Assignment 7 - Bonus Project Topic Mastermind

Assigned: Apr 21, 2016
Value: 20 points
Program Due: 11:59 pm, Thursday, Apr. 28th

Pair Programming Assignment. You must work with one other person on this assignment using the pair programming technique. Choose your partner by yourself, and sign up the partner spreadsheet here http://bit.ly/1XKoUzP for the Bonus Project Partner Tab.

General Assignment Requirements

The purpose of this assignment is to design and implement an OO program with multiple classes to play a text based version of the board game called Mastermind. You are free to use whatever classes and methods from the JAVA library you wish to use. External jars are not allowed.

First thing to do is - Read the Wikipedia article on the game of Mastermind at:
http://en.wikipedia.org/wiki/Mastermind_(board_game)
Here is a good example of what an interactive GUI for Mastermind would look like:
http://www.web-games-online.com/mastermind/index.php

The game you implement will have the following properties.

- The computer will randomly generate the secret code.
- The player will try to guess the secret code.
- By default, the player has 12 guesses to guess the code correctly.
- If the player does not guess the code correctly in 12 guesses they lose the game.
- By default, the secret code consists of 4 colored pegs in specific position order.
- The valid colors for the pegs are blue, green, orange, purple, red, and yellow. Capital letters will be used in the examples to indicate colors. B for blue, R for red, and so forth.
- The results of a guess are displayed with black and white pegs. The Wikipedia article refers to the results as feedback.
- A black peg indicates one of the pegs in the player's guess is the correct color and in the correct position.
- A white peg indicates one of the pegs in the player's guess is the correct color, but is out of position.
- A peg in the guess will generate feedback of either: 1 black peg, 1 white peg, or no pegs. A single peg in the guess cannot generate more than 1 feedback peg.
- The order of the feedback does not give any information about which pegs in the guess generated which feedback pegs.
- The player's guesses must be error checked to ensure they are the correct length and only contain valid characters.

The output of the game can be a simple text based display on the console, similar to the following example:

Sample user dialogue

Initial greeting (it is required that you output the greeting below).

```
Welcome to Mastermind.  Here are the rules.

This is a text version of the classic board game Mastermind.
The computer will think of a secret code. The code consists of 4 colored pegs.
The pegs may be one of six colors: blue, green, orange, purple, red, or yellow. A color may appear
more than once in the code. You try to guess what colored pegs are in the code and what order they are
in.   After you make a valid guess the result (feedback) will be displayed.
The result consists of a black peg for each peg you have guessed exactly correct (color and position) in
your guess.  For each peg in the guess that is the correct color, but is out of position, you get a white
peg.  For each peg, which is fully incorrect, you get no feedback.

Only the first letter of the color is displayed. B for Blue, R for Red, and so forth.
When entering guesses you only need to enter the first character of each color as a capital letter.

You have 12 guesses to figure out the secret code or you lose the game.  Are you ready to play? (Y/N):
Y

Generating secret code ....    (for this example the secret code is YRBY)

You have 12 guesses left.
What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: OOOO

OOOO ->  Result: No pegs

You have 11 guesses left.
What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: oooo  -> INVALID GUESS

What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: kkkk -> INVALID GUESS

What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: RRRRR -> INVALID GUESS

What is your next guess?
Type in the characters for your guess and press enter.
```

```
Enter guess:     -> INVALID GUESS

What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: RRRR

RRRR -> Result:  1 black peg

You have 10 guesses left.
What is your next guess?

(Etc. etc., . . . )

What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: YRBY
YRBY -> Result:  4 black pegs – You win !!

Are you ready for another game (Y/N): N
```

Testing mode: The top-level class of your program can be called **Game**. It can have a constructor that takes a boolean value as its parameter. The boolean value is used for running the game in the testing mode until you are finished. If it is true, then the secret code is revealed throughout the game. The Game class can also have a method named runGame that carries out the actual games. Your main method creates and uses a new Game object for each game played.

History: In correctly guessing the code, it is helpful for you to be able to review the history of the guesses and replies. This represents the state of the Board. Therefore, you should implement a way to have the program output the history in order from the beginning whenever you want to see it. So, when the user types in "history" for the guess, the program would then display that history.

```
What is your next guess?
Type in the characters for your guess and press enter.
Enter guess: history

. . . (the history of all guesses and replies for this game are displayed in tabular format)
```

Potential Extensions:

- Your program could be altered to allow a different maximum number of guesses, or a different number of pegs in the code. New colors could be added, assuming they start with a different letter than other existing colors. For example, user can change the code to have 5 pegs, allow pegs to be the color Maroon, and guess 15 times.
- You may want to create a fancy UI and more interactive way to play the gram.

Tips:

- Recall when designing a program in an object oriented way you should consider implementing a class for each type of entity you see in the problem. For example in Mastermind there is a game board, pegs, colors, codes (the secret code and the player's guesses can both be considered the same data type, a code), feedback results, a computer player, a human player, and an over-all game runner.
- The hardest thing algorithmically about this assignment is how to generate a result given a secret code and a guess. It is important to realize black pegs should be generated first and that a given peg from the secret code and the guess cannot be reused once matched.

Here are some examples:

Code: RRRR
Guess: BBBB

This one is easy. There aren't any B's in the code, so the guess of BBBB would generate a result with no pegs.

Code: RRRR
Guess: RBBY

The R in the first position of the guess matches up exactly with the R in the first position of the code. This generates one black peg. The key is once a peg has generated a peg it cannot be used to generate any more. So the R in the first position of the code and the first position of the guess cannot generate any more pegs. The guess of RBBY generates a result with 1 black peg and 0 white pegs.

Here is another example.

Code: RBYR
Guess: BBRG

This code and guess would give a result of 1 black peg and 1 white peg. Black pegs take precedence so the B in the second position of the code and guess result in a black peg. One useful way of solving this problem is to determine the black pegs and scratch out or replace the pegs that were used. So given the Code of RBYR and the Guess of BBRG there could be a temporary result of

Code: R-YR
Guess: B-RG

To get the number of white pegs in the result the remaining characters can be compared for matches, independent of position. If a match is found then those pegs should be scratched out as well. The B in the first position of the guess doesn't match any characters in the code. The "-" in the second position of the guess should be skipped. The r in the third position of the guess matches the R in the first position of the code. This is a peg in the guess that is the right color, but in the wrong position so it generates 1 white peg. Once used the R in the first position of the code and the R in the third position of the guess can be reused so they should be scratched out.

Code: --YR
Guess: B--G

The G in the fourth position of the guess doesn't match anything so the result would be 1 black peg and 1 white peg. To summarize

Code: RBYR
Guess: BBRG

generates a result of 1 black peg and 1 white peg.

Deliverables and Grading:

- Submit all your java files. Include README with name and github repository.
- We encourage creative implementation for Mastermind Game. TA will compare your implementation with others and give a “relative” score up to 20 points.
- The score for bonus project will be added to the sum of your other 6 assignments. Your final assignment score is (A1+A2+…+A6+bonus_project) / 6 .
- Potential grading rubrics:
  + 8 points, finish default functionality (4 color, 4 pegs, 12 guesses, console interaction)
  + 2 points, able to change number of color, number of pegs, and number of guesses.
  + x points, GUI, and anything else. It depends.

Adapted from an assignment written by Mike Scott.