CSCM94 Coursework 2: Object-Oriented Software Design (Group Work)

February 11, 2019

Number of Credits: 15% of the 15 credit module.
Submission Deadline: Friday 1st March 2019 via Blackboard.
Learning Outcome: To gain experience of designing larger software as part of a team.

1 Problem Statement and Overview

In this coursework, each group will complete an object oriented design of an application that provides a platform for playing the popular game *'Go'*. This electronic system shall adhere to the detailed list of the functionality and requirement specifications provided in this document (Functional Specification). Hold on to this document. You will need it for Coursework 3.

2 Functional Specification

The main purpose of Go94 is to support the playing of the game Go by two players. Such a system would normally (these days) be run as a web application, but to make this assignment manageable, Go94 will run as a stand alone PC application. Only two users can be logged in at a time. All data will be stored in files on the local computer - no networking or web technology will be involved.

It is up to you to design the data structures, algorithms and GUI involved in realising this system.

Figure 1: A typical board of Go.

2.1 Users

There are two categories of users within the system: normal players and administrators.

Each user has the following information associated with their account:

1. A username that uniquely identifies their account
2. A first name and a last name
3. The date and time of the last time a user was logged in
4. A profile image
5. A win percentage

Additionally, administrators also have:

- Join date.
- Admin Number.

You should design and implement classes to manage this information. Also, you should create a graphical user interface that allows Administrators to create new accounts.

The system needs to support two player games on a stand-alone local machine. Players should be able to pick a profile before starting a game.

2.2 The Game 'Go'

Go is a traditional 2-player game dating back 2,500 years and is considered to be the oldest game still actively played.

The game is played on a board made up of a 9x9 or 13x13 grid. Each player places tokens onto the board with a view to controlling the most territory at the end of the game.

At the start of the game, players decide on playing as either Black or White. The player who starts as Black has an advantage in that the black player moves first, this is usually given to the lesser experienced or skilled player.

Players take alternate turns placing stones on the intersections of the lines (points). The objective being to surround territory and keep it free from stones of the opposite player. Once placed, stones are not moved unless they are captured. A capture occurs when all of the stones of a colour in a group are surrounded and no liberties (available free adjacent spaces) remain.

To avoid stalemate or draws in games, any moves that return the game to a previous state, or a repetition of a position are illegal.

When a player feels it is no longer to his advantage to move, he or she may pass. When 3 turns in a row are passed, the game is considered completed and is now over.

Points are then awarded by counting up stones that have previously been captured along with any dead stones that still remain on the board.

A good explanation of the game can be viewed here `https://www.youtube.com/watch?v=5PTXdR8hLlQ`.

Once the game has finished, player data on the system should be updated.

2.3 Favourite Players

Users shall be able to browse the limited profiles of other users (username, profile image and win percentage only) and mark them as favourites. Users shall also be able to unmark a user as a favourite. When browsing the leaderboard, the user shall also have the ability to mark a player as a favourite.

2.4 Information Since Last Login

The user will have the ability to look at a variety of lists which will contain information that has changed since the last time they logged in. Specifically:

- Games that have been completed since they last logged in
- Changes to their position in the leaderboard
- New players that have been added to the system by an administrator (only by username)

2.5 Profile Images

Each user must have a profile image that can be selected from a file. The system will have a set (say 5 or 6) built in Avatar images. The user can select one of these as their profile picture.

2.6 Betting

Within the game, each user is initially awarded 100 credits that can be used to bet on games that a user enters. If both players in a game place a bet, then the winner takes away all of the credits. If only 1 user places a bet, then betting on the game is cancelled and the credits are returned to the original player.

2.7 Extra In-game Credits

Should a player lose enough games to be left with 0 credits, they can apply to an administrator on the system for more credits. This will be achieved by providing a written application to the administrator who can approve or dismiss the awarding of extra credits in game. The administrator should be able to see the application, along with previous applications made.

Should the administrator agree to the application, the player will be given 75 credits for future games.

2.8 Player Records

The system should track the wins and losses of each player and administrator in the system, along with their earnings in game. This information will then be used to calculate win percentages for every users' profile page, as well as a leaderboard. This leaderboard will provide a tabular view of the players on the system and rank them according to either their win percentage, total number of wins or overall winnings.

