

# Coding Your First Program

by Sophia



## WHAT'S COVERED

In this lesson, you will learn about how to think like a developer and what's involved in the development process. Specifically, this lesson covers:

### Table of Contents

- [1. Mentalism Program](#)
- [2. Validating and Debugging](#)

## 1. Mentalism Program

For this first program, you will use various parts of a program that has been discussed to this point. Your goal will be to create a program that can perform a mentalism magic trick in Java.



### TRY IT

**Directions:** Let's first try it out on you, no programming yet! For now, consider the individual steps of the program:

Hello! What is your name?

Now, we'll perform some mind reading on you.

First, think of a number between 1 and 10.

Multiply the result by 2.

Now, add 8 to it.

Now, divide the number that you have by 2.

Now, subtract the original number that you thought about.

Alright, let's do some processing now.

Is the number that you now have a 4?



#### REFLECT

Wow! If you didn't get the number 4, it might help to go through the steps carefully again. Magic tricks normally have a basis for how they work, and they also usually have restrictions on how they are viewed. Still stuck? Maybe it will be clearer to you if you think about this as an equation. Let  $X$  be the number between 1 and 10. If you put all the math in an equation it will look like this:  $((X * 2) + 8) / 2 - X$ . Can you simplify this? Step 1 would be to simplify it to  $(X + 4) - X$ , and step 2 will give you 4. So, the basis is that regardless of the value of  $X$  you choose, the result will be 4, or more generally the result (4 in this case) will be half the value added (8 in this case). The main restriction here is that the number added has to be an even number.



#### HINT

It is important to note that the number that you first came up with doesn't have any effect on the result at all. The only number that mattered was the number that was added since the result is half that number. An important part to ensure that we have nice round numbers is that the number we add should be an even number. All the rest of the calculation was an illusion to hide what was being done in the program.



#### THINK ABOUT IT

Consider what that program should look like. It is important to define what inputs we have, as a starting point. You will want to ask the person's name. In this case, you don't want to ask the person for the number, as that doesn't affect the process directly. What is needed is to generate a random number that is even for the individual to add.

Processing will be similar to what you have done previously using random number generation. For simplicity, generate a random number between 1 and 5 that number will be the final answer. This number multiplied by 2 is the number that the user will have to add. Multiplication by 2 will make it an even number. The output will need to let the user know what number they currently have.



#### TRY IT

**Directions:** In Replit, add a new file named `Mentalism.java`. Carefully type in the following code:

```
import java.util.Random; // Needed for random number generation
import java.util.Scanner; // Needed for reading user input
public class Mentalism {
    public static void main(String[] args) {
        // Create random number generator
        Random randomGenerator = new Random();
        final int maxValue = 5;
        final int minValue = 1;
        // Generate random number in range from 1 to 5
        int numberToGuess = randomGenerator.nextInt(maxValue) + minValue;
        int numberToAdd = numberToGuess * 2;
        // Scanner named keyboardInput used to read input
```

```
Scanner keyboardInput = new Scanner(System.in);
```

```
// Use System.out.print() to display prompt text
System.out.print("Hello. What is your name? ");
// Read input as a String (which may contain spaces)
String name = keyboardInput.nextLine();
// System.out.println() adds new line (\n) at end of output
System.out.println("Welcome, " + name + ", we'll perform some mind reading.");
System.out.println("Think of a number between 1 and 10.");
System.out.print("Hit Enter/Return when ready for the next step.");
// Read input as String but don't save it, since it's blank
keyboardInput.nextLine();
```

```
System.out.println("Multiply the number by 2.");
System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
```

```
System.out.println("Now add " + numberToAdd);
System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
```

```
System.out.println("Now, divide the number by 2.");
System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
```

```
System.out.println("Now, subtract the original number that you thought of.");
System.out.print("Hit Enter/Return when ready for the last step.");
keyboardInput.nextLine();
```

```
System.out.print("Well, " + name + " let me read your mind... ");
System.out.println("The number that you have now is " + numberToGuess + ".");
}
```

```
}
```

These are the results of a sample run:

```

> java Mentalism.java
Hello. What is your name? Sophia
Welcome, Sophia, we'll perform some mind reading.
Think of a number between 1 and 10.
Hit Enter/Return when ready for the next step.
Multiply the number by 2.
Hit Enter/Return when ready for the next step.
Now add 10
Hit Enter/Return when ready for the next step.
Now, divide the number by 2.
Hit Enter/Return when ready for the next step.
Now, subtract the original number that you thought of.
Hit Enter/Return when ready for the last step.
Well, Sophia let me read your mind... The number that you have now is 5.
> 

```



## REFLECT

You might be asking, why are we generating a random number for this. It's mostly to add a bit of mystery to the output. If you always ask the user to add 8 to the code, the result will always be 4. To make the program seem like it is reading your mind you need a bit of randomness in the response.

