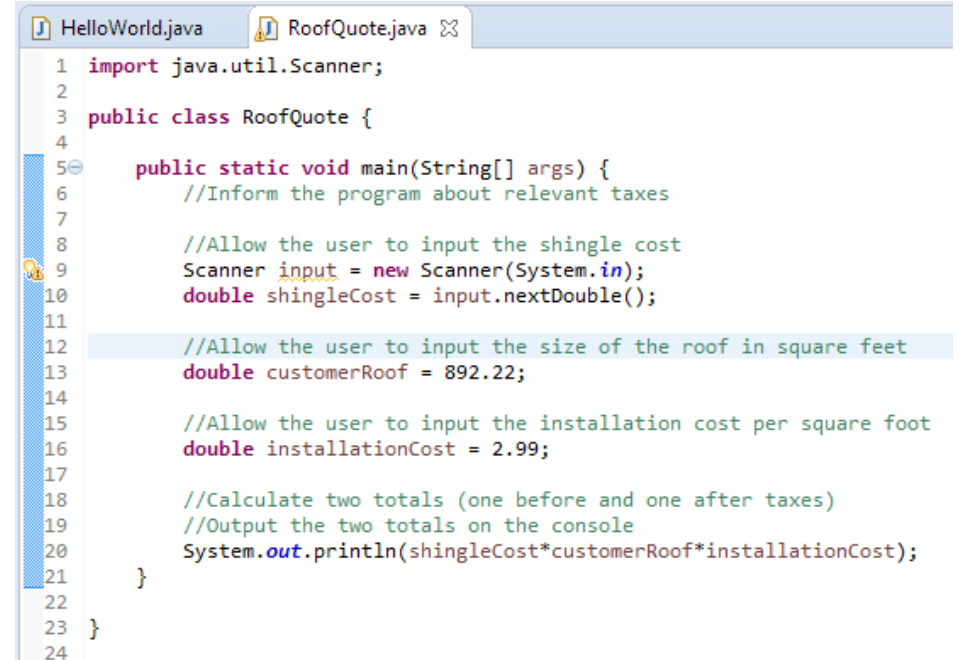
Java uses the scanner class to accept input from the console.

Make the changes in your code that you see on the right and then we will analyze the code

Programming Hint:

Java uses `System.out` to refer to the output device, The default output device is the monitor/console.

Java uses `System.in` to refer to the input device. The default input device is the keyboard.



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = 892.22;
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = 2.99;
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22 }
23
24
```

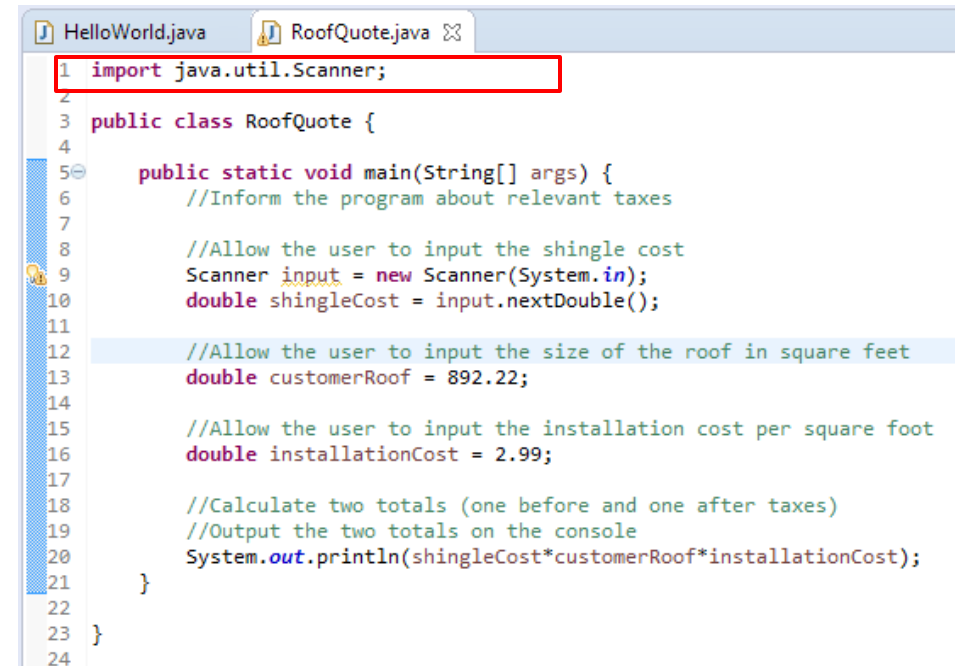

Step 1-3) Scanner Class

Line number 1 imports a class:

Java allows us to import in different libraries that may contain files or features we find useful.

In this case we import the Scanner class which is in the “java.util.Scanner” library

Importing the scanner class allows us to use a scanner object.



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = 892.22;
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = 2.99;
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22 }
23
24
```

Step 1-3)Scanner Class

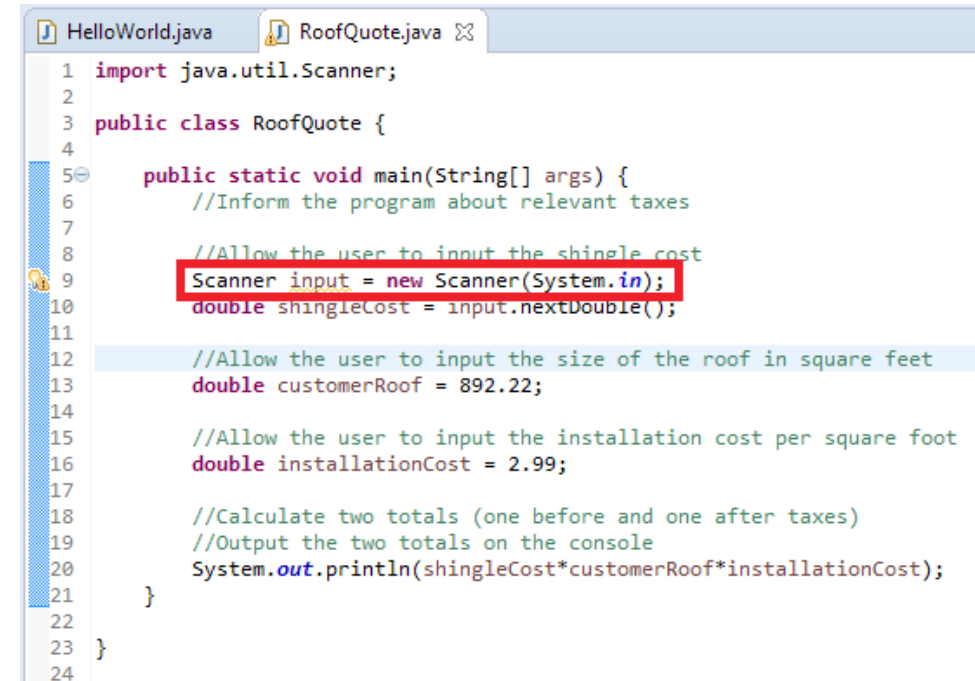
Line number 9 create a new scanner object:

A scanner is a data type much like an integer, double, String etc.

“**Scanner input**” creates a new variable named input of the type scanner

“**input = new Scanner(System.in)**” stores a scanner inside of the input variable.

We can now use this scanner to read from the System.in, which if you recall is the keyboard.



