Project 5: Guessing Game

Programming Assignment #5: Guessing Game

(This assignment is borrowed from the UW 142 class taught by Stuart Reges, ideas and text below are modified slightly from their original on the UW 142 pages. An alternative and more concise version of the assignment is available here: http://www.cs.washington.edu/education/courses/cse142/12au/homework/5/spec.pdf)

This assignment will give you practice with while loops and pseudorandom numbers. You are going to write a program that allows the user to play a simple guessing game in which your program thinks up an integer and allows the user to make guesses until the user gets it right. For each incorrect guess you will tell the user whether the right answer is higher or lower. Your program is required to *exactly* reproduce the format and behavior of the log of execution at the end of this write-up.

You are to come up with your own introduction to the game in the form of a haiku. Recall that a haiku has three lines of text: a line with five syllables followed by a line with seven syllables followed by a line with 5 syllables. **Your introduction must be in the form of a haiku.**

At a minimum, your program should have the following static methods in addition to method main:

- a method that introduces the game with your haiku
- a method to play one game with the user (just one game, not multiple games)
- a method to report overall results to the user

```
<< your haiku intro message here >>

I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's higher.
Your guess? 35
It's lower.
Your guess? 30
It's higher.
Your guess? 32
It's lower.
Your guess? 31
You got it right in 6 guesses!
Do you want to play again? y

I'm thinking of a number between 1 and 100...
Your guess? 50
It's higher.
Your guess? 75
It's lower.
Your guess? 65
It's lower.
Your guess? 64
You got it right in 4 guesses!
Do you want to play again? YES

I'm thinking of a number between 1 and 100...
Your guess? 60
It's lower.
Your guess? 20
It's higher.
Your guess? 30
It's higher.
Your guess? 40
It's higher.
Your guess? 50
It's lower.
Your guess? 47
It's higher.
Your guess? 49
You got it right in 7 guesses!
Do you want to play again? no

Overall results:
Total games    = 3
Total guesses  = 17
Guesses/game   = 5.7
Best game      = 4
```

You may define more methods than this if you find it helpful, although you will find that the limitation that methods can return only one value will tend to limit how much you can decompose this problem.

You are to define a class constant for the maximum number used in the guessing game. The sample log shows the user making guesses from 1 to 100, but the choice of 100 is arbitrary. By introducing a constant for 100, you should be able to change just the value of the constant to make the program play the game with a range of 1 to 50 or a range of 1 to 250 or some other range starting with 1. You must use the Math.random() method to get the random number (please do NOT use the Random object).

When you ask the user whether or not to play again, you should use the “next()” method of the Scanner class to read a one-word answer from the user. You should continue playing if this answer begins with the letter “y” or the letter “Y”. Notice that the user is allowed to type words like “yes”. You are to look just at the first letter of the user’s response and see whether it begins with a “y” or “n” (either capitalized or not) to determine whether to play again.

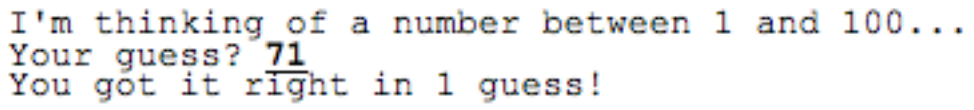
Assume that the user always types an integer when guessing, that the integer is always in an appropriate range and that the user gives you a one-word answer beginning with "y", "Y", "n" or "N" when asked whether to play again. You may assume that no game involves more than 9,999 guesses.

You will notice at the end of the log that you are to report various statistics about the series of games played by the user. You are to report the total number of games played, the total number of guesses made (all games included), the average number of guesses per game, and the best (fewest) number of guesses used in any single game. The average number of guesses per games should be rounded to one decimal place (you can use either the round1 method or a printf – we will go over an example of printf in class to help out).

Here are a few helpful hints to keep in mind.

