I Love Being Right

Andrew Rosen

First Part due before Lab 2, Second Part due before Lab 3

1 Let's play a game

For this lab, we will be making a small text based game to teach you and your friends a little bit about human psychology and get you back into the habit of programming. If you're in lab, we'll start with a small demo of the game. You can find the game here for reference. When the game is over and we all know how to play it, flip to the next page.

2 Let's make a game

Your task is to recreate this game in Java, albeit in a simpler, text-based format. The player enters sequences of three numbers to see if they follow the hidden rule. When the player has entered their sequence, the program should reply whether or not the sequence followed the rule.

You can use the same rule or create your own. Regardless, the game is played the same way.

2.1 How

Create a loop which gives the player instructions on what to input. Then read input from the player. The input will be either one of two options:

- If the user enters the word "answer" or some other string you choose to indicate the player is ready to end the game and guess.
  - In this case, prompt the user to guess our game's rule, then output the answer.
  - **Advanced** We can use the `.contains()` method to analyze the guess and our output accordingly:
    - If the guess contains the string "doubled" or "doubling" or "multiple," the guess if most likely wrong.
    - If the guess contains the string "increase" or "increasing," the guess is most likely right and you can tailor the output accordingly.
- Three numbers separated by spaces.
  - If the user enters a sequence that follows the rules, output "Yes!"
  - Otherwise output "No."

2.2 Error Handling

Your program should not crash, regardless of what I input. Possible inputs I might enter

- Too many numbers for a guess
- Too few numbers for a guess
- An invalid option

2.3 Useful Methods

There are a many ways to parse your input, but only a few that are easy. Here are a few different suggestions.

- **Only** use `Scanner.nextLine()` to read in input. Using `Scanner.nextInt()` and `Scanner.nextLine()` together will lead to confusion.
  - `String.split()` is useful for breaking apart strings seperated by *delimiters*, such as commas, or, in this program, spaces.
- How do I tell what the string the user input is? Check either if it's the word "answer" or a sequence of numbers.
- Regular expressions are also useful here, if you know how to use them.[1]
- Break up individual tasks into methods. For example you can have one method to grab user input, another method to detect whether a guess follows a rule, and so forth.

[1] Regular Expressions or regex are not often formally taught as part of a CS curriculum. Messing around with them for two hours might be the most two useful hours of the semester.

3 Lab 2: Making the Game More Interesting

The following objectives need to be completed for Lab 2:

- Exception Handling - Handle cases where the user inputs invalid inputs for the numbers.
- Storing your previous answers - Modify your program to contain an ArrayList and store your guesses as they are made.
  - What should the ArrayList hold?
    - One option is to hold an array of `int`, with each guess being stored as an array.
    - Another way is to create a `Guess` object, which stores the three guesses and a boolean to indicate whether the guesses follow the rule.
- Recalling previous answers - Add a new command "previous" that prints out all previous guesses.
- Extra Credit (5 points) When "previous" prints out the previous guesses, the program should also tell you whether or not the guess followed the rule.

4 Your Grade

Turn in your assignment on Blackboard and come see the Professor or a TA to demo your work for credit.