IFQ563 Assignment Specifications

# Overview

The goal of these assignments is to gain practical experience in object-oriented software development through object modelling with design diagrams and applying standard software design principles and patterns. This knowledge will be demonstrated by creating and documenting an object-oriented design and implementing it using C# for a real-world motivated problem.

While you can talk about the assignments with your OLA and peers, the submissions must be your own original work. Copying the work of others is not tolerated at QUT and submissions will be checked for code plagiarism against online repositories.

# Description

You are to provide an extensible framework for many different two-player board games. To demonstrate that your framework can be easily adapted to different games, your design (including all design diagrams) must include ONE of the following games in the same software:

* Numerical Tic-Tac-Toe: A variation of the classic Tic-Tac-Toe game. Two players take turns placing the numbers 1 to 9 on a 3x3 board. The first player plays with the odd numbers, the second player plays with the even numbers. All numbers can be used only once. The player who first puts down 15 points in a line (sum of a horizontal, vertical, or diagonal row of three numbers) wins the game. Any other variant of Tic-Tac-Toe is not acceptable.
* Wild Tic-Tac-Toe: Like the classic Tic-Tac-Toe game, two players take turns placing an X or an O piece on a 3x3 board. However, in this game players can choose to place either X or O on each move. The first player who creates a line of three X 's in a row or three O 's in a row (horizontally, vertically, or diagonally) wins the game. Any other variant of Tic-Tac-Toe is not acceptable.

To demonstrate the feasibility and effectiveness of the design, you must implement the games that correspond to your design using C# on .NET 6.

# Requirements

Your design should extract as many commonalities as possible from the different games so that the framework is extensible and reusable for a wide variety of board games.

Your system should cater for different modes of play, including:

* Human versus human
* Computer versus human

With human players, the system must check the validity of moves as they are entered. With computer players, the system can randomly select a valid move.

Games can be played from start to finish, and should be able to be saved and restored from any state of play (i.e. a save file). A game should be replayable from any position after being loaded from a save file.

During a game, all moves made by both players should be undoable and redoable (i.e. the full history of moves are tracked). But the history of the moves does not have to be saved into a save file. That is, immediately after loading the saved state from a file, undo and redo operations are not available (until new moves have been made).

You should provide at least a primitive online help system (i.e. accessible within the running game, rather than a separate offline help document) to assist users with the available commands (also give some examples if they are not obvious to use).

# Comments on requirements

* The requirements are what you should aim to achieve in your project. Even if you do not implement all features, you should at least cover all requirements in your design.
* Some requirements may be vague and open to interpretation. As long as you clearly state any assumptions you make in your design report, we will accept reasonable interpretations.
* The important aspect is that your design for the core structure and functionality of the two-player board game meets the requirements, in a clear and easily understandable way. You should steer your design towards a general framework and consider your implementation as a proof-of-concept of your design, rather than a fully-featured, highly intelligent game-playing program.
* Your implementation must be a C# console application on .NET 6 and provide a text-based command-line interface (e.g. using either ASCII or Unicode) of the game as it is played. Some marks have been allocated for usability, so be sure that whatever interface you use, the commands are simple and clear. The following screenshot shows a simple example of playing a game of the classic Tic-Tac- Toe by two human players, displayed in Unicode. For the game of Numerical Tic-Tac-Toe, numbers 1-9 are placed on the board.