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = 892.22;
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = 2.99;
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22 }
23
24
```

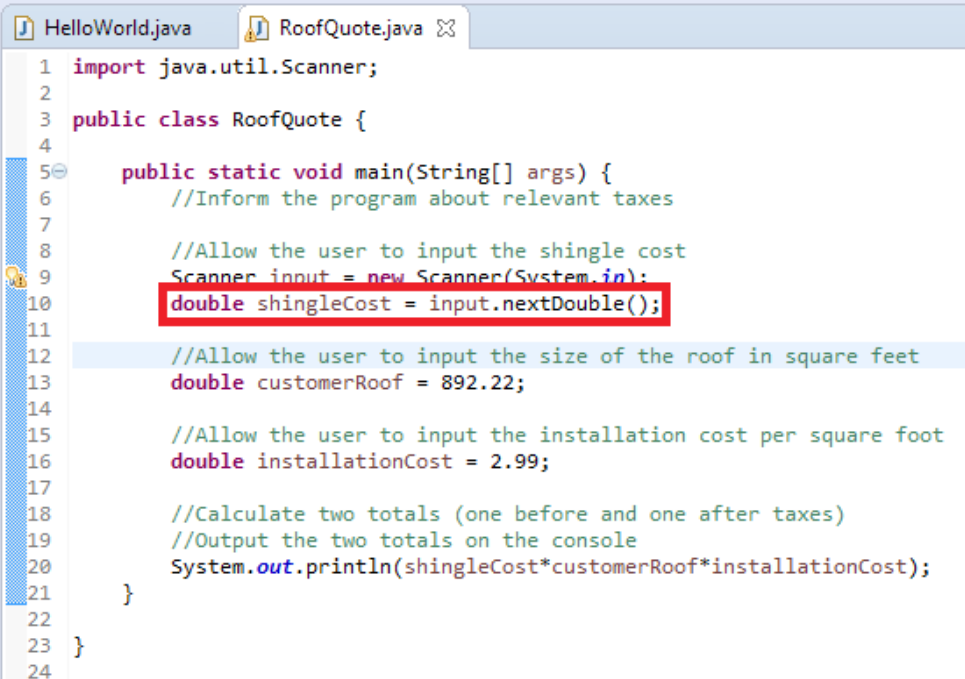
Step 1-3) Scanner Class

Line number 10 stores input into the shingleCost variable:

shingleCost is still a variable of data type double.

When we read input from the keyboard, we only want to store within shingleCost a double value.

input.nextDouble() reads input from the user and the next double entered will be put into shingleCost.



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = 892.22;
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = 2.99;
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22
23 }
24
```

Numeric Data Types

Reading numeric values from the keyboard:

Java has a range of numeric data types. Currently we have only learned about doubles, integers and floats.

Name	Range	Storage Size
Byte	-2^7 to 2^7	8-bit signed
Short	-2^{15} to 2^{15}	16-bit signed
Int	-2^{31} to 2^{31}	32-bit signed
Long	-2^{63} to 2^{63}	64-bit signed
Float	Refer to book	32-bit IEEE 754
Double	Refer to book	64-bit IEEE 754

Numeric Data Types

Reading numeric values from the keyboard:

Java has a range of numeric data types. Currently we have only learned about doubles, integers and floats.

Name	Method used to grab input
Byte	<code>input.nextByte()</code>
Short	<code>input.nextShort()</code>
Int	<code>input.nextInt()</code>
Long	<code>input.nextLong()</code>
Float	<code>input.nextFloat()</code>
Double	<code>input.nextDouble()</code>

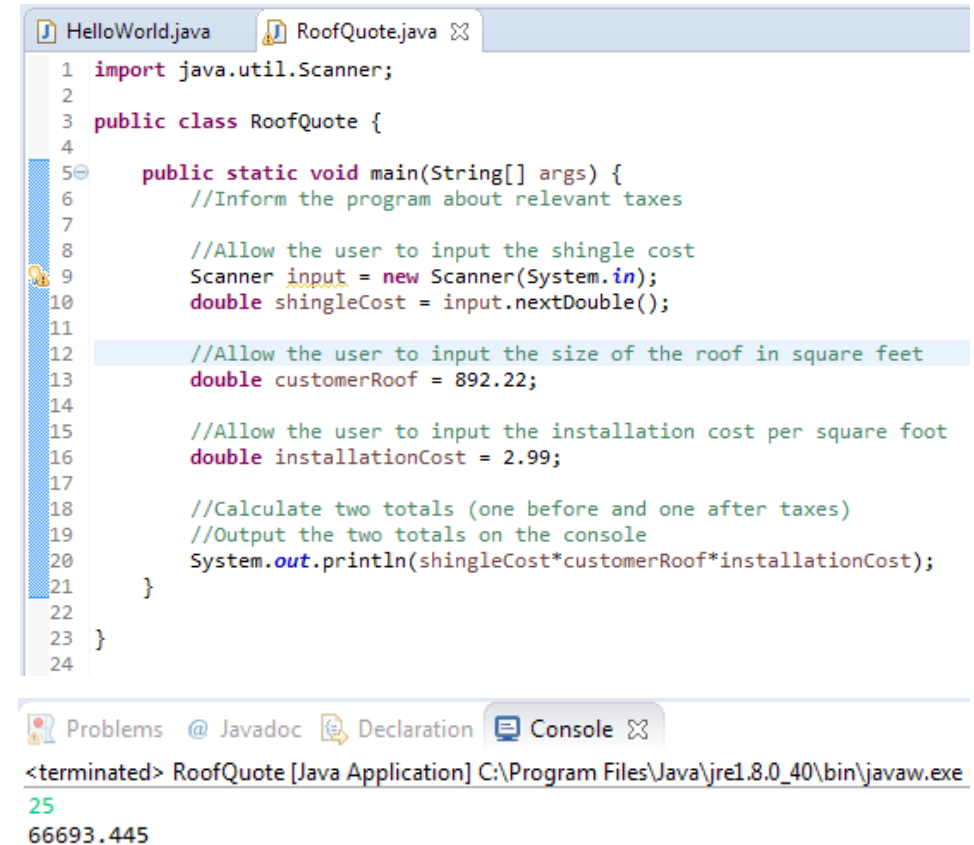
Step 1-3) Testing the program

Compile and run the program and enter a double value within the console:

- Once you have typed a double value within the console hit the enter key
- Try the value 25 and make sure you have the same output as me: 66693.445

We can reuse the scanner stored within the input variable for each of the values we want to retrieve from the keyboard

Try and use the input variable to retrieve and store a value within the customerRoof and installationCost variables.



The screenshot shows an IDE with two tabs: 'HelloWorld.java' and 'RoofQuote.java'. The 'RoofQuote.java' tab is active, displaying the following code:

```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = 892.22;
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = 2.99;
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22 }
23
24
```

At the bottom of the IDE, there is a 'Console' tab showing the output of the program:

```
<terminated> RoofQuote [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe
25
66693.445
```

Testing the program

Alter the code to retrieve inputs for the remaining variables:

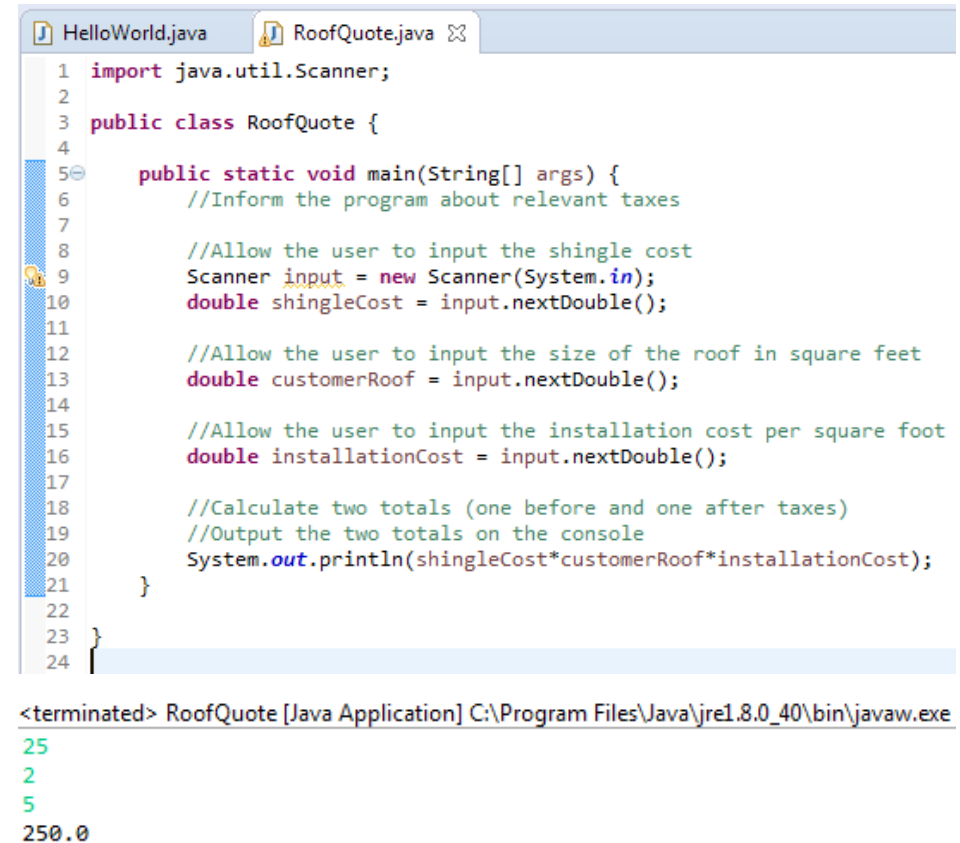
- `double customerRoof = input.nextDouble();`
- `double installationCost = input.nextDouble();`

Compile and run your software.

