Guessing Game

Introduction

In this exercise you'll write a command line application that allows users to play a "guess my number" game, but this time, in Java!

Setup

Download the materials for this exercise here.

Step 1: Check Out Lab Materials

If you ran ***ls*** after ***cd***-ing into ***~/src/guessing-game***, you already know that the folder contains one file — ***Game.java***. Let's get the lay of the land and see what it does.

One way to get a feel for what a Java file does is to run it! You'll always run code in your preferred command line using the ***javac*** and ***java*** commands:

```
$ javac Game.java
$ java Game
```

…nothing happened! What gives?

```
$ cat Game.java
/*

A number-guessing game.

*/

public class Game {
  public static void main(String[] args){
    // Put your code here.
  }

}
```

Now it makes sense that nothing happened when we tried to run ***Game***. There isn't anything exciting going on in this file going on in this file.

Step 2: Hello World

In VS Code, add a print statement to ***Game.java***. You can print whatever you want (you’ll remove this line of code later) but whatever you do, **don’t save the file yet**. Your code should look something like this:

```
public class Game {
  public static void main(String[] args){
    System.out.println("Hello world");
  }
}
```

We need to **compile** this class again, because the underlying sourcecode has changed. This is what you should see when you run the following commands.

```
  $ javac Game.java
  $ java Game
 Hello World
```

Step 3: Build Out Guessing Game

About the Guessing Game

For the rest of this exercise, you'll write a program in the ***Game.java*** file that plays the classic number guessing game. The game will work like this:

- The computer will choose a random number between 1–100 and ask the user to guess the number
- Once the user guesses, the computer will tell the user if their guess was too high or too low
- The game is over once the user guesses the computer's number. When the game is over, the computer prints the total number of guesses the user made before getting the right answer.

Here's the terminal output for an example game:

```
$ java Game
Howdy, what's your name?
(type in your name) Jessica
Jessica, I'm thinking of a number between 1 and 100.
Try to guess my number.
Your guess? 50
Your guess is too low, try again.
Your guess? 80
Your guess is too high, try again.
Your guess? 60
Your guess is too low, try again.
Your guess? 70
Your guess is too high, try again.
Your guess? 63
Your guess is too low, try again.
Your guess? 64
Your guess is too low, try again.
Your guess? 67
Your guess is too low, try again.
Your guess? 69
Your guess is too high, try again.
Your guess? 68
Well done, Jessica! You found my number in 9 tries!
```

Here's a rough pseudocode outline of the program that gave this output:

```
greet player
get player name
choose random number between 1 and 100
repeat forever:
    get guess
    if guess is incorrect:
        give hint

        increase number of guesses
    else:
        congratulate player
```

Greet the Player

The first feature to implement is the greeting:

- In VS Code, add code to ***Game.java*** that displays a greeting to the player
- Since you've added new code, re-compile and run it again to test that it works
- It's a good time to ***git commit*** 💾

Get the Player's Name

After "greet player," our pseudocode outline says "get player name".

- Write code in ***Game.java*** that get the player's name and save it to a variable
- Run your code to test it. Remember to re-compile your code before running, with `javac`.
- Once everything works, make another commit.

The next thing you'll do is choose a random number.

Choose a Random Number

Java comes with a ***Random*** module that contains functions for generating random numbers. To access the code in a module, all you have to do is import it. You can put ***import*** statements at the top of your file like this:

Game.java

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

To import any module, all you have to do is write `import` followed by the name of the module. After that, you can use any function included in the module like so:

```
// create an instance of the Random object
Random rand = new Random();

// call method nextInt and pass in upper range bound
int number = rand.nextInt(100);
```

Now that you know how to import functions from modules, check out the official docs on the ***Random*** library here: JDK 16 Documentation: Random.

Keep Going!

Keep adding code to ***Game.java***!

Along the way, you may need to learn about how to capture an integer from the user.

While we do recommend using the ***Scanner.nextInt()*** method, you could also capture user input as a string, and cast the string into an integer. This is closer how we would do this in Python. If you want to learn more about how to do this, you can read the docs for ***Integer.parseInt()*** or read this short tutorial.

Notably, in Java, `break` and `continue` work the same as they do in Python, but feel free to do some googling to see some examples of these statements in Java.

Once you have a game that…

- Asks for the user's name
- Greets the user and prompts them to guess a number
- Cycles through until the user guesses the right number
- Ends when the user guesses the right number and displays the number of guesses that the user made

… feel free to move on to the next sections.

Step 4: Check for Faulty User Input

Confirm that the User’s Guess Falls Within Your Range

If the user inputs a number that isn’t between 1-100 as requested, mock them for their crimes and ask them to enter a valid number.

Make Sure the User’s Guess is an Integer

If the user inputs something that is not an integer, mock them for their crimes and ask them to enter a valid integer. If you’re not sure where to start with this feature, check the hint below.

Hint: Validating user input

▼ *Click to hide*

A good way forward is to learn what happens when you *don’t* enter a number. Then, learn about Java’s ***try*** and ***catch*** statements. They are very similar to Python’s ***try*** and ***except*** statements.

Notably, if you want to use `continue` in the ***catch*** block to progress to another iteration after the bad input, you need to add an additional line to your ***catch*** block to progress your ***Scanner*** past the bad input as well.

So, your catch block should contain the following lines:

```
String bad_input = input.next();    // need to progress past bad input
System.out.println("That's not an integer, try again");
continue;
```

Further Study

If you have more time, you can click here to continue to the Further Study