Number Guessing Game in Java

Number Guessing Game or "**Guess A Number**" is a very simple and short JavaScript gaming mini project. This game is built for students who are looking for mini-games built in Java to learn and practice some basic Java tools they're familiar with.

The complete source code for this game is given below with a step-by-step description. You can copy it from there or you can download the code from this link.

Before going through the steps, here's an outline of the general rules of the game:

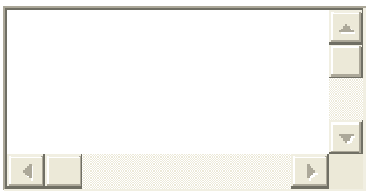
- The system or computer will generate a random number from 1 to 100.
- A dialogue box is displayed where user is asked to enter their **guess number**.
- Computer tells if the **guess number** matches or it is higher/lower than the one it generated.
- The game continues until the user guesses the computer number.

Step 1: Calling Class & Main Function

First, we're going to call a class **GuessingGame** and add empty main function as follows:

Number Guessing Game: Calling Class & Main Function

Java



```
1 public class GuessingGame {
2     public static void main(String[] args) {
3     }
4 }
```

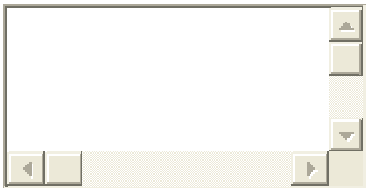
With only these lines, the program is completely valid; you can compile and run, but it doesn't display anything to the console yet.

Step 2: Computer Number

To generate a number which will be later guessed by the user, let's declare an integer-type variable **computerNumber** and use this instruction: *(Math.random()*100 + 1)*to assign it a random number in the range of 1 to 100.

Number Guessing Game: Computer Number

Java



```
1 public class GuessingGame {
2     public static void main(String[] args) {
3     int computerNumber = (int) (Math.random()*100 + 1);
4     System.out.println("The correct guess would be " + computerNumber);
5     }
6 }
```

The fourth line shows the random number to user at the moment, but this line is not printed upon running of the final version of this game. For now, this line simply logs correct answer to the console for verification.

Step 3: User Answer

Now, the random number generated by the computer is to be guessed by the user. In order to get answer from the user, we declare another int variable **userAnswer**and initialize it.

Step 4: Add Number of Attempts

This is very simple and you can do it by initializing an int variable **count**: *int count = 1*. This additionally displays the input dialog box until the user guesses the right number.

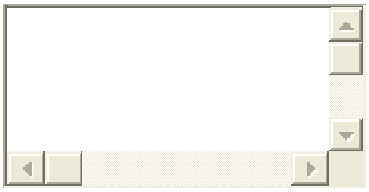
Step 5: Check User Answer:

It's quite obvious that the user cannot be given only one attempt to guess the number in this game. So, we need to give the user as many attempts as they need and the number guessed in all attempts is to be checked. Counting the number of attempts is already done in earlier step.

Now, the answer input by the user is checked with the computer's random number using **while loop** starting with this code: *while (userAnswer != computerNumber)*. The bulk of code under the "while" loop is explained below:

- The 3rd line, beginning with "*String response* =", displays initial input dialog box at the console.
- The next line converts string to integer for use in check method below.
- The next line passes **userAnswer** and **computerNumber** along with **count** to**determineGuess**.
- Count++ is for increment in number of tries for each attempt.

Number Guessing Game: Check User Answer
Java



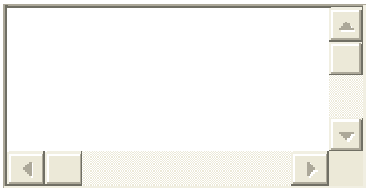
```
1 while (userAnswer != computerNumber)
2       {
3           String response = JOptionPane.showInputDialog(null,
4               "Enter a guess between 1 and 100", "Guessing Game", 3);
5           userAnswer = Integer.parseInt(response);
6           JOptionPane.showMessageDialog(null, ""+ determineGuess(userAnswer, computerNumber, count));
7           count++;
8       }
```

Final Step:

As arguments are passed from **while** loop to **determineGuess**, we need to check how close the number guessed by the user is to computer generated number and display the number of attempts made. There are five conditional statements that will be executed based on the number input by the user.

Number Guessing Game: Adding Conditions

Java

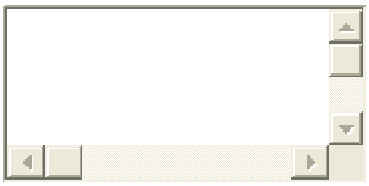


```
1  public static String determineGuess(int userAnswer, int computerNumber, int count){
2        if (userAnswer <=0 || userAnswer >100) {
3           return "Your guess is invalid";
4        }
5        else if (userAnswer == computerNumber ){
6           return "Correct!\nTotal Guesses: " + count;
7        }
8        else if (userAnswer > computerNumber) {
9           return "Your guess is too high, try again.\nTry Number: " + count;
10       }
11       else if (userAnswer < computerNumber) {
12          return "Your guess is too low, try again.\nTry Number: " + count;
13       }
14       else {
15          return "Your guess is incorrect\nTry Number: " + count;
16
17       }
18 }
```

Number Guessing Game Source Code:

Guess A Number Java Game

Java



```
1  package guessinggame;
2
3  /** www.codewithc.com
4   * Java game “Guess a Number” that allows user to guess a random number that has been generated.
5   */
6
7  import javax.swing.*;
8
9  public class GuessingGame {
10    public static void main(String[] args) {
11        int computerNumber = (int) (Math.random()*100 + 1);
12        int userAnswer = 0;
13        System.out.println("The correct guess would be " + computerNumber);
14        int count = 1;
15        while (userAnswer != computerNumber)
16        {
17            String response = JOptionPane.showInputDialog(null,
18                "Enter a guess between 1 and 100", "Guessing Game", 3);
19            userAnswer = Integer.parseInt(response);
20            JOptionPane.showMessageDialog(null, ""+ determineGuess(userAnswer, computerNumber, count));
21            count++;
22        }
23    }
```

```
24
25    public static String determineGuess(int userAnswer, int computerNumber, int count){
26        if (userAnswer <=0 || userAnswer >100) {
27            return "Your guess is invalid";
28        }
29        else if (userAnswer == computerNumber ){
30            return "Correct!\nTotal Guesses: " + count;
31        }
32        else if (userAnswer > computerNumber) {
33            return "Your guess is too high, try again.\nTry Number: " + count;
34        }
35        else if (userAnswer < computerNumber) {
36            return "Your guess is too low, try again.\nTry Number: " + count;
37        }
38        else {
39            return "Your guess is incorrect\nTry Number: " + count;
40
41        }
42    }
43 }
```

Possible Improvements:

If you'd like to practice developing a similar number guessing game on your own, you can incorporate further details such as:

- limiting the number of attempts
- adding more than one round
- displaying win/loss score
- giving points to the user based on the number of attempts, etc.

Any questions and suggestions relevant to this game can be brought up to us from the comments box below. Also, try answering this question: "How many attempts the user needs at most to guess the number?