Confirm your software now accepts 3 inputs from the console.

Try the number 25, 2, and 5.

The output should be 250



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = input.nextDouble();
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = input.nextDouble();
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22 }
23
24
```

<terminated> RoofQuote [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe

25
2
5
250.0

Variable Naming Conventions

Camel Case:

Variable names should be done in a camel case fashion. Where each word after the first word in a variable name begins with a capital.

Ex) `thisIsCamelCase, anotherExampleOfCamelCase`

Bad Example) ~~`ThisIsNotCamelCase, NEITHERISTHIS, orTHIS`~~

Named Constants

A value that is constant is one that never changes.

- Within java we can also have named constants.
- Named constants should be placed at the top of the code within the main method
- Named constants should be named in all uppercase
- To make a value constant proceed its type declaration with a final keyword

```
final String BIRTH = "December 5th, 1980";
```

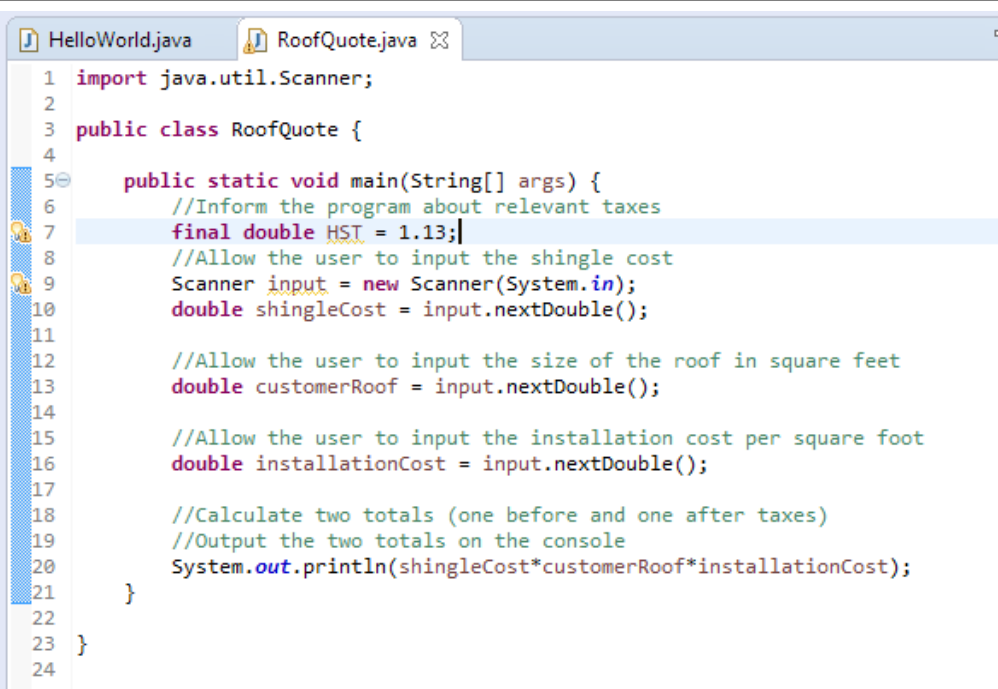
```
final double HST = 1.13;
```

Step 4)

Named constants are useful when we need to ensure the value never changes:

- Within our RoofQuote software the HST should be a constant value.
- As this value rarely if ever changes.
- Declare the named constant HST and set its value to 1.13

```
final double HST = 1.13;
```



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7         final double HST = 1.13;
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        double shingleCost = input.nextDouble();
11
12        //Allow the user to input the size of the roof in square feet
13        double customerRoof = input.nextDouble();
14
15        //Allow the user to input the installation cost per square foot
16        double installationCost = input.nextDouble();
17
18        //Calculate two totals (one before and one after taxes)
19        //Output the two totals on the console
20        System.out.println(shingleCost*customerRoof*installationCost);
21    }
22 }
23
24
```

Try the following:

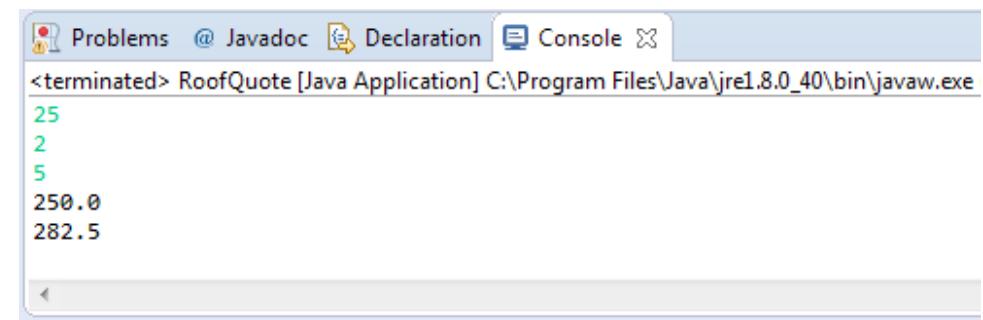
Input the values 25, 2 and 5.

Your first output should show 250.0 (*just like the first time we ran the software*)

Next you should see 282.5 (*which is the total with taxes*)

We have currently fulfilled the requirements set forth by Truss Goodman.

Before we send the software back what are some possible changes that we can foresee Truss asking us to make?



The screenshot shows a Java IDE window with a tab labeled 'Console'. The console output displays the results of a Java application named 'RoofQuote'. The output shows the input values 25, 2, and 5, followed by the calculated values 250.0 and 282.5. The window title bar indicates the application is running at 'C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe'.

```
<terminated> RoofQuote [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe
25
2
5
250.0
282.5
```

Issue #1: Undescriptive inputs

Let's examine a piece of the code:

Notice how on line 26 and 27 we did something like the following:

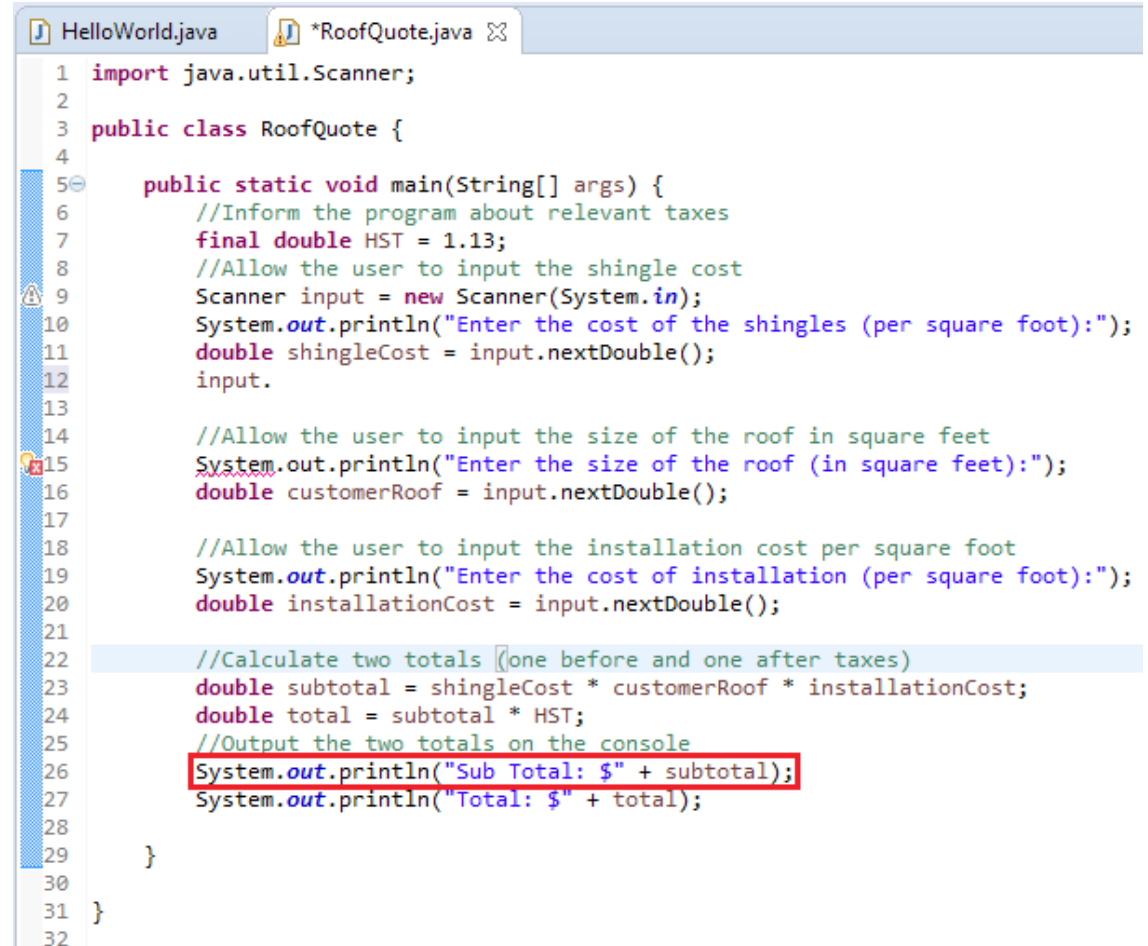
System.out.println("String" + double);

The **+** symbol is referred to as the **addition** and **concatenation** operator.

