CS5031 - Software Engineering Practice

Assignment: **P1 – Master Mind Word**

Deadline: 21 February 2025 (Friday Week 4) *Credits*: 20% of module

MMS is the definitive source for deadline and credit details

You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

Aim / Learning objectives

The aim of this assignment is to practise the use of test-driven development as well as good coding practices, refactoring methods, build tools, and version control.

Requirements

You are required to develop a portable Java version of Master Mind, a word game where players have ten attempts to guess a five-letter word. The rules are given in the Appendix; it is similar to the popular web game of Wordle, though with slightly different rules.

The game must be developed using a test-driven development process and use Junit 5 as the unit-testing framework. The code must build, and tests must run using the Maven build framework. A Maven configuration file is provided.

You must use Git to version-control the code so that you can easily revert to a previous state if anything goes wrong, and the marker can assess how you have used a step-wise, systematic approach following the logic of test-driven development. For this purpose, check in versions of your code that pass the current set of unit tests before you add further tests and improve the quality of the code.

You must document all code with comments or JavaDoc. You may, but are not required to, use Maven plugins to generate JavaDoc where appropriate; if you do, please provide instructions on plugin usage in your report.

Your code must be able to model the game rules and game state. Your code must be able to be compiled into a portable jar file using Maven and run from the command line (as opposed to running from your IDE).

The code must select a word randomly from the provided word-list and accept user input (guesses) from the command line with correct error handling.

After each input, the appropriate feedback should be displayed to the user before accepting another input. Finally, your code should provide a score to the user based on how many guesses were required.

A simple command line interface is sufficient for this practical. You may use coloured outputs, or report results in text format.

As long as the game play is implemented, you are allowed to add additional features to your implementation. If you do so, please discuss this in your report.

Report

Your report must include:

- An overview of your implementation, including How to Play
- A description of your test-driven approach
- A description of your refactoring approach throughout development
- A description of any other software development practices used
- Any other relevant discussions, such as a description of extra functionality if provided
- References

The length of the report should be *no more than 800* words (excluding references). This is an advisory limit.

What you are provided with

We have provided for you a zip file containing starter code in a git repository. It has the standard Java source code structure you will have seen in exercises. Download it at

https://studres.cs.st-andrews.ac.uk/CS5031/Coursework/p1/mastermind.zip

Please unzip this file in a place of your choosing. There is an initialised git repository; please keep committing your changes, and at the end, zip up the directory including the git repository. The directory includes a project you can import into your favourite IDE. It also contains a Maven configuration file (`pom.xml`) to build and test (instructions in the README). You should not need to edit the configuration, though there is no problem if you do. The project contains two word-lists: `src/main/resources/wordlist.txt` and `src/test/resources/wordlist-test.txt`. `wordlist.txt` contains the entire list of valid five-letter words for our game; use it for playing your implementation. `wordlist-test.txt` contains three five-letter words that can be used for your unit tests. You must use these as the word-lists. You must implement the model for the game using the contents of this project as a starting point and commit successive partial implementations to your git repository.

Autochecking

This assignment does not use `stacscheck`. You are expected to write your own tests.

Submission

Create a single zip file of the entire directory containing your code, including the Git repository. Make sure there is:

(1) A usable git repository (check with git log; this should give the incremental versions of the source code).
(2) A README file with instructions on building, unit testing, and running your code.
(3) A PDF copy of your report.

The zip file must be submitted electronically via MMS by the deadline. The latest version of the code should be checked out in the workspace. Please ensure that the repository has everything committed (clean working directory), and the git repository is included in the zip file. Submissions in any other format will be rejected.

Marking

Marking will follow the guidelines given in the school student handbook (http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors).

Some specific descriptors for this assignment are given below:

Mark range	Descriptor
1 - 6	An attempt to develop a game model that fails to implement data structures and game logic for the game. A minimal attempt at a report showing a lack of understanding of the requirements.
7 - 10	Successful implementation of the game logic but with significant problems, either in terms of faults in the implementation or in not following a test-driven process with no or few unit tests produced. A reasonable attempt at a structured report with some evidence covering the required information.
11 - 13	A version of the game that may have shortcomings in the quality of the implementation or may not pass all the unit tests. Evidence of an attempt to follow the test-driven development process. A competent attempt at the report with a clear structure and writing covering most required aspects.
14 - 16	An implementation that passes all the unit tests. Unit tests cover all functionality, including edge cases and executing all branches in the code. The code quality might need to be revised. Evidence that the test-driven development process was used. Good attempt at the report addressing all required aspects in good writing without major problems.
17 - 20	An implementation that is fully working and thoroughly tested, applying the test-driven approach and evidence of code quality refinements through refactoring methods, leading to excellent code quality. A well-written report outlining excellent work addressing all required aspects.

Lateness

The standard penalty for late submission applies (Scheme A: 1 mark per 24 hour period, or part thereof): http://info.cs.st-andrews.ac.uk/student-handbook/learningteaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:
https://www.st-andrews.ac.uk/students/rules/academicpractice/

Appendix: Master Mind Word Game Rules

The objective of the game is to guess the correct five-letter word in ten or fewer tries. The five-letter word is unknown to the player until a correct guess, or there are no more guesses. After each guess, feedback is provided depending on the position and correctness of each letter.

- A correct letter in the correct place counts as a green.
- A correct letter in the wrong place counts as a yellow.
- An incorrect letter counts as a grey.

The total number in each category is reported (note, unlike Wordle, it is not reported which letter in the guess is green, yellow, or grey). Letters can be used more than once, in reporting feedback each instance of the letter counts as a guess. Guessed words must exist in the word-list (Note: the word-list does not contain all five-letter words).

Some examples below:

(Hidden word: DREAM)

Guess: DAISY; Feedback: 1 Green, 1 Yellow, 3 Grey (explanation: D counts as a green, A as a yellow, the other three are grey)

Guess: FREED; Feedback: 2 Green, 1 Yellow, 2 Grey (explanation: D counts as a yellow, the E in the 3rd position counts as a green, and so does the R. Note that the E in the 4th position counts as a grey as there is no other E in the hidden word).

(Hidden word: FREED)

Guess: DREAM; Feedback: 2 Green, 1 Yellow, 2 Grey (explanation: R and the E in the 3rd position counts as a green, the D counts as a yellow. Note no indication is made that the E is repeated).

Guess: DRIED; Feedback: 3 Green, 2 Grey (explanation: The D in the 5th position, the R and the E counts as green. Note that the D in the 1st position counts as a grey).