## 2. Validating and Debugging

The next step in this process is to validate that the calculations are correct. Adding some functionality to the program to perform the calculations will help to ensure that it matches. One new bit of functionality to add is to ask the user to enter a number, instead of just thinking of it. Next, store this entered number in a new variable called `enteredNumber`. Another new task to aid validation is to add a check after each calculation. Be sure to output the result after each one. The process of testing and fixing these errors is called **debugging**. The two terms are ones that you'll hear used quite often. **Testing** is normally just to identify that there are bugs, whereas debugging is going through the process of fixing those bugs. Outputting variables as you progress is a very useful way to debug code.



## TRY IT

**Directions:** Let's try this again. Enter the following code to Replit in a file named `MentalismDebug.java`. You can copy and paste if you are in a hurry, but we recommend that you take the time to type it into Replit yourself so you can practice catching and addressing errors.

```

import java.util.Random;
import java.util.Scanner;

public class MentalismDebug {
    public static void main(String[] args) {
        Random randomGenerator = new Random();

```

```

// final indicates that the value doesn't - indeed can't - change.
// These are constants rather than variables.
final int maxValue = 5;
final int minValue = 1;
int numberToGuess = randomGenerator.nextInt(5) + minValue;
int numberToAdd = numberToGuess * 2;

System.out.print("Hello. What is your name? ");
Scanner keyboardInput = new Scanner(System.in);
String name = keyboardInput.nextLine();
System.out.print("Welcome, " + name + ", we'll perform some mind reading.\n");
// Following changed to have user enter number
System.out.print("Enter a number between 1 and 10: ");
int enteredNumber = keyboardInput.nextInt();
// The following code is needed to remove leftover new line in input stream
keyboardInput.nextLine();

System.out.print("Hit Enter/Return when ready for the next step.");
String response = keyboardInput.nextLine();
System.out.println("Multiply the number by 2.");
// Check result of multiplying user's number by 2
int userNumber = enteredNumber * 2;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
response = keyboardInput.nextLine();
System.out.println("Now add " + numberToAdd);
userNumber = userNumber + numberToAdd;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
response = keyboardInput.nextLine();
System.out.println("Now, divide the number by 2.");
userNumber = userNumber / 2;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
response = keyboardInput.nextLine();
System.out.println("Now, subtract the original number that you thought of.");
userNumber = userNumber - enteredNumber;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
response = keyboardInput.nextLine();
System.out.print("Well, " + name + " let me read your mind... ");
System.out.println("The number that you have now is " + numberToGuess + ".");

```

```
}  
}
```

This is the result:

```
Console Shell  
  
❏ java MentalismDebug.java  
Hello. What is your name? Sophia  
Welcome, Sophia, we'll perform some mind reading.  
Enter a number between 1 and 10: 5  
Hit Enter/Return when ready for the next step.  
Multiply the number by 2.  
>> userNumber at this step = 10  
Hit Enter/Return when ready for the next step.  
Now add 2  
>> userNumber at this step = 12  
Hit Enter/Return when ready for the next step.  
Now, divide the number by 2.  
>> userNumber at this step = 6  
Hit Enter/Return when ready for the next step.  
Now, subtract the original number that you thought of.  
>> userNumber at this step = 1  
Hit Enter/Return when ready for the next step.  
Well, Sophia let me read your mind... The number that you have now is 1.  
❏
```

## REFLECT

Outputting the variables as you program for testing purposes can be very helpful for debugging the logic as you go. At each step, you should output the current calculation so that you can verify at the end that the logic is correct.

Once we are done, we can simply comment out the lines of code.

## TRY IT

**Directions:** Using Replit, make sure you comment out the following highlighted lines of code so your testing code does not produce output that might confuse the end user. After commenting the debugging statements out, run the program again to make sure it still runs after the changes.

```
import java.util.Random;  
import java.util.Scanner;
```

```
public class MentalismDebug {  
    public static void main(String[] args) {  
        Random randomGenerator = new Random();  
        // final indicates that the value doesn't - indeed can't - change.  
        // These are constants rather than variables.
```

```

final int maxValue = 5;
final int minValue = 1;
int numberToGuess = randomGenerator.nextInt(5) + minValue;
int numberToAdd = numberToGuess * 2;

System.out.print("Hello. What is your name? ");
Scanner keyboardInput = new Scanner(System.in);
String name = keyboardInput.nextLine();
System.out.print("Welcome, " + name + ", we'll perform some mind reading.\n");
// Following changed to have user enter number
System.out.print("Enter a number between 1 and 10: ");
int enteredNumber = keyboardInput.nextInt();
// The following code is needed to remove leftover new line in input stream
keyboardInput.nextLine();

System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
System.out.println("Multiply the number by 2.");
// Check result of multiplying user's number by 2
int userNumber = enteredNumber * 2;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
System.out.println("Now add " + numberToAdd);
userNumber = userNumber + numberToAdd;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
System.out.println("Now, divide the number by 2.");
userNumber = userNumber / 2;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
System.out.println("Now, subtract the original number that you thought of.");
userNumber = userNumber - enteredNumber;
System.out.println(">> userNumber at this step = " + userNumber);

System.out.print("Hit Enter/Return when ready for the next step.");
keyboardInput.nextLine();
System.out.print("Well, " + name + " let me read your mind... ");
System.out.println("The number that you have now is " + numberToGuess + ".");
}
}