The **+** symbol can be used to add two numbers together (5 + 9) -> outputs 14

The **+** symbol can also be used to concatenate.

Concatenating is another way of adding but one of the values is a string and instead of using addition, it appends the value to the beginning or end of the string.



```
1 import java.util.Scanner;
2
3 public class RoofQuote {
4
5     public static void main(String[] args) {
6         //Inform the program about relevant taxes
7         final double HST = 1.13;
8         //Allow the user to input the shingle cost
9         Scanner input = new Scanner(System.in);
10        System.out.println("Enter the cost of the shingles (per square foot):");
11        double shingleCost = input.nextDouble();
12        input.
13
14        //Allow the user to input the size of the roof in square feet
15        System.out.println("Enter the size of the roof (in square feet):");
16        double customerRoof = input.nextDouble();
17
18        //Allow the user to input the installation cost per square foot
19        System.out.println("Enter the cost of installation (per square foot):");
20        double installationCost = input.nextDouble();
21
22        //Calculate two totals (one before and one after taxes)
23        double subtotal = shingleCost * customerRoof * installationCost;
24        double total = subtotal * HST;
25        //Output the two totals on the console
26        System.out.println("Sub Total: $" + subtotal);
27        System.out.println("Total: $" + total);
28
29    }
30
31 }
32
```

Augment Assignment Operator

Operator	Name	Description	Example (assume i is 1)
<code>++var</code>	Preincrement	Increment var by 1, use the new var value in the statement	<code>int j = ++i;</code> <i>// j is 2, i is 2</i>
<code>var++</code>	Postincrement	Increment var by 1, but use the original var value in the statement	<code>int j = i++;</code> <i>// j is 1 i is 2</i>
<code>--var</code>	Predecrement	decrement var by 1, and use the new var value in the statement	<code>int j = --i;</code> <i>// j is 0 i is 0</i>
<code>var--</code>	Postdecrement	Decrement var by 1, and use the original var value in the statement	<code>int j = i--;</code> <i>// j is 1 i is 0</i>

Marty A. Theodore

NEW CLIENT

Email

Marty A. Theodore

Grade 2 Teacher

Hello,

I need a piece of software to help my students check their answer on some basic math problems. Could you possibly build a piece of software that can do the following:

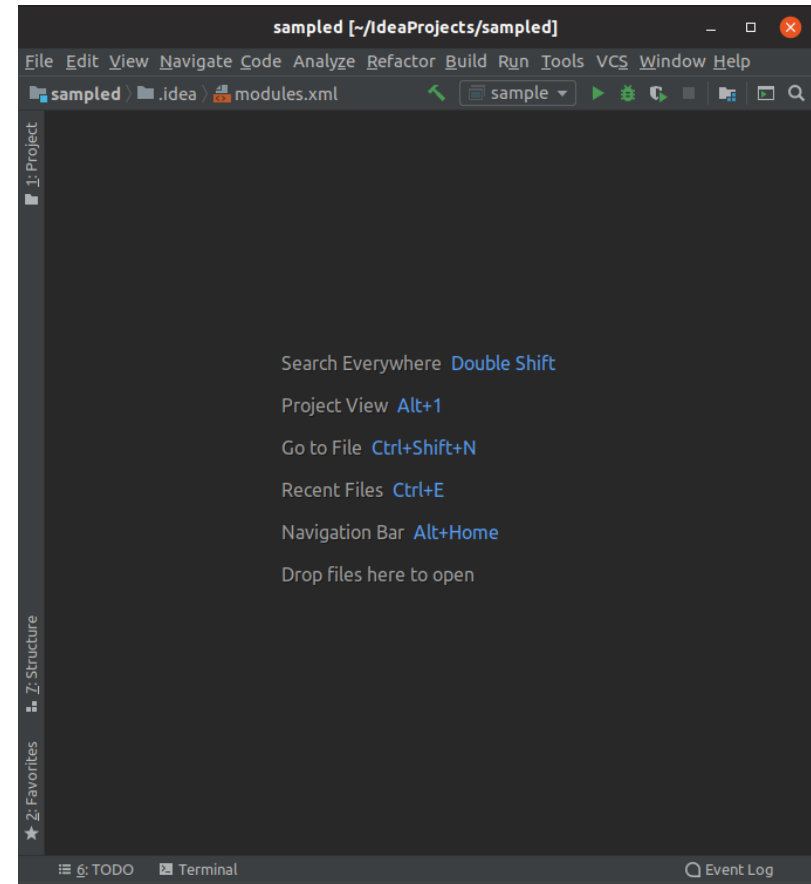
- Multiply a number by a number
- Divide a number by a number
- Subtract a number from a number
- Add one number to another number
- Calculate the mod of a number by another number

-Thanks

Step 1: Creating a new project

Start by opening IntelliJ and selecting the following:

File> New> Project



Step 1: New Project

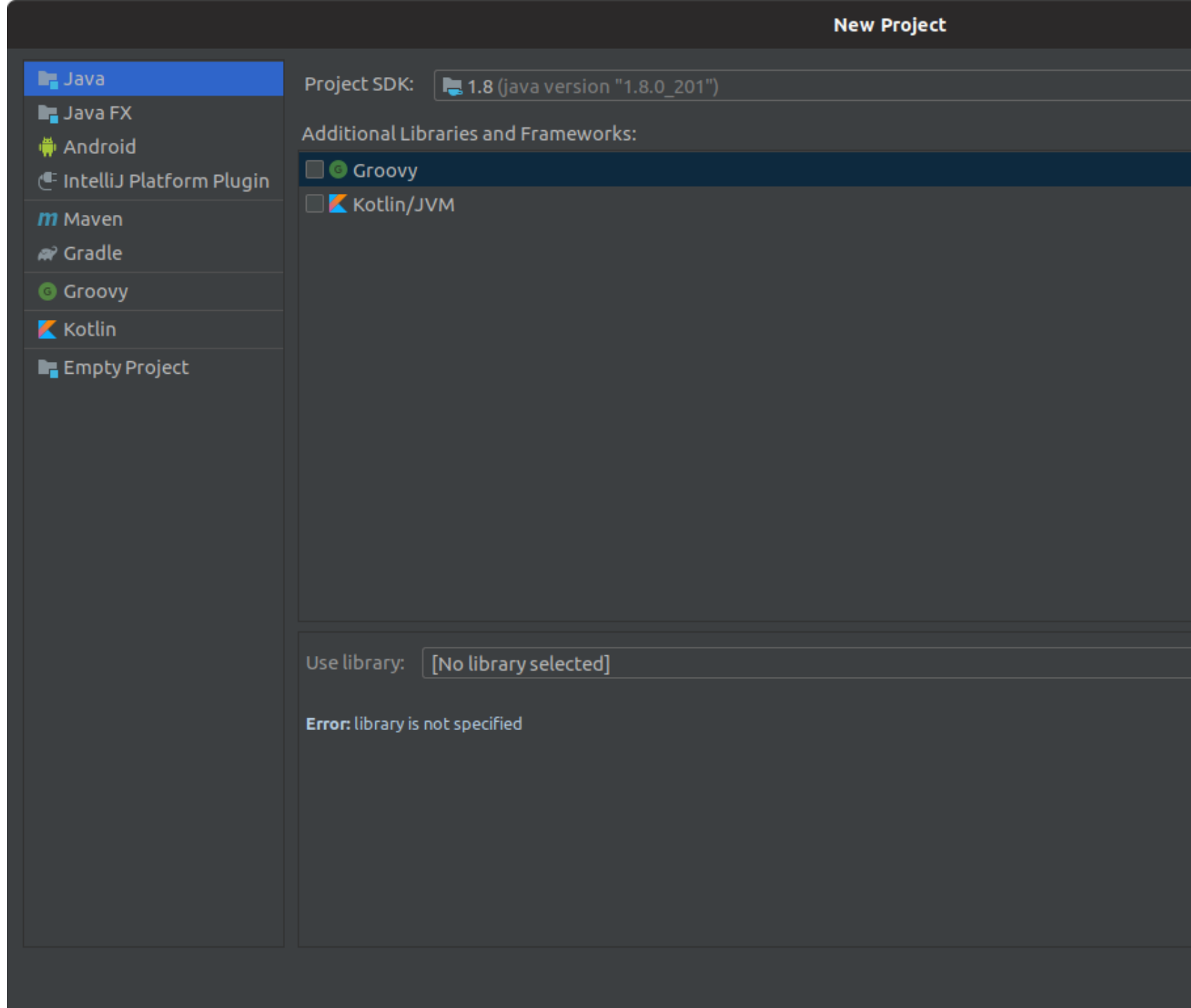
You will now see a dialog box appear *(It may be a little different than the one that I have shown)*

