Assignment 1 C212 FA22

Last modified 2022-09-02 5:00 PM

Summary

The gist of this assignment document's contents are as follows:

- Program Description
  - Sample program outputs
- Tutorial on random numbers in Java
- Deliverables & Reminders
  - Use a project named `Assignment1`, and a class named `GuessingGame`
  - Submit a .zip file containing, at minimum, `GuessingGame.java`
- Rubric
  - Also visible on Canvas assignment page
- Changelog
  - No updates yet…

Program Description

Write a program, `GuessingGame.java` - a guessing game for two players where both players guess a number in between 1 and 100. Your program should generate the "secret number" randomly. The winner is determined as follows:

- If one player's guess is closer than the other player, then that player wins.
  - This is the "general case", in other words, the outcome that happens most often.
- If both players guess the same number, the first player wins.
- If both players' guesses are off by the same value, then the player that guessed the lower number wins.
  - This, and the bullet point before it, are some "edge cases" of the program – outcomes that are unlikely but require special program logic to handle correctly.

For example, if the secret number is 71 and the two players guesses are 64 and 77, then the player that guesses 77 wins, since 77 - 71 = 6 and 71 - 64 = 7. Alternatively, if the secret number is 45 and the two players guesses are 36 and 54, then the player that guesses 36 wins, since 45 -36 = 9, 54 - 45 = 9, and 36 is less than 54.

The program should:

1. First, ask for the two players' names.
2. Then, generate the secret number, and ask the players to guess it.
3. Finally, it should print out who the final winner was and why.

Sample program outputs

Sample run

```
Enter player 1 name: John
Enter player 2 name: Adam
Hmmmmm, let me think of a number between 1 and 100
Alright, I have thought of a number.
John, enter your guess: 77
Adam, enter your guess: 64
The secret number was 71
John had 77 with a difference of 77 - 71 = 6
Adam had 64 with a difference of 71 - 64 = 7
John wins!!!
```

Another sample run:

```
Enter player 1 name: Sarah
Enter player 2 name: Justin
Hmmmmm, let me think of a number between 1 and 100
Alright, I have thought of a number.
Sarah enter your guess: 36
Justin enter your guess: 54
The secret number was 45
Sarah had 36 with a difference of 45 - 36 = 9
Justin had 54 with a difference of 54 - 45 = 9
Since both scores are equal, the winner is the one who guessed the lower number
wins, and that happens to be Sarah.
Sarah wins!!!
```

Tutorial on random numbers in Java

The statement "pick a number between 1 and 10" isn't very specific. Does that mean I can choose 1? Or 10? What about 2.5? Or π?

In a previous math course, you should have been introduced to **interval notation** for numbers. This is a very useful way of describing exactly which numbers we want to include in some range of numbers, and they work with both integers and numbers with decimal components, which we call floating-point numbers in computer science. *Interval notation denotes the least and greatest numbers in the range, as well as using either brackets or parentheses to denote whether those numbers are included or not, respectively.*

Say we wanted to restrict the choices to the numbers 0, 1, 2, …, 7, 8, 9, but not 10. In interval notation, this could be written as [0, 10). But if we just wrote [0, 10) it doesn't make it clear whether that includes fractions, so to be perfectly specific we'd say our range is "integers on [0, 10)." Notice this is the same as saying "integers on [0, 9]", but only because we're talking about integers.

For this assignment, the players are guessing integers on [1, 100).

Here's how you can generate random numbers in Java:

1. Import the class `java.util.Random`
2. Make the instance of the class `Random`, i.e., `Random rand = new Random()`
3. Invoke one of the following methods of your `rand` object:
   `nextInt(upperBound)` generates random integers in the range [0, `upperBound`)
   `nextFloat()` generates a float in the range [0, 1.0)
   `nextDouble()` generates a double in the range [0, 1.0)

Here is a code snippet you can try out. This program simply generates a random number between 0 and 9. Run it a few times and you will see the random number is different every time (Hopefully! With random numbers you never know ☺).

```
import java.util.Random; //Import the class java.util.Random

public class MyProg {
   public static void main(String [] args ) {
      Random rand = new Random(); //instance of random class
      int randomNumber = rand.nextInt(10); //generate random number
      System.out.println("Random random generated: " + randomNumber);
   }
}
```

You can vary the range by using arithmetic operations, such as adding a constant number to the result of the method call or multiplying the result by a constant number. Doing the math on the lower and upper bound of the interval is a good way of determining what the result would be.

