Assignment 1: Clean Code and Testing

CS5031

Due date: Wednesday 20th February, 21:00
30% of the overall module grade

Aims

The aim of this assignment is that you should:

- learn to improve the code quality and structure of a system through successive application of refactoring steps;
- using unit tests to ensure that the systems interface contracts are not broken;
- using version control to create a version record of the changes made to the system.

The System

The system you are going to work with is a console implementation of the game of Hangman. In this game, the player attempts to guess a word or phrase selected at random by the computer, either by guessing individual letters (in which case the computer shows where those letters appear in a word) or by guessing the entire word. If the player guesses the word correctly, they win. After a limited number of incorrect guesses, the player loses.

You can read a full description of the game at `https://en.wikipedia.org/wiki/Hangman_(game)` or play an online version at `https://www.coolmath-games.com/0-hangman`

Requirements

You can find the code for the system on `studres`, in the directory `CS5031/Practicals/P1/Hangman`. The code you are given works (more or less) but you will find several problems with it. For example:

- The code is not very clean, nor is it very well organised or testable, so you will need to begin by examing the code to understand how it works.
- Because of this, there are (almost) no unit tests, so if it is refactored and improved it will not be clear whether the system works in the same way as before.
- Since the code is not clean, and there are no unit tests, there are several bugs.

Your task is to use refactoring to iteratively improve the code quality and structure of the system, and to use version control to record and justify each of the changes you have made. Specifically, the requirements are to:

- Refactor the code so that it follows the clean coding guidelines given in the lecture in week 1.
  - Or any other guidelines, provided that you can justify them!
- While refactoring, add unit tests to clearly define how the system's interfaces should behave.
- For each refactoring and test (or set of related tests), record the change in a source control system with a commit message justifying the change.

  - You can either use git, as covered in the lecture in week 1, or, if you prefer, Mercurial.

- Fix any bugs you find in the code, again recording the change in source control.
- Add a `build.gradle` for ease of building and deployment

You are free to change any part of the design or make any implementation decisions you deem necessary. For extra credit, you can investigate tools such as Checkstyle and Findbugs, and use them to guide your refactoring.

Deliverables

You should submit to MMS a `.zip` or `.tar.gz` file containing:

- The final version of the code, in a top level directory `Code/`
- The complete git (or Mercurial) repository containing a complete version record of the code with appropriate commit messages
- A short report (up to around 500 words) describing the refactorings you have used, the rationale behind your choice of unit tests, and any design decisions you have made

Marking

The assignment is worth 30% of the overall marks for CS5031. This practical will be marked according to the guidelines at `https://studres.cs.st-andrews.ac.uk/Library/Handbook/academic.html#/Mark_Descriptors`

Some more specific guidelines for this assignment:

- For a grade of 10 or higher, you should provide a minimal set of unit tests, and make clear improvements to the given code with their correctness justified by your unit tests, with each of your changes checked in to version control.
- For a grade of 13 or higher, you should provide an improved set of unit tests, with the rationale for those tests describe in the report, and show clear improvements to the given code, with disciplined use of version control using good commit messages.
- For a distinction grade (16.5 or higher), you should provide a thorough set of unit tests, runnable via gradle, and significantly improve the code so that it clearly follows clean coding standards, with clear justification for the refactorings given in the commit messages in the version control repository, and demonstrating use of external tools such as Checkstyle or Findbugs to direct your refactorings.

Note that you will be graded on your application of software engineering techniques and the quality of the final code. In particular, this means that you will *not* receive extra credit for adding features to the system (other than those explicitly specified above).

Finally:

- Standard lateness penalties apply as outlined in the student handbook at `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html`
- Guidelines for good academic practice are outlined in the student handbook at `https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html`