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



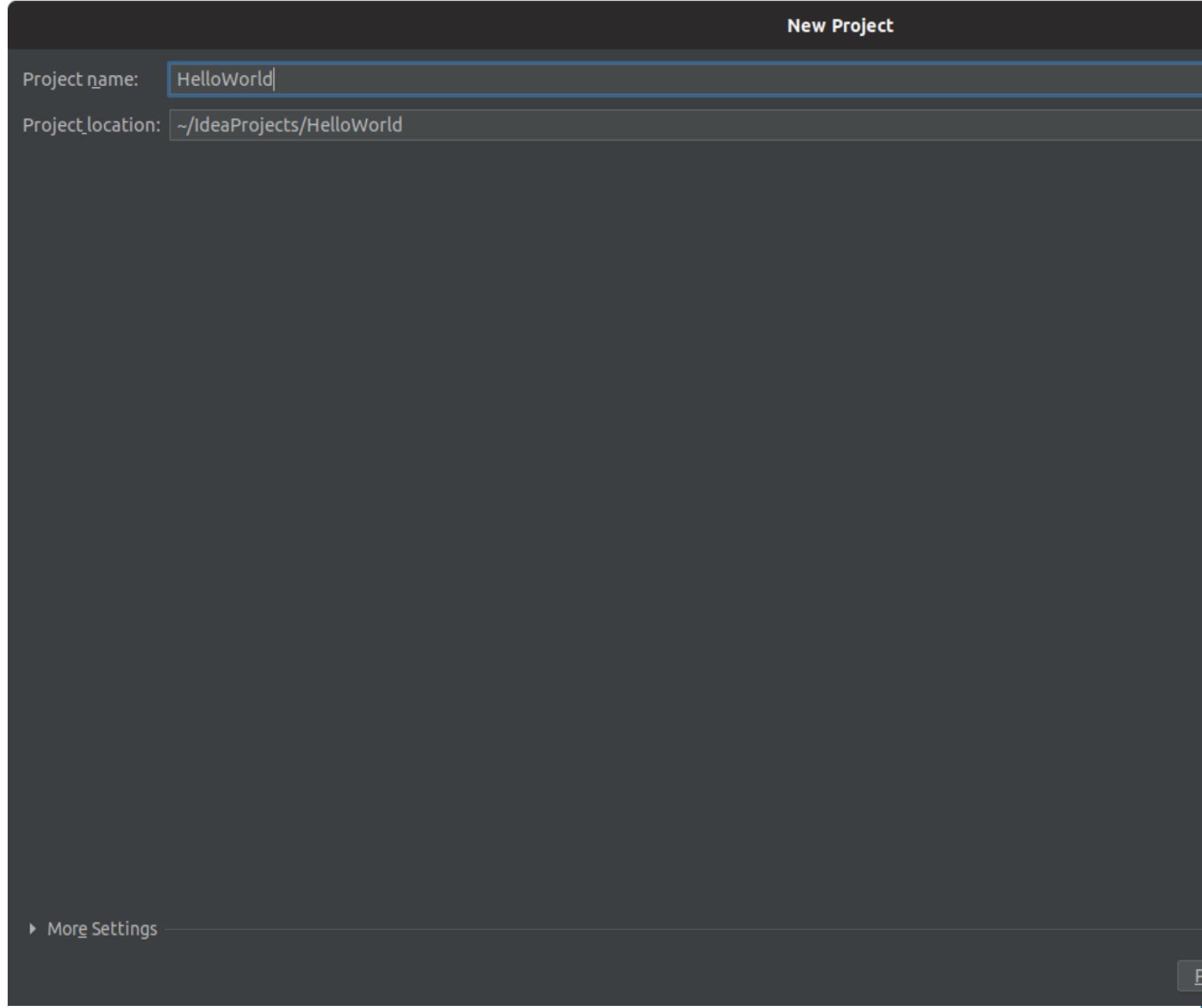
Step 1: New Project

Give the project a name of:
SimpleCalculator

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.

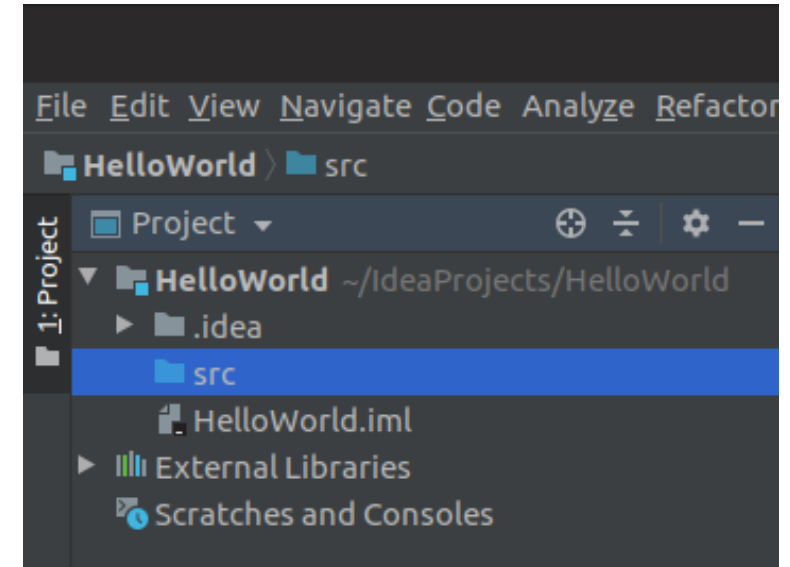


Step 1: Creating a new project

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

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

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



Step 2: Creating a new class file

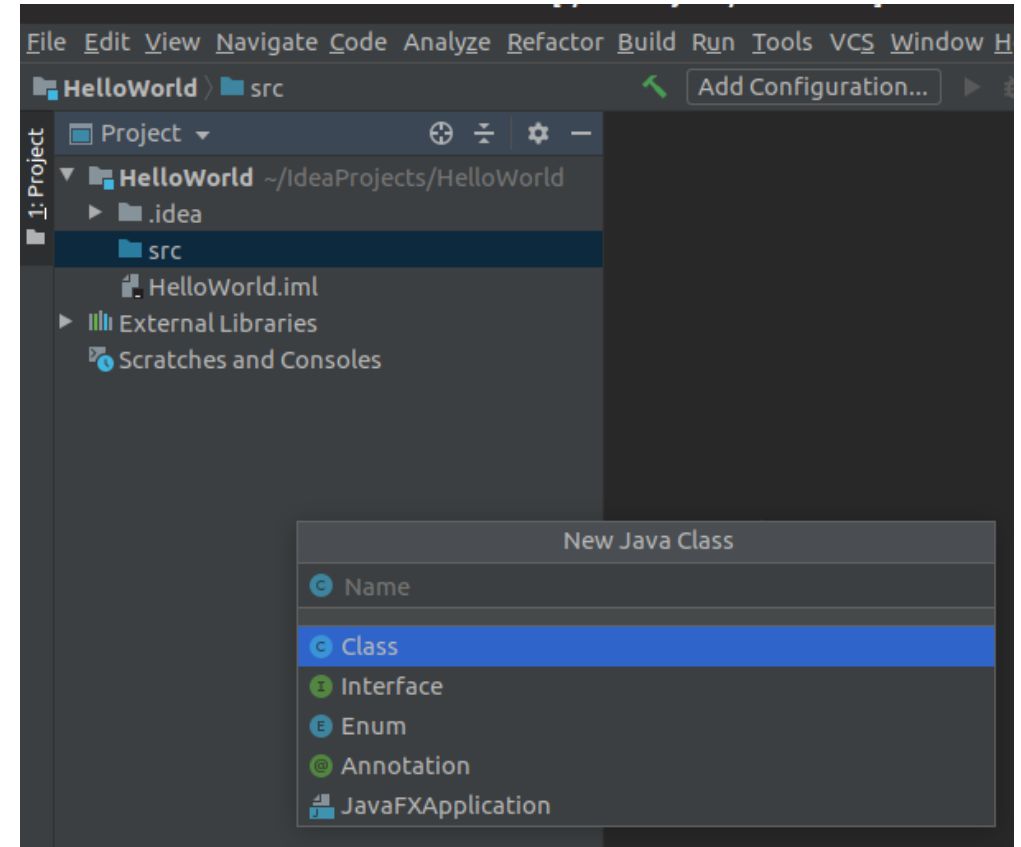
Next, we will need to create a new class file.

