TCSS142 Introduction to Object-Oriented Programming
Assignment 5

Let's Play
GUESS THE NUMBER!

DUE: Monday, August 4, 2014 by 12 midnight

You are to design and implement an interactive game program that will ask the user to guess a predetermined number (program will use a Random object within a given range). The user will be allowed to continue guessing until the correct number is guessed. Once successful, the user will be adorned with pleasantries for their accomplishment.

During the guessing process, the program will give hints if the current guess is either too high or too low. Thus, allowing the user to narrow in on the correct number. It will also keep track of how many guesses are being made (a running total).

The number to be guessed will have to be generated using a Random object and the method nextInt(). The range will also be randomly produced as 1 to 10, 1 to 100, or 1 to 1,000. You will first generate a random range (one of the 3 above) and then use this current range to generate a random number to guess.

Until the user guesses correctly, he/she will remain within a loop (looks like a good use of a while loop here).
Basic loop operations:
After it tests to see if the current guess is correct, it will do the following:

- Offer a hint of too high or too low (I strongly suggest a method call for this)
- Prompt for another guess
- Read another guess
- Increment the running total

The implication above (loop) is that you will need to have an initial guess (priming) to get things going (a Fencepost problem). This would incur one guess on your running total and will require the same, or similar prompt and guess input statements as seen inside the guessing loop.

When you exit the loop with a correct guess, a congratulatory message should be displayed.

Once you get your game method to play a single game correctly, surround the appropriate code with a do / while construct that allows for unlimited games. This will be controlled by a prompt and user response. For example, you will need something implemented as follows, where the method playMore(input) is passed a Scanner for input from the keyboard and returns a boolean value which is true if the user wants to play another game or false otherwise (details of playMore and the method it will call are listed on page 2):

```
do {
    .
    .
    Whatever process you are repeating (getRange, getNumber, start guessing)

    } while (playMore(input));      // Details on playMore are listed on page 2
```

Finally, implement a method, called by main before any other operations are performed, that displays what this program does.

playMore, The BIG Picture:

The essence of playMore is to return a boolean type which will be **true** if the user types a Y or y in response to a prompt to play more games and **false** if the user types any other character:

```
return response == 'Y' || response == 'y';   // Where the return value is boolean and response is of type char.
```

From where does this typed character come? The character stored in response is assigned a return value from a method which prompts the user for another play and reads the user's response:

```
char response = getResponse(input);
```

If you place the above 2 statements in correct order, you have completed the body of **playMore**. However, we now see yet another method; getResponse.

getResponse will prompt the user for another play, read the user response, and return this response to the calling method:

Prompting the user should be straight forward and is up to you to write the statement.
However, reading the response and storing it as a char type (for comparison) is a bit sticky. Remember, getResponse is to return a char type, not a String. When you read non-numeric data you use either the Scanner methods: next() or nextLine() which both return Strings.

In order to access a char type from a given string, you have learned to use the String method charAt(i), where i is an index to the desired character.

Therefore, the steps involved in the getResponse operactions are:

- Prompt user to play again
- read the response into a string
- return (to the calling method) the 1st character of the entered string

NOTE: Because getResponse reads from the console (keyboard), it requires a Scanner (for console input) to be passed as a single parameter (you should have noted this in the examples above). Because playMore calls getResponse, playMore will also need to have a scanner sent as it's only parameter.

Mostly, these 2 methods are required for academic purpose to test your understanding of boolean and char return values from method calls.

You might note, if you are careful with your guesses and take full advantage of the hints offered, you should never take more than $\lceil \log_2 N \rceil$ guesses (or ceiling(lg N) - where the range is N). E.g. guessing in a range of 1 to 1,000 should find success by the 10th guess, range of 1 – 100, 7 guesses. If you look at my guesses, you will see I didn't do better than this possible worst case.

A sample run based on the criteria above is listed on page 3.

Be sure to include proper documentation in the proper places as described on all previous assignments. Not following proper documentation will incur a heavier penalty with this assignment.

SAMPLE RUN NEXT PAGE ---------- >

SAMPLE RUN

This program prompts the user to play the game; Guess A Number from 1 to ...

The upper limit being a power of 10 up to and including an exponent of 3 (1 - 10, 1 - 100, or 1 - 1,000)

Once a correct guess is made, the number of guesses is displayed and the user is prompted to play again.

Games continue until the user responds to the above prompt by typing anything other than a Y or y.

```
Pick a number from 1 to 1000: 500
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 750
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 875
No Dude!  Your guess is to like elevated man.
Guess again: 812
No Dude!  Your guess is to like elevated man.
Guess again: 781
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 796
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 804
No Dude!  Your guess is to like elevated man.
Guess again: 800
No Dude!  Your guess is to like elevated man.
Guess again: 798
No Dude!  Your guess is to like elevated man.
Guess again: 797
AWSOME DUDE!  You guessed it in 10 tries.
Hey Dude, want to Party On [Y/N]?  y

Pick a number from 1 to 100: 50
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 75
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 88
No Dude!  Your guess is to like elevated man.
Guess again: 82
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 85
Bummer man. Like what a downer.  That guess is like under the waves.
Guess again: 87
No Dude!  Your guess is to like elevated man.
Guess again: 86
AWSOME DUDE!  You guessed it in 7 tries.
Hey Dude, want to Party On [Y/N]?  n
```