2.9 User Dashboard

Each user shall have their own dashboard where they can view the following:

- A history of all results the games a player has played in chronological order.
- A list of favourite players
- The current balance of the account (i.e. the credit balance)

2.10 Other Properties

Initially, all the information shall be stored in files on disk. More specifically, the Go94 game must have the following property:

- For all users, their profile information, game history and favourite user list must be saved to disk locally. When you restart the program, these details will be loaded from a series of files.

The first version of Go94 that you create should operate on a single machine. A user logs in to the program and performs actions which are saved locally to disk on that machine. The user logs out. A new user can then log in to view what has changed. When you start the implementation part of the project, please do not proceed to something more complicated until you have all Go94 functionality working well locally on one machine.

2.11 Extra Go94 Properties

This section is not technically part of Coursework 2, but it will be part of Coursework 3. It will be helpful for you to plan for extensibility and hence know about this section.

To achieve top marks in Coursework 3, you will need to be creative. At a minimum, all the functionality of the Functional Specification should be completed to a high standard. All features should adhere strictly to the specification. You need to get all this working very well in order to get a low first class mark (in Coursework 3). In order to get higher marks, you would be required extend the implementation in a novel way. All extensions that do not violate the specification will be considered. Substantial extensions to the software, extra reading and learning, will be required to achieve a high first class mark (in Coursework 3).

For this assignment (Coursework 2), do not include these extra features. Simply design without the extra features, but be aware that they will be required for the implementation (in Coursework 3).

3 Coursework 2 Tasks

Your team is asked to provide a complete design for the entire system. This design must include at least one complete class hierarchy. Each team member is required to contribute to at least one class in the design. The designer of the class does not necessarily need to implement it in Coursework 3.

3.1 Submission Requirements

Only electronic submission via Blackboard will be accepted. Please submit your files in the following formats:

- Design Document (PDF)
- Meeting Minutes (txt or PDF)
- Contributions Report (PDF)

All group members must review the document prior to submission. Each individual group member is responsible for understanding the contents of the entire document. Submission indicates that all group members have read and approved the document unless a conversation that has been documented by your tutor indicates otherwise.

3.2 Design Document

The Design Document (report) proposes an object-oriented design for the entire application. In other words, your team incorporates the full functional specification into the design. The Application Design Report is no more than 30 pages including text and diagrams and consists of the sections as detailed below:

3.2.1 Candidate Classes and Responsibilities

Provide a list of candidate classes and their responsibilities. This list is like the CRC cards in lectures but with a bit more detail. For each candidate class the team has identified, the following information is provided:

1. **Class Name:** (in bold)
2. Author:
3. SuperClass:
4. SubClasses:
5. Responsibilities: a list of services this class provides.
6. Collaborations: a list of classes with which the class communicates.

There should be one of these cards for every class that appears in your class diagrams (see Section 2.2.2).

3.2.2 Class Diagrams

After specifying the information in Section 2.2.1, draw the classes using UML Class Diagram notation. Your class diagrams must use appropriate arrows and UML syntax to represent the relationships such as collaborations (i.e., associations, compositions, and aggregations) and generalisations/specialisations (i.e., inheritance). The drawing style is to be modelled after the UML drawing style used in lectures. You should carefully think about navigability and multiplicities when drawing the collaboration relationships (associations, compositions and aggregations). If you would like, you can use UML tools such as Papyrus1[1] (or others).

[1] `1http://www.eclipse.org/papyrus/`

When providing details about the attributes and operations, you must think carefully about type information and your design. The classes and methods should fit together and function to achieve the intended behaviour.

Each class in your design (See Section 2.2.1) must appear in at least one UML Class Diagram. Each class hierarchy identified during your design process must be depicted in a hierarchy in a UML Class diagram.

3.2.3 Hierarchy Descriptions

Each generalisations/specialisations (i.e., inheritance) relationship (See Section 2.2.2) must be accompanied by a short textual description that describes the "iskind-of" relationships used. Make sure that your team provides a justification that backs up its choices. Make sure that abstract classes are distinguishable from concrete classes in your diagrams. Your team should be able to identify two (or more) class hierarchies.