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

New> Java Class

Creating a new class is like creating a new program

Name the class **SimpleCalculator** and hit **enter**



Step 3: Writing the program

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

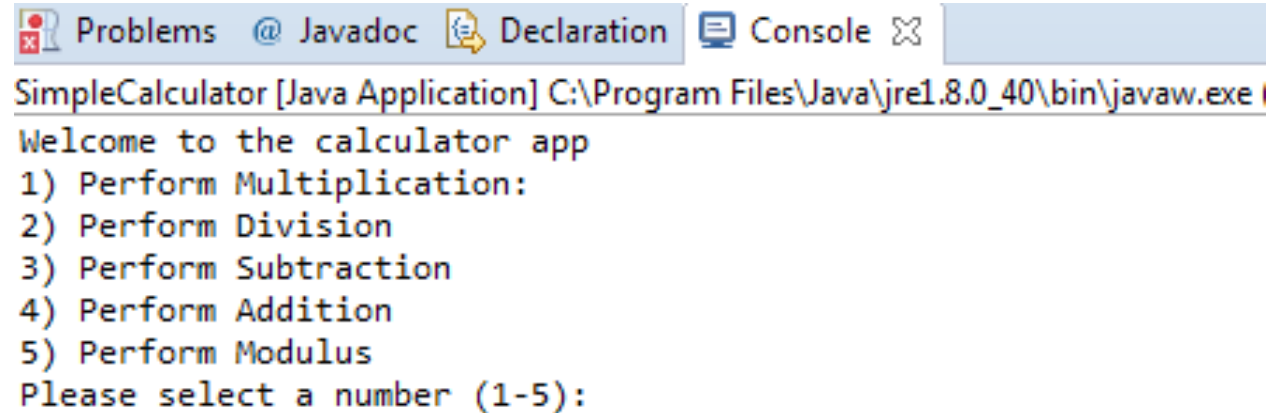
- Refer to the image on the right to see what your file should look like
- The first thing we are going to do is type "psvm" and hit the enter key
- This is what's known as "auto-completion"

```
1
2 public class SimpleCalculator1 {
3
4     public static void main(String[] args) {
5         // Display all possible forms of arithmetic
6         // Allow the user to input which form of arithmetic they would like to perform
7         // Allow the user to input two numbers
8         // perform the selected arithmetic on the inputted numbers
9         // Output the solution
10    }
11
12 }
13
```

Step 3: Writing the program

First, we will want to display to the screen all the different options that the user will have

Take a second and try writing the code that will give the output seen on the right



```
Problems @ Javadoc Declaration Console
SimpleCalculator [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe
Welcome to the calculator app
1) Perform Multiplication:
2) Perform Division
3) Perform Subtraction
4) Perform Addition
5) Perform Modulus
Please select a number (1-5):
```

Step 3: Writing the program

Your first guess may be to do the following:

```
System.out.println("Welcome to the calculator app");
```

```
System.out.println("1) Perform Multiplication");
```

```
System.out.println("2) Perform Division");
```

```
System.out.println("3) Perform Subtraction");
```

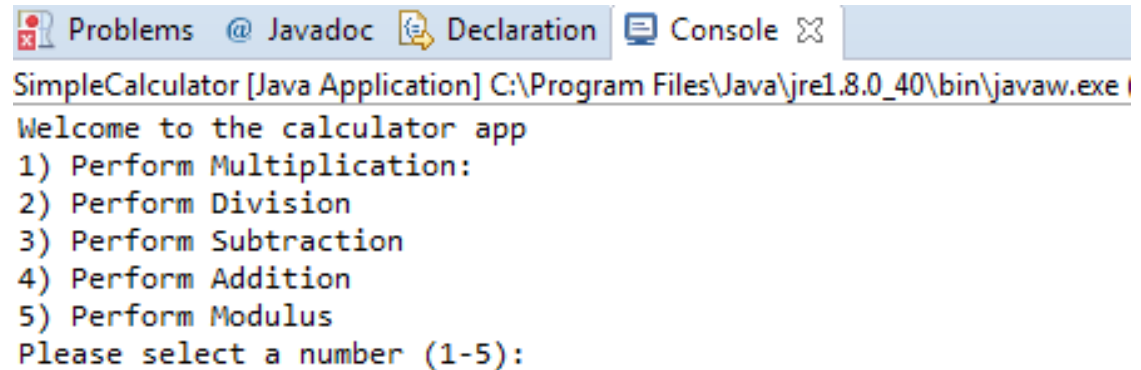
```
System.out.println("4) Perform Addition");
```

```
System.out.println("5) Perform Modulus");
```

```
System.out.println("Please Select A Number:");
```

This answer is not incorrect, but it is very inefficient.

Let's look at another way this can be done.



The screenshot shows a Java IDE window titled 'SimpleCalculator [Java Application]'. The 'Console' tab is active, displaying the following output:

```
SimpleCalculator [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe l
Welcome to the calculator app
1) Perform Multiplication:
2) Perform Division
3) Perform Subtraction
4) Perform Addition
5) Perform Modulus
Please select a number (1-5):
```

Escape Sequences

A character that is preceded by a backslash is called an escape sequence and has a specific purpose when it occurs within a string.

Escape Sequence	Description
<code>\t</code>	Insert a tab
<code>\b</code>	Insert a backspace
<code>\n</code>	Insert a newline
<code>\r</code>	Insert a carriage return
<code>\f</code>	Insert a form feed
<code>\' OR \"</code>	Insert a single quote or double quote
<code>\\</code>	Insert a backslash

Step 3: Writing the program

Using escape sequences

```
1
2 public class SimpleCalculator {
3
4     public static void main(String[] args) {
5         // Display all possible forms of arithmetic
6         System.out.println("Welcome to the calculator app\n"
7             + "1) Perform Multiplication:\n"
8             + "2) Perform Division\n"
9             + "3) Perform Subtraction\n"
10            + "4) Perform Addition\n"
11            + "5) Perform Modulus \n"
12            + "Please select a number (1-5):");
13         // Allow the user to input which form of arithmetic they would like to perform
14         // Allow the user to input two numbers
15         // perform the selected arithmetic on the inputted numbers
16         // Output the solution
17     }
18 }
19
20
```

Step 3: Writing the program

Concatenation transforms:

- "Welcome to the calculator app\n"
- "1) Perform Multiplication\n"
- "2) Perform Division\n"
- "3) Perform Subtraction\n"
- "4) Perform Addition\n"
- "5) Perform Modulus\n"