```



## REFLECT

Another approach to debugging and testing could be to add the code at the bottom of the program so that the code is all combined together.



## TRY IT

**Directions:** Try removing the testing code from the main program and add it to the bottom like you see highlighted below. The code listing is for a file named `JavaValidate.java`.

```
import java.util.Random;
import java.util.Scanner;

public class MentalismValidate {
    public static void main(String[] args) {
        Random randomGenerator = new Random();
        // final indicates that the value doesn't - indeed can't - change.
        // These are constants rather than variables.
        final int maxValue = 5;
        final int minValue = 1;
        int numberToGuess = randomGenerator.nextInt(5) + minValue;
        int numberToAdd = numberToGuess * 2;

        System.out.print("Hello. What is your name? ");
        Scanner keyboardInput = new Scanner(System.in);
        String name = keyboardInput.nextLine();
        System.out.print("Welcome, " + name + ", we'll perform some mind reading.\n");

        System.out.print("Hit Enter/Return when ready for the next step.");
        keyboardInput.nextLine();
        System.out.println("Multiply the number by 2.");

        System.out.print("Hit Enter/Return when ready for the next step.");
        keyboardInput.nextLine();
        System.out.println("Now add " + numberToAdd);

        System.out.print("Hit Enter/Return when ready for the next step.");
        keyboardInput.nextLine();
        System.out.println("Now, divide the number by 2.");

        System.out.print("Hit Enter/Return when ready for the next step.");
        keyboardInput.nextLine();
        System.out.println("Now, subtract the original number that you thought of.");

        System.out.print("Hit Enter/Return when ready for the next step.");
        keyboardInput.nextLine();
        System.out.print("Well, " + name + " let me read your mind... ");
```



```
System.out.println("The number that you have now is " + numberToGuess + ".");
```

```
// Validation code
```

```
System.out.print("Enter the number guessed between 1 and 10: ");
```

```
int enteredNumber = keyboardInput.nextInt();
```

```
// The following code is needed to remove leftover new line in input stream
```

```
keyboardInput.nextLine();
```

```
// Check result of multiplying user's number by 2
```

```
int userNumber = enteredNumber * 2;
```

```
System.out.println("Multiplied by 2 = " + userNumber);
```

```
// Confirm result of adding
```

```
userNumber = userNumber + numberToAdd;
```

```
System.out.println("Plus " + numberToAdd + " = " + userNumber);
```

```
// Check the result of division by 2
```

```
userNumber = userNumber / 2;
```

```
System.out.println("Divided by 2 = " + userNumber);
```

```
// Confirm difference produced by final subtraction
```

