**CMSC203**

**Assignment #2**

**Random Number Guesser**

**Assignment Description**

Build an application that will receive a guess and report if your guess is the random number that was generated. Your application will narrow down the choices according to your previous guesses and continue to prompt you to enter a guess until you guess correctly.

Notice that if you divide the choices in half each iteration, you will need at most log2(100) ~ 7 guesses.

You will use the utility class RNG.java.

**Concepts tested by this assignment**

* A driver and a utility class
* Pseudo-code
* Java fundamentals, including decision structures, loops
* Selection control statements
* Repetition control statements
* Input validation loops (in RNG.java)
* Relational and logical operators
* Random number generation (in RNG.java)

**Classes**

**Data Element Class – RNG**

* Provided
* This file will generate a random number between 1 and 100
* Note that the method “rand” is a static method, so the java file does not need to be instantiated to use it. Call rand with the following: RNG.rand(100) to generate a random number between 0 and 99.
* Study this class. You will want to use four methods from this class: rand, resetCount, getCount and inputValidation.

**Driver Class – RandomNumberGuesser**

* Student created.
* This is the driver class for RNG that contains a main method.
* The driver is responsible to:
  + print a header.
  + ask the user for an initial guess of the Random Number between 0 and 100.
  + Print out the result for that guess using the methods from the RNG class.
  + Allow user to give another guess between the previous low and high guesses.
  + Display the number of guesses.
  + When user guesses correctly, ask if the user wants to try another round.
  + Print the Programmer's name at the end.
  + Refer to the program sample run for more clarification.
* Data Validation. The following data is validated by the RNG method inputValidation:
  + Guesses must be an integer between the previous low guess and high guess.
* Naming.
  + There should be an attribute named randNum.
  + There should be attributes named nextGuess, highGuess, and lowGuess.
* Add any necessary methods to modularize your code.

**Assignment Details**

Your program should respond like the following sample runs:

== Example Run 1 (one iteration) ===========

A screenshot of a cell phone

Description automatically generated

== Example Run 2 (one iteration, one guess over the limit) ===========A screenshot of text

Description automatically generated

== Example Run 3 (two iterations) ===========A screenshot of text

Description automatically generated

Take screenshots of runs of your program based on your Test Plan. Your runs should include feedback on previous low and high guesses, at least one example of a guess not within the previous low and high guesses, and an example of a second try with a new random number. You use Eclipse. Store your files in the repo you created in GitHub in Lab 1 and take a screenshot.

**Good Faith Attempt**

To satisfy the “Good Faith Attempt” (see Blackboard) your code must compile, run, and print the output.

Note that the GFA is not graded, so you must submit your totally working code by the submission deadline to have a non-zero grade.

**Test Plan**

Test your program with at least 3 test cases. **Make sure your tests cover all the possible scenarios.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Input** |  | **Expected Output** | **Actual Output** | **Did the test pass?** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
|  |  |  |  |  |  |

**Pseudocode**

Write the pseudo code for Assignment 2 based on the Assignment 2 Description given to you. Refer to the Pseudocode Guidelineon how to write Pseudocode.

**Deliverables**

**Deliverables / Submissions and Deliverable format:**

The Java application must compile and run correctly, otherwise Project grade will be 0.

The deliverables will be packaged as follows. Two compressed files in the following formats:

* FirstInitialLastName\_Assignment2\_Complete.zip, a compressed file in the zip format, with the following:
  + Source Code: RandomNumberGuesser.java and RNG.java
  + Word document that includes (use provided template):
    - * Design: pseudo-code
      * Test Plan (test cases, for each test case provide screenshot of the running application)
      * Screenshots:
        + One screenshot of the application running from the command prompt.
        + One screenshot of the application running in your IDE.
        + Screen shots of Java files (RandomNumberGuesser.java and RNG.java) in your GitHub repository
      * Lessons Learned: Provide answers to the questions listed below:
        + Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.
        + What have you learned?
        + What did you struggle with?
        + What would you do differently on your next project?
        + What parts of this assignment were you successful with, and what parts (if any) were you not successful with?
        + Provide any additional resources/links/videos you used to while working on this assignment/project.
* FirstInitialLastName\_Assignment1\_Moss.zip, a compressed file containing one or more Java files (This folder should contain Java source files only):
  + - Source Code: RandomNumberGuesser.java and RNG.java
    - Documentation within a source code should include:
      * comments for each method.
      * additional Comments to clarify a code.
      * one block comment at the top of each program containing the course name, the project number, your name, the date, and platform/compiler that you used to develop the project, for example:

/\*

 \* Class: CMSC203

 \* Instructor:

 \* Description: (Give a brief description for each Class)

 \* Due: 2/23/2021

\* Platform/compiler:

 \* I pledge that I have completed the programming assignment independently.

   I have not copied the code from a student or any source.

   I have not given my code to any student.

   Print your Name here: \_\_\_\_\_\_\_\_\_\_

\*/

**Grading Rubric**

|  |  |  |
| --- | --- | --- |
| **CMSC203 Grading Rubric - Template** | **Possible total Grade: 100** | **Points Earned:** |
| **Name** |  |  |
|  |  |  |
| **TESTING** |  |  |
| Project must compile. If it doesn't compile | 0 |  |
| Project must run. If it's run time error | 0 |  |
| Follows assignment document instructions | 25 |  |
| Passes private instructor tests | 75 |  |
| **Possible Sub-total** | **100** |  |
| **REQUIREMENTS (Subtracts from TESTING total)** |  |  |
| **Documentation:** |  |  |
| Documentation within a source code is missing or incorrect |  |  |
| Header comments at the top of the program are missing | -5 |  |
| Comments for each method are missing | -5 |  |
| Additional comments should be provided to clarify a code | -5 |  |
| Description of what class is missing | -5 |  |
| Design: Pseudocode/algorithm missing or incorrect | -5 |  |
| Javadoc is mission (if Applicable) | -5 |  |
| Required output screenshots are missing | -5 |  |
| Lessons Learned are missing or incomplete | -10 |  |
| MOSS zip file is missing | -5 |  |
| GitHub screenshot is missing | -5 |  |
| **Programming Style:** |  |  |
| Incorrect use of indentation, naming convention, etc: | -5 |  |
| see coding/style standards |  |  |
| User interface |  |  |
| Not clear to user how data is to be entered | -4 |  |
| Output is not easy to understand | -4 |  |
| **Design:** |  |  |
| Does not follow the given flow-chart | -6 |  |
| Does not print application header | -4 |  |
| Does not print the Programmer's name at the end | -4 |  |
|  |  |  |
| **Possible decrements:** | -82 |  |
| **Possible total grade:** | **100** | **Points Earned:** |