INTO

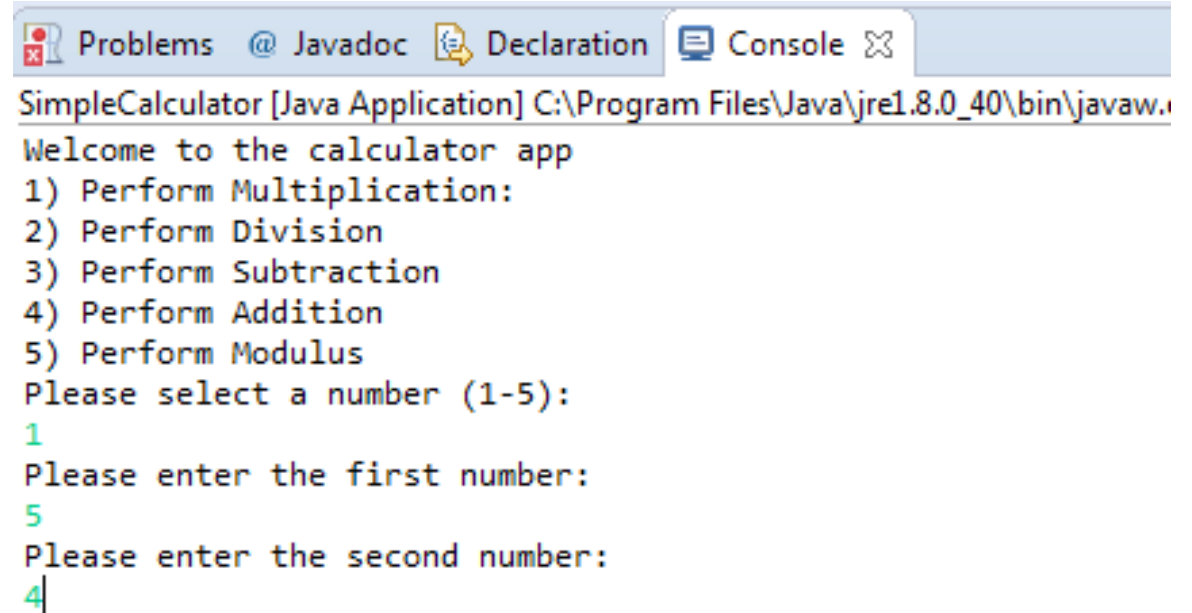
"Welcome to the calculator app \n
1) Perform Multiplication\n
2) Perform Division\n
3) Perform Subtraction\n
4) Perform Addition\n
5) Perform Modulus \n"

```
1
2 public class SimpleCalculator {
3
4     public static void main(String[] args) {
5         // Display all possible forms of arithmetic
6         System.out.println("Welcome to the calculator app\n"
7             + "1) Perform Multiplication:\n"
8             + "2) Perform Division\n"
9             + "3) Perform Subtraction\n"
10            + "4) Perform Addition\n"
11            + "5) Perform Modulus \n"
12            + "Please select a number (1-5):");
13         // Allow the user to input which form of arithmetic they would like to perform
14         // Allow the user to input two numbers
15         // perform the selected arithmetic on the inputted numbers
16         // Output the solution
17     }
18 }
19
20
```

Step 3: Writing the program

Allow the user to input two numbers

Use the concepts we have just learned to allow the program to accept two inputs



```
SimpleCalculator [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.
Welcome to the calculator app
1) Perform Multiplication:
2) Perform Division
3) Perform Subtraction
4) Perform Addition
5) Perform Modulus
Please select a number (1-5):
1
Please enter the first number:
5
Please enter the second number:
4
```

Step 3: Writing the program

Concatenation transforms:

- "Welcome to the calculator app\n"
- "1) Perform Multiplication\n"
- "2) Perform Division\n"
- "3) Perform Subtraction\n"
- "4) Perform Addition\n"
- "5) Perform Modulus\n"

INTO

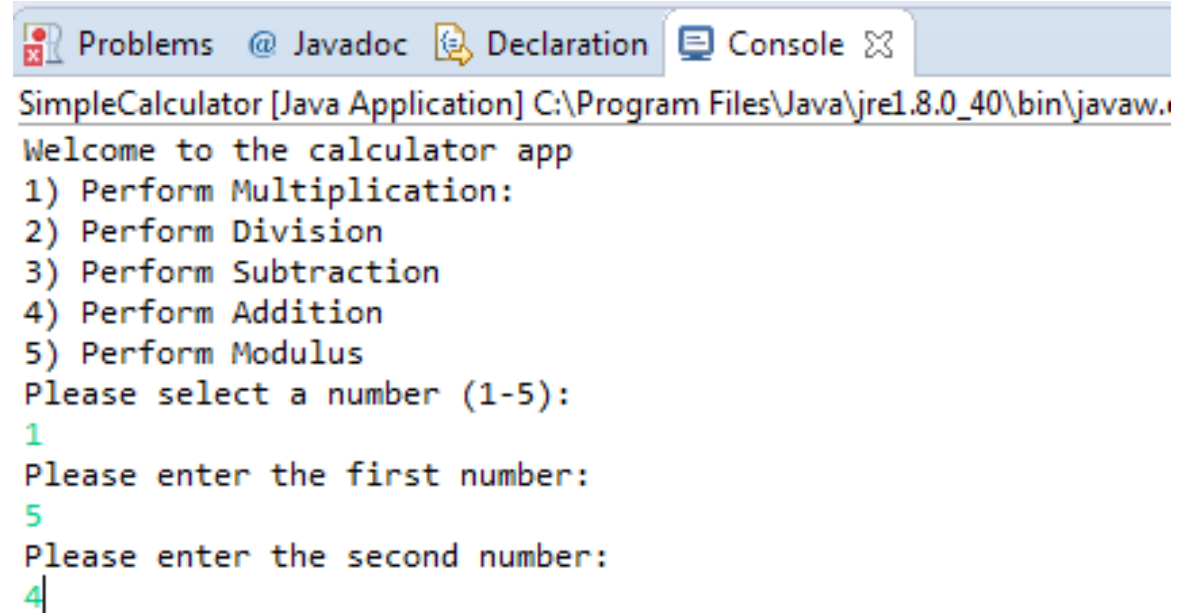
"Welcome to the calculator app \n
1) Perform Multiplication\n
2) Perform Division\n
3) Perform Subtraction\n
4) Perform Addition\n
5) Perform Modulus \n"

```
1
2 public class SimpleCalculator {
3
4     public static void main(String[] args) {
5         // Display all possible forms of arithmetic
6         System.out.println("Welcome to the calculator app\n"
7             + "1) Perform Multiplication:\n"
8             + "2) Perform Division\n"
9             + "3) Perform Subtraction\n"
10            + "4) Perform Addition\n"
11            + "5) Perform Modulus \n"
12            + "Please select a number (1-5):");
13         // Allow the user to input which form of arithmetic they would like to perform
14         // Allow the user to input two numbers
15         // perform the selected arithmetic on the inputted numbers
16         // Output the solution
17     }
18 }
19
20
```

Step 3: Writing the program

Allow the user to input two numbers

Use the concepts we have just learned to allow the program to accept two inputs



```
SimpleCalculator [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.
Welcome to the calculator app
1) Perform Multiplication:
2) Perform Division
3) Perform Subtraction
4) Perform Addition
5) Perform Modulus
Please select a number (1-5):
1
Please enter the first number:
5
Please enter the second number:
4
```

Step 3: Writing the program

Step 3: Allow the user to input two numbers

- Line numbers 8 and 9 show the creation of variables that will store the first and second number to be calculated.
- Line 22 and 24 are messages displayed to the screen that inform the user what numbers they are about to enter (first or second)
- Finally lines 23 and 25 grab the two double values from the user and record them in the variables `firstNumber` and `secondNumber`

```
1 import java.util.Scanner;
2
3
4 public class SimpleCalculator {
5
6     public static void main(String[] args) {
7         int choice;
8         double firstNumber;
9         double secondNumber;
10        Scanner input = new Scanner(System.in);
11        // Display all possible forms of arithmetic
12        System.out.println("Welcome to the calculator app\n"
13            + "1) Perform Multiplication:\n"
14            + "2) Perform Division\n"
15            + "3) Perform Subtraction\n"
16            + "4) Perform Addition\n"
17            + "5) Perform Modulus \n"
18            + "Please select a number (1-5):");
19        // Allow the user to input which form of arithmetic they
20        choice = input.nextInt();
21        // Allow the user to input two numbers
22        System.out.println("Please enter the first number:");
23        firstNumber = input.nextDouble();
24        System.out.println("Please enter the second number:");
25        secondNumber = input.nextDouble();
26        // perform the selected arithmetic on the inputted numbers
27        // Output the solution
28    }
29
30 }
```

Perform the correct arithmetic

This step is broken down into multiple sub steps:

- Create a variable to store the solution of the calculation
 - What data type should the solution be and why?
- Write out each possible calculation and store its solution
- Determine a way the program will only execute the correct calculation

Step 3: Writing the program

Sub Step 1:

- Solution should be the data type double because both of the inputs are double and could produce a number with a remainder
- Insert a variable named solution as shown on line 10

```
1  import java.util.Scanner;
2
3
4  public class SimpleCalculator {
5
6      public static void main(String[] args) {
7          int choice;
8          double firstNumber;
9          double secondNumber;
10         double solution;
11         Scanner input = new Scanner(System.in);
12         // Display all possible forms of arithmetic
13         System.out.println("Welcome to the calculator app\n"
14             + "1) Perform Multiplication\n"
15             + "2) Perform Division\n"
16             + "3) Perform Subtraction\n"
17             + "4) Perform Addition\n"
18             + "5) Perform Modulus \n"
19             + "Please select a number (1-5):");
20         // Allow the user to input which form of arithmetic
21         choice = input.nextInt();
```


Step 3: Writing the program

Sub Step 2:

- Your code should look something like picture provided
- What is wrong with the way we currently done our calculations?

