This assignment is to be completed as a pair

Assignment Two
Mastermind

Software Design and Programming and
Software and Programming III

Spring Term 2016

The purpose of this assignment is to design and implement a program with multiple classes utilising the functional programming aspects of the programming language. You should use immutable data structures where possible and a dependency injection mechanism to bind the specification with the implementation.

This assignment can be completed in either `Java` or `Scala`; you choose.

Preamble

This is a pair assignment. You may work with one other person on this assignment using the pair programming technique. Review this paper on pair programming. The intent is that you work together, at the same computer, on the assignment. One person "drives" (does the typing and explains what they are doing) and the other person "navigates" (watches and asks questions when something is unclear). You should **not** partition the work, work on your own, and then put things together.

You may not acquire, from any source (e.g., another student or student pair or an internet site), a partial or complete solution to a problem or project that has been assigned. You may not show another student or student pair your solution to an assignment. You may not have another person (current student, former student, tutor, friend, anyone) *walk you through* how to solve an assignment.

Description

You are required to design and implement a program to play a text based version of the board game Mastermind. You are free to use whatever classes and methods from the Java standard library you wish to use. You may also use one of the following dependency injection frameworks:

- SpringDI

- Google Guice
- Dagger 2
- MacWire

The version of 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.
- The player has twelve guesses to obtain the code.
- If the player does not guess the code correctly in twelve or fewer guesses they lose the game.
- The code consists of four coloured pegs.
- The valid colours for the pegs are *blue*, *green*, *orange*, *purple*, *red*, and *yellow*.
- 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 colour and in the correct position.
- A white peg indicates one of the peg's in the player's guess is correct, but is out of position.
- A peg in the guess will generate either one black peg, one white peg, or no pegs. A single peg in the guess cannot generate more than one feedback peg.
- The order of the feedback does not give any information about which pegs in the guess generated the feedback pegs.
- Capital letters will be used to indicate colours. `B` for blue, `R` for red, and so forth.
- A user's guesses are error checked to ensure they are the correct length and only contain valid characters.
- The output of the game should be a simple text based display.

Sample Output

Your program does not have to match this output exactly. You can make changes to the style of the output if you wish. You will find a sample of the output in the file `SampleOutput.txt` on the module repository.

Provided Files

The following files are provided in the module repository folders `cw-pair-java` or `cw-pair-scala`.

	File/Class	Description
Sample Output	`SampleOutput.txt`	Some sample output.
Implementation	`MasterMind`	Driver — with `main` method
Implementation	`Game`	interface
Implementation	`GameAbstractImpl`	adds constructor to `Game`

The top level "class" of your program must be called `Game`. It must have a constructor that takes a boolean. The boolean is used for testing purpose. If it is true, then the secret code is revealed throughout the game. The `Game` class must also have a method named `runGames` that carries out the actual games. Your program must run correctly when the main method of class `Mastermind` is called.

Evaluation Criteria

Part of the assignment grade will be determined by how easily your program could be altered to allow a different number of pegs in the code and how easily different colours could be added, assuming they start with a different letter than other existing colours. (Up to 26, one for each capital letter. Are there any colours that start with an X?) For example, how easily would it be to change the code to have five pegs and allow pegs to be the colour Maroon?

Part of the assignment grade will be determined by if you broke the problem up into different classes and interfaces. One of the criteria of the assignment is to break the problem up into multiple classes even if you think the problem could be solved more easily with no discernible structure or one big class. For this assignment you should err on the side of having lots of simple classes instead of a few complex ones.

Part of the assignment grade will be determined by appropriately applying the SOLID principles, especially interface separation and dependency injection.

Part of the assignment grade will be determined by how comprehensive a test suite you provide.

Submission

Only one of the pair will submit the version of your code. You should pick which account to submit the code to. The other member of the pair should include a `README.md` in their repository indicating who their partner is. The submission is part of your portfolio repository and it will be cloned at the appropriate due date and time.