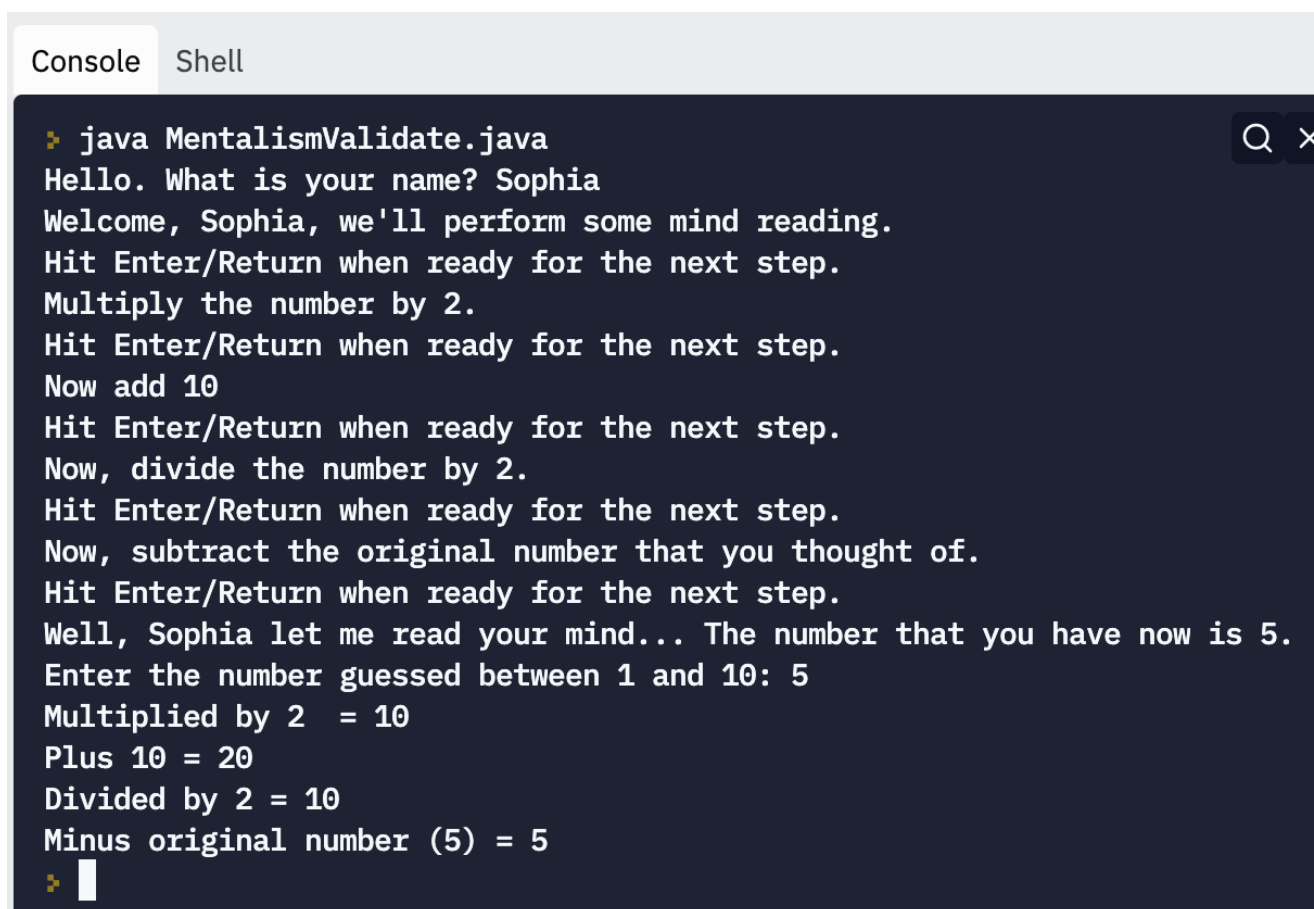
```
userNumber = userNumber - enteredNumber;
```

```
System.out.println("Minus " + "original number (" + enteredNumber + ") = " +  
    userNumber);
```

```
}
```

```
}
```

This version of the program should produce results like this:



```
Console Shell
> java MentalismValidate.java
Hello. What is your name? Sophia
Welcome, Sophia, we'll perform some mind reading.
Hit Enter/Return when ready for the next step.
Multiply the number by 2.
Hit Enter/Return when ready for the next step.
Now add 10
Hit Enter/Return when ready for the next step.
Now, divide the number by 2.
Hit Enter/Return when ready for the next step.
Now, subtract the original number that you thought of.
Hit Enter/Return when ready for the next step.
Well, Sophia let me read your mind... The number that you have now is 5.
Enter the number guessed between 1 and 10: 5
Multiplied by 2 = 10
Plus 10 = 20
Divided by 2 = 10
Minus original number (5) = 5
>
```

Admiral Grace Hopper coined the term debugging while fixing a computer way back in the 1940's. She and her team actually found a moth stuck to a relay within the computer as the cause of a problem so the term debug is actually based on a real bug. Hopefully you won't find any moths within your computer but you will run into programming syntax and logic errors. Debugging your code systematically will help to make this process seamless and thorough.



#### TERMS TO KNOW

##### Debugging

The process of testing and fixing errors in the code.

##### Testing

Identify errors, or bugs, in the code.



#### SUMMARY

In this final lesson of Challenge 1.2, you thought like a developer while reviewing the **Mentalism Program**. You then added code snippets of the program to Replit to test individually. Finally, you performed tasks to **validate and debug** the program by outputting variables to the screen as the program progressed through the code.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source [cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf](https://cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf)

It has also been adapted from "Python for Everybody" By Dr. Charles R. Severance. Source [py4e.com/html3/](https://py4e.com/html3/)



#### TERMS TO KNOW

##### Debugging

The process of testing and fixing errors in the code.

##### Testing

Identify errors, or bugs, in the code.