For example, if you wanted a range of integers on [-5, 5], you would do this:

1. Call `rand.nextInt(11)`, giving integers on [0, 11), equivalent to [0, 10].
2. Subtract 5 from the result of Step 1 to get [-5, 5]. You can tell that that would be the result because the lowest number is 0, and 0 - 5 is -5, while the highest number is 10 and 10 - 5 is 5.

If you wanted a range of floats on [-2.3, 0.3), you would do this:

1. Call `rand.nextFloat()` to get floats on [0, 1.0).
2. Multiply the result of Step 1 by 2.6, to get floats on [0, 2.6).
3. Subtract 2.3 from the result of Step 2 to get [-2.3, 0.3).

You can read more about the Random class in Java 18 here. This is the API specification, commonly known as the documentation for the language, for our preferred version of Java.

Deliverables & Reminders

Submit a .zip file on Canvas. This .zip file can either be your zipped project folder from your IDE, or a zipped folder containing your GuessingGame.java file.

Your IntelliJ project can have any appropriate – `Assignment1` would be a good choice.

In addition to solving the problem, your code should adhere to these requirements:

- All variables should have informative names.
- Your code should be clean, presentable, and consistent with Java conventions which includes:
    - Classes are named in PascalCase, while variables are in camelCase
    - Be consistent with your spacing and opt for readability. Leave spaces between, for example, numbers and operators, but not extraneous spaces in places like method arguments.
        - ex: `System.out.println("22 / 7 is " + (22 / 7))` or `System.out.println("22 / 7 is " + (22/7))`, not `System.out.println("22 / 7 is "+( 22/7) )`.
    - Use correct indentation for both code and curly braces.
- Every program must have comments. There should be header comments in your program including your name, a brief introduction of the program, and step-by-step description of what the program does and how it does it. You should have at least one comment in-line with the code besides that.

Your input/output should match the examples. You'll want to check all the edge-cases when you test your solution, because the graders will be doing that. The edge-cases we are NOT considering (i.e., you can assume these don't happen) are listed below.

- If a user inputs a non-integer, or a number outside the specified range.

Ensure your submission contains the necessary file(s) and is the correct version of your code that you want graded. This is your warning. You will not be allowed to resubmit this assignment after the due date if you submit the wrong file.

Since this is your first assignment and not a lab, remember that it is graded on accuracy and you are not allowed to collaborate on your solutions with others. Plagiarism or cheating will be dealt with seriously.

Rubric

A detailed rubric is also available on the assignment page on Canvas, but here is an outline of the point breakdown for different categories.

- (20pts) Code cleanliness, conventions, and comments
  - (20/20pts) for >95% correctness and all required comments
  - (15/20pts) for repeated minor mistakes or one egregious mistake, or for missing either overall or inline comments
  - (10/20pts) for code that is sloppy and hard to understand or is missing all comments
  - (<=5/20pts) for code that has no comments, is hard to understand, and consistently does not follow Java conventions
- (40pts) Program works in the general case
  - (10pts) for collecting user input
  - (10pts) for deciding and displaying the winner correctly
  - (10pts) for valid output of WHY the winner won in the general case
  - (10pts) for proper formatting of the output, e.g., no lines of output are missing line breaks, user input is able to be typed in on the same line as the prompt, etc.
- (30pts) Program works in all valid edge cases
  - (-5pts) for each edge-case that was acknowledged and had a fix attempted, but not completely addressed
  - (-10pts) for each edge-case overlooked in solution
- (10pts) Valid generation of random numbers

Changelog

If any corrections, clarifications, etc. are made to the assignment after its release, they will be listed here. Any changes more substantial than simply fixing typos will get corresponding Canvas announcements. That way, you don't need to keep checking whether updates have occurred.