CSE 143X: Accelerated Computer Programming I/II

HW3: Bagels (due Monday, October 19, 2015 11:30pm)

This assignment focuses on conditionals, while loops, strings, arrays, and pseudorandom numbers. Turn in the following file using the link on the course website:

- `Bagels.java` – A program that plays a game of Bagels with the user.

This assignment is out of **20 points** (which will be scaled to be "worth a single assignment").

The Game of Bagels

Bagels is a variant of the board game Mastermind. In our version, the computer generates a random number that the user has to guess. The solution will always be a number made up of the digits 1 through 9 (no zeroes).

In each individual game, your program should repeatedly prompt the user for guesses and provide a clue after each incorrect guess. After each guess, the computer should provide hints based on the value and position of digits guessed:

- If no digits match between the guess and the answer, your program should print "bagels."
- For each correct digit in a correct position, your program should print "fermi"
- For each correct digit in an incorrect position, your program should print "pica".

Your program should report all of the fermi clues before any of the pica clues to avoid leaking information to the player. After a game, your program should ask the user if they want to play again. If they don't, it should report some overall statistics on how they did.

Program Details

User Input

When you ask the user if they want to play again, you should use the "`next()`" method of the `Scanner` class to read a one-word answer from the user. You should continue playing if this answer begins with "y" or "Y". To do this, look at just the first letter of the user's response and see if it begins with a "y" or and "n" (either capitalized or not) to determine whether to play again. You will want to use some of the `String` class methods described in Section 3.3 and 4.1 of the textbook. You will also need to read simple `int`s from the user. To do this, you will want to use the `nextInt()` method of the `Scanner` class.

User Assumptions

You may assume that all user input is valid. That is, when asking the user the number of digits to play, asking their guesses, and asking them if they want to play again, you may assume that their inputs make sense. That is, you should expect an integer between 1 and 9, a valid guess (all digits, correct length, no zeroes), and a single word that starts with "y", "Y", "n", or "N", respectively.
You may also assume that no game involves more than 9,999 guesses.

Reporting Statistics

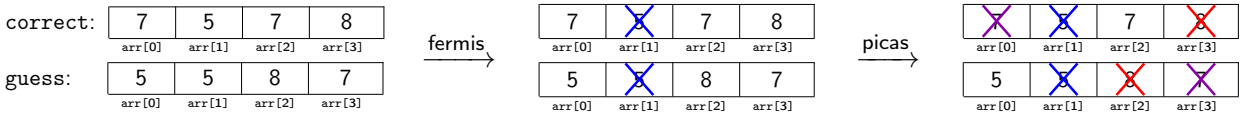
When the user no longer wants to play, you should report the total number of games played, the total number of guesses made (all games included), the average number of guesses per game, and the best (fewest) number of guesses used in any single game. The average number of guesses per games should be rounded to one decimal place (you can use either the `round1` method or a `printf`).

Storing and Manipulating Guesses

You *must* use arrays to store and manipulate the guesses inside your program. This could theoretically be accomplished with `String`s, but, stylistically, arrays are a better choice, and you must use them.

Generating Hints

To determine fermi and pica hints, we recommend doing two passes over the digits: the first one to identify fermi matches (matching digits at matching locations) and the second one to identify pica matches (matching digits at different locations). You must make sure that the same digit is not used for two different matches. To do this, you should mark digits as "used" so that they will not be used again in a second match. For example, in the figure below, the numbers with an 'X' through them should no longer be considered:



In this case, there was one fermi and two picas; so, the correct clue would be "fermi pica pica."

Testing & Random Seed

To make it easier to check your output, we require that you put together your answer in a particular way. You should construct a new `Random` object each time you generate a number to be guessed, you should generate one digit at a time, and you should put the answer together in forwards order (1st digit generated is 1st digit of answer, 2nd digit generated is 2nd digit of answer, etc.). You can test your solution by constructing your `Random` object with the seed 42:

```
Random r = new Random(42);
```

Example Output

```
>> Welcome to CSE 143x Bagels!
>> I'll think up a number for you to guess.
>> Each digit will be between 1 and 9.
>> Try to guess my number, and I'll say "fermi"
>> for each digit you get right and in the right
>> place, and "pica" for each digit you get right
>> that is in the wrong place.
>>
>> How many digits this time? 3
>> Your guess? 123
>> pica
>> Your guess? 456
>> pica pica
>> Your guess? 265
>> pica pica
>> Your guess? 542
>> You got it right in 4 guesses
>> Do you want to play again? y
>>
>> How many digits this time? 3
>> Your guess? 123
>> fermi fermi
>> Your guess? 456
>> bagels
>> Your guess? 789
>> bagels
>> Your guess? 122
>> fermi
>> Your guess? 113
>> fermi fermi
>> Your guess? 133
>> You got it right in 6 guesses
>> Do you want to play again? n
>>
>> Overall results:
>>     total games   = 2
>>     total guesses = 10
>>     guesses/game  = 5.0
>>     best game     = 4
```

You should check your output using the Output Comparison Tool.

Hints for Development

- While you are developing your program, you might want to have it print out the answer before the user begins guessing. Obviously, you don't want this in the final version of the program, but it can be helpful while you are developing the code.
- You should handle the case where the user guesses the correct number on the first try. Print the following message: "You got it right in 1 guess"

Style

For this assignment you are limited to the language features in Chapters 1-5 shown in lecture or the textbook, and you are not allowed to use the break or continue statement for a loop.

We will once again expect you to use good programming style and to include useful comments throughout your program. We will expect you to make appropriate choices about when to store values as int versus double, which if/else constructs to use, what parameters to pass, and so on. You are not allowed to use a break statement and you are not allowed to use a return from a void method for this or any future 143X assignment.

At a minimum, your program should have the following static methods in addition to method `main`:

- a method that introduces the game
- a method to play one game with the user (just one game, not multiple games)
- a method to report overall results to the user

You should introduce other methods as well so that no single method has more than 25 lines of code. You are encouraged to include the `round1` method discussed in lecture.

Use whitespace and indentation properly. Limit lines to 100 characters. Give meaningful names to methods and variables, and follow Java's naming standards. Localize variables whenever possible. Include a comment at the beginning of your program with basic description information and a comment at the start of each method. Some students try to achieve repetition without properly using while loops, by writing a method that calls itself; this is not appropriate on this assignment and will result in a deduction in points.