John Mobarry

CMSC 203

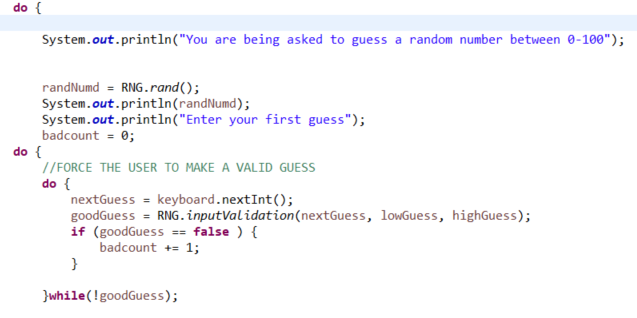
**Assignment 2 Write Up**

The completion of Assignment 2 helped reinforce the lessons learned regarding methods and classes while emphasizing the importance of holistic preparation prior to implementing code. In this assignment I was able to create a program that was able to complete the required functions as long as the numbers were within the required input parameters of 0-100. I toiled for hours trying to modify my code to come to terms with the question and was unable to make it work. As I have discussed in the past with the programmers I know in my life, sometimes when you cannot make your approach work, you have to take a step back and create a new design document from scratch without the influence of your current code/previous document.

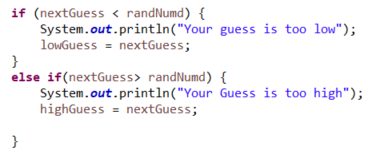
My first issue when attempting to complete assignment 2 is that I was encapsulating the entire problem within the while loop attached below:



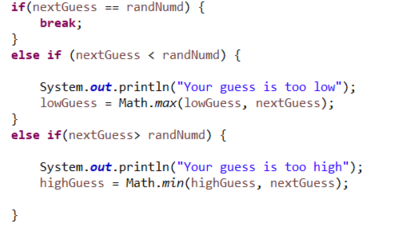
This resulted in a major issue when attempting to complete the code as it did not require validation that the input number is a valid input prior to beginning the loop. This would result in things such as the counter increasing on bad guesses and the highGuess and lowGuess being modified in cases of bad inputs. Instead of using a while loop as the outer shell of the code processing the input guess I implemented nested do loops. As you can see below:



This encapsulation requires the user to put in an input and validate if the guess is correct prior to proceeding through the subsequent code. The use of the boolean variable goodGuess allowed me to keep track of the number of bad guesses to keep the counter of guesses correct. Another issue that I was grappling with during the process of running guesses was verifying that the high and low guesses were not affected by incorrect inputs. To head with this I had to implement the use of Math methods. My original code was as follows:



This would set the low guess and high guess variables to the next guess in cases where they were less or greater than the random number respectively. This worked in the case of correct guesses but would result in strange outputs in the case of incorrect guesses. To remedy this issue I implemented the use of Math.min() and Math.max():

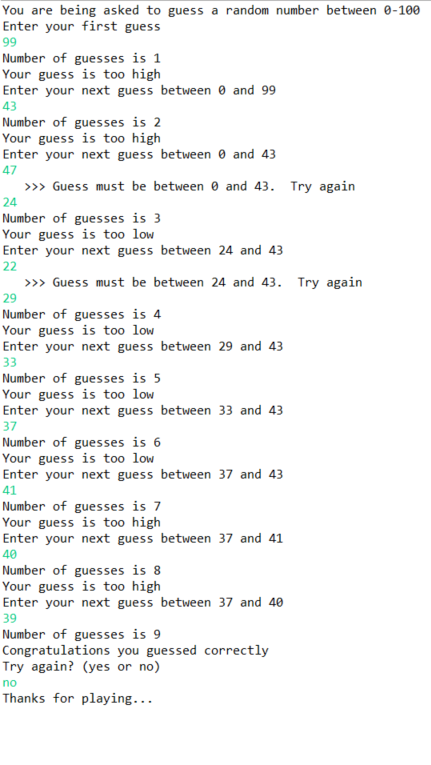


This resulted in the low guess being assigned to the value of the max of the lowGuess and the next guess and the high guess being assigned to the value of the minimum of the lowGuess and the next guess. In cases where the input was lower than the suggested low range it would not change the lowGuess into the erroneous input and in the case where the input was higher than the suggested max range it would not change the highGuess into the erroneous input.

This process of tweaking code from being almost right to meeting the requested parameters taught me about how a programmer needs to often step away and reconsider their approach instead of trying to frankenstein what they already have into functioning code. You cannot always solve presented problem from one perspective and it often takes reconsidering your methods, doing research, and looking back at all of the skills you have learned while programming in the language.

**TEST RUNS**

1.)

****

2.)