Hint: What still needs to be done?

```
1  int choice;  
2  
3  double firstNumber;  
4  double secondNumber;  
5  double solution;  
6  Scanner input = new Scanner(System.in);  
7  // Display all possible forms of arithmetic  
8  System.out.println("Welcome to the calculator app\n"  
9      + "1) Perform Multiplication\n"  
10     + "2) Perform Division\n"  
11     + "3) Perform Subtraction\n"  
12     + "4) Perform Addition\n"  
13     + "5) Perform Modulus \n"  
14     + "Please select a number (1-5):");  
15  
16 // Allow the user to input which form of arithmetic they  
17 choice = input.nextInt();  
18 // Allow the user to input two numbers  
19 System.out.println("Please enter the first number:");  
20 firstNumber = input.nextDouble();  
21 System.out.println("Please enter the second number:");  
22 secondNumber = input.nextDouble();  
23 // perform the selected arithmetic on the inputed numbers  
24 solution = firstNumber * secondNumber;  
25 solution = firstNumber / secondNumber;  
26 solution = firstNumber - secondNumber;  
27 solution = firstNumber + secondNumber;  
28 solution = firstNumber % secondNumber;  
29 // Output the solution  
30  
31 }  
32  
33 }  
34  
35 }  
36  
37 }
```

Switch Statements

Determine a way the program will only execute the correct calculation

Before we can do this, we need to learn about a new concept:

- The Switch Case statement
- The switch case statement to select an outcome when all possible outcomes are known.

Consider the following example:

- Say we wanted to make a program that would output the number of days in the current month.
- We know there are only twelve possible months so we could have the following code:

Switch Statements

```
String month;
switch (month) {
    case "January":
        System.out.println("31");
        break;
    case "February":
        System.out.println("28"); //we are ignoring leap years for the sake of this example
        break;
    case "March":
        System.out.println("31");
        break;
    ...this would continue for the rest of the months
}
```

Step 3: Writing the program

- Your switch statement should look something like the picture provided
- Earlier in the code we asked the individual to select which calculation they wanted to perform and stored that number inside the variable choice
- Lines 29, 32, 35, 38, 41 all check to see if it is values 1-5.
- Finally we see on line 44 “default:”
- This is known as a base case and will be matched if no other value is matched.
- Notice that we take no action and break out of the switch statement

```
22 // Allow the user to input two numbers
23 System.out.println("Please enter the first number:");
24 firstNumber = input.nextDouble();
25 System.out.println("Please enter the second number:");
26 secondNumber = input.nextDouble();
27 // perform the selected arithmetic on the inputted numbers
28 switch (choice) {
29     case 1:
30         solution = firstNumber * secondNumber;
31         break;
32     case 2:
33         solution = firstNumber / secondNumber;
34         break;
35     case 3:
36         solution = firstNumber - secondNumber;
37         break;
38     case 4:
39         solution = firstNumber + secondNumber;
40         break;
41     case 5:
42         solution = firstNumber % secondNumber;
43         break;
44     default:
45         break;
46 }
47 // Output the solution
```

Step 3: Writing the program

Output the solution:

- When outputting the solution to screen we want it to be in a format that shows the whole equation and the solution.
- Ex) $4 + 8 = 12$
- This means if addition took place, we want to see what was added and the solution
- The simplest way to complete this is to add an additional statement to each of the switch statement options.
- Refer to the next slide for a demonstration

Step 3: Writing the program

- Notice on lines 31, 35, 39, 43, and 47 we have outputted the appropriate solution for each corresponding choice.

```
27 // perform the selected arithmetic on the inputted numbers
28 switch(choice){
29     case 1:
30         solution = firstNumber * secondNumber;
31         System.out.println(firstNumber + " * " + secondNumber + " = " + solution);
32         break;
33     case 2:
34         solution = firstNumber / secondNumber;
35         System.out.println(firstNumber + " / " + secondNumber + " = " + solution);
36         break;
37     case 3:
38         solution = firstNumber - secondNumber;
39         System.out.println(firstNumber + " - " + secondNumber + " = " + solution);
40         break;
41     case 4:
42         solution = firstNumber + secondNumber;
43         System.out.println(firstNumber + " + " + secondNumber + " = " + solution);
44         break;
45     case 5:
46         solution = firstNumber % secondNumber;
47         System.out.println(firstNumber + " % " + secondNumber + " = " + solution);
48         break;
49     default:
50         break;
51 }
52 // Output the solution
53 }
54
55 }
```

Test your software

Take a moment to test your software

Truss Goodman

OUR FIRST CLIENT: BACK AGAIN

Email

Truss Goodman

Local business owner

Owns a roofing business

Roof Quoting

Hello,

I am having the following issues with the software:

I hand wrote out all the numbers and found that the software is not billing the customers the appropriate amount of money

-Thanks

Truss

Types of errors

Before we can diagnose the issue with the software, we need understand a few concepts about programming:

1. The types of errors
2. The effects of those errors

Types of errors

Errors are common in programming, so much so that if you google “comic errors in programming” you will get thousands of images full of common errors that people face

The types of errors encountered by a programmer usually fall into one of the following categories:

- Syntax Errors
- Logic Errors
- Run time errors

Syntax Errors

- When the programmer codes something that does not follow the syntax of the language.
- This is one of the most common errors when programming.
- A syntax error could be missing a curly brace { or a semicolon ; at the end of the line.

Logic Errors

- Are when the code complies and executes correctly but does not give the desired output.
- A good example of this is if we were to have $2 + 2$ and the output was not 4

Runtime errors

- Occur when the users inputs an unexpected value.
- For instance if the user inputs a character when we are expecting a double.
- Another example would be if the user inputs an integer, but it was one we were not expecting. Think back to when we built the calculator app for Marty.
- Dividing by zero

What type of error?

Understanding the types of errors can help us find the issues with the software built for Truss Goodman

What type of error do you think is within the roof quoting software?

Truss Goodman

- The software was calculating inaccurately the total cost to complete a roof
- This is a logic error
- Let's fix the calculation so we are outputting the correct roof cost.

```
5 public class RoofQuote {
6
7     public static void main(String[] args) {
8         //Inform the program about relevant taxes
9         final double HST = 1.13;
10        //Allow the user to input the shingle cost
11        Scanner input = new Scanner(System.in);
12        System.out.println("Enter the cost of the shingles (per square foot):");
13        double shingleCost = input.nextDouble();
14
15        //Allow the user to input the size of the roof in square feet
16        System.out.println("Enter the size of the roof (in square feet):");
17        double customerRoof = input.nextDouble();
18
19        //Allow the user to input the installation cost per square foot
20        System.out.println("Enter the cost of installation (per square foot):");
21        double installationCost = input.nextDouble();
22
23        //Calculate two totals (one before and one after taxes)
24        double subtotal = shingleCost * customerRoof * installationCost;
25        double accurateSubtotal = (customerRoof * installationCost) + (customerRoof * shingleCost);
26        double total = subtotal * HST;
27        double accurateTotal = accurateSubtotal * HST;
28        //Output the two totals on the console
29        System.out.println("Sub Total: $" + subtotal);
30        System.out.println("Total: " + total);
31        System.out.println("Accurate Sub Total: $" + accurateSubtotal);
32        System.out.println("Accurate Total: " + accurateTotal);
33
34    }
35
36 }
```


Homework

Read Pages 120-148 of your textbook

Next Week

- Boolean Data Type
- Selection Structures
- If statement
- Nested if
- If-else statement
- Boolean conditions
- Relational Operators
- Math Library – Random Numbers
- Logic Operators
- Conditional Expressions