3.2.4 Collaboration Descriptions

Each composition and aggregation relationship (See Section 2.2.2) must be accompanied by a short textual description that describes the "is-made-up-of" relationship used. Make sure that your team provides a justification that backs up its choices.

4 Coursework 2 Submission: Design Report and Contributions Report

Electronic copies of each file and PDF report are to be submitted to Blackboard. One member of each group, **the Planning and Quality Manager**, should lead the submission of all files and reports on behalf of the group, such that they are all in one place (and not scattered amongst several group members). Points will be deducted for those submissions that do not follow the file naming conventions and required file formats.

4.1 Design Document

85% of Coursework 2 (25% presentation, 60% content) Your Design Document is to be included in your submission to Blackboard. The file must be named "GroupGNDesign-Report.pdf" where GN is replaced by your group number. PDF format is required. Word document format (.doc or .docx) is not acceptable. Attention: you will be assessed on your conformity to the instructions.

4.2 Minutes Protocol

10% of Coursework 2 A copy of your group minutes for three (or more) meetings held before the assignment deadline must be included in your submission to Blackboard. Your minutes files must be names "GroupGNMinutes-yyyy-mm-dd.txt" where GN is replaced by your group number, yyyy-mm-dd is replaced by the calendar date your group met (with yyyy replaced by the year, mm with the month, and dd by the day), e.g., Group3minutes-2019-02-23.txt. (You will be assessed on this.)

A **minimum of three minutes** of meeting protocol is required and are submitted on behalf of the group by one of its members. The minutes format should follow Bob's Minutes of Meeting Protocol. See blackboard for a copy of this document.

4.3 Contributions Report

5% of Coursework 2 A Contribution Report is to be included in your submission to Blackboard. This document contains a description of what and how each group member contributed to the project design is required. The file must be named "GroupGNMemberContributions.pdf" where GN is replaced by your group number.

Each group member is obliged to contribute at least one class, both design and implementation, to courseworks 2 and 3. The intention is that the classes each student chooses to design are also the classes they implement. However, students can change between assignments if necessary.

The Contribution Report, no more than 5 pages, describes who contributed to classes, hierarchies, subsystems in the design, and other contributions, e.g., the minutes etc. The Contribution Report is also written collectively. In other words, each group member describes their respective contribution to the project.

The report also informs the reader if any unexpected problems arose during the course of the assignment. For example, if a group member skipped too many lectures, and as a result didn't have the background necessary to contribute, this should be stated in the group report. Likewise, the group experienced success in some areas, both expected and unexpected, this is included.

5 Issues with Contribution

The group assignments assess the performance of the group when delivering a non-trivial piece of software. As mentioned in the course, real software is developed in teams and therefore group work is a necessary skill that needs to be developed. Individual contribution levels will mainly be assessed at the end of term when the groups are interviewed and each member has the opportunity to demonstrate their understanding of the course material in reference to the project. Under normal circumstances, this interview shall be used to adjust marks and determine the level of contribution. However, we realise that abnormal circumstances during group work may occur.

In the event of any non-contribution, please follow this procedure:

- As early as possible, discuss the situation with your lecturer.
- Work with your lecturer to try and get the non-contributor participating.
- Keep your lecturer posted on the situation.

In the case of significant non-contribution despite these efforts, a confidential non-contribution report can be filed to reduce the marks of non-contributors. This report will only be used to lower the marks of non-contributors. The report will not be used to raise the marks of any group member. The report should contain an explanation of the situation and the members involved. This report will only be accepted if the lecturer is aware of the situation and the group has taken all possible and reasonable steps to get the group member contributing.

This document must be sent in a confidential email to the instructor (Dr. Tom Owen, T.Owen@swansea.ac.uk).

Likewise, if any particular group member disruptively dominates group meetings and the assignment outputs in a way that is damaging to the team, the tutor should be notified immediately. Action on the marks of group members that are overly disruptive in this way. Please be supportive of all group members.

6 Project Hints

1. This coursework is very time consuming. Some teams usually lose marks because they don't manage to finish. Allow extra time for teamwork.
2. All group members are expected to contribute to both design and implementation. Group members who are weak at programming may get help from other members who are better at it. However, all group members are responsible for understanding the full submission. During
3. Ask questions during the problem help sessions.
4. Why not use Skype or Slack for some of your group meetings?