- To deal with the yes/no response from the user, you will want to use some of the String class methods described in section 3.3 and 4.1 of the book or the lecture slides for 11/18. You should use the next() method of the Scanner class to read the word from the console.
- Because this program uses pseudorandom numbers, you won't be able to recreate this exact log. The key requirement is that you reproduce the *format* of this log and that your calculations for overall statistics are correct for your log.
- It's a good idea to change the value of your class constant and run the program to make sure that everything works right with the new value of the constant. For example, turn it into a guessing game for numbers between 1 and 5. Make sure that it will allow 5 to be a included in the range and not 0, and use the Math.random() method to determine the number.
- While you are developing your program, you might want to have it print out the answer before the user begins guessing. Obviously you don't want this in the final version of the program, but it can be helpful for you while you are developing the code.
- The chapter 5 case study is a particularly relevant example for this assignment. Check it out if you have trouble.

You should handle the case where the user guesses the correct number on the first try. Print the message "You got it right in 1 guess":



```
I'm thinking of a number between 1 and 100...
Your guess? 71
You got it right in 1 guess!
```

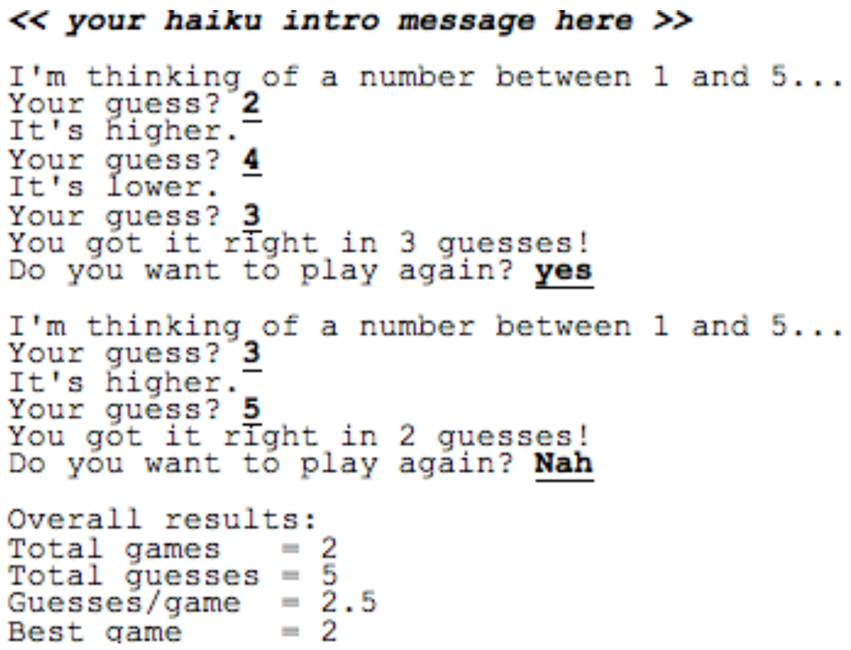
In the last program we asked you to write very short methods that were no longer than 20 lines long and to have a very short main. This program is more difficult to decompose into methods, so you may end up having methods that are longer than 20 lines, but not longer than 35. You can also include more code in your main method than we allowed in the last program. In particular, you are required to have a while loop in main that plays multiple games and prompts the user for whether or not to play another game. You shouldn't have all of the code in main because you are required to have the methods described above.

We will once again expect you to use good programming style and to include useful comments throughout your program. We will expect you to make appropriate choices about when to store values as int versus double, which if/else constructs to use, what parameters to pass, and so on.

For this assignment you are limited to the language features we have covered in class (Chapters 1-5 shown in the textbook) and you are NOT allowed to use the break statement or what are described as "forever loops."

Use whitespace and indentation properly. Limit lines to 100 characters. Give meaningful names to methods and variables, and follow Java's naming standards. Localize variables whenever possible. Include a comment at the beginning of your program with basic description information and a comment at the start of each method. Some students try to achieve repetition without properly using while loops, by writing a method that calls itself; this is not appropriate on this assignment and will result in a deduction in points

Your program should be stored in a file called **GuessingGame.java.**
Here is another sample run:



```
<< your haiku intro message here >>

I'm thinking of a number between 1 and 5...
Your guess? 2
It's higher.
Your guess? 4
It's lower.
Your guess? 3
You got it right in 3 guesses!
Do you want to play again? yes

I'm thinking of a number between 1 and 5...
Your guess? 3
It's higher.
Your guess? 5
You got it right in 2 guesses!
Do you want to play again? Nah

Overall results:
Total games   = 2
Total guesses = 5
Guesses/game  = 2.5
Best game     = 2
```