1 Description

This project requires creating a single-player game, using only the command-line interface. To play the game, the user first chooses an acceptable range of non-negative integers by entering the minimum and maximum elements in the range (e.g. if the user enters 1 and 10, the answer will lie in the set \(\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\)). Next, the program must randomly choose one of the numbers in the given range, without telling the user this number. The user then enters a guess, and the program should reply whether that guess is correct, too high, or too low. If the user cannot guess the correct number after 10 attempts, they lose and the program should terminate. In the event of a loss, the program should print what the correct answer was. When the user guesses the correct number, the program should congratulate them and terminate.

2 Specifications

Use the java \texttt{Math.random} function to generate your secret number randomly. You may then use any of the methods discussed in class to make sure it is an integer that lies between the numbers chosen by the user.

This project should be created using the Eclipse IDE. Create a new project called \texttt{GuessingGame[lastname]} where \texttt{[lastname]} is replaced with your own last name.

Your project should use only one class file, called \texttt{GuessingGame}, which contains all the code used by the program. Be sure to clearly document your code!

Any and all user input should be validated! Check to make sure they enter valid numbers for each of the guesses as well as the initial setup of the game. Use the \texttt{Scanner} method called \texttt{hasNextInt} to check and make sure the user enters a valid integer guess. If they did not, you can read the erroneous output into a string with \texttt{nextLine} and simply ignore it, allowing the user to enter a new guess.
3 Submission

Upload your project files in a single zip archive (for this project only!) to the course canvas system.

Turn in your source code, properly commented and formatted with your name, course number, and complete description of the code.

Also turn in text files reflecting the output of several different runs of your program (you can copy/paste from the Eclipse output window). Be sure to test different situations, show how the program handles erroneous input and different edge cases.

4 Bonus

You may or may not add the following features for a maximum of 2 extra points each:

- At the end of a game, prompt the user to play again, without having to restart the program.

- Count the number of guesses that the user took to get the right answer and display it at the end if they win (e.g. “You won in only n guesses!”)

- Allow the user to choose from 3 difficulty levels (Easy, Medium, or Hard) where each different level affects the number of guesses they can make (say, 20, 10, 5, respectively).