Assignment 2: Object-Oriented Design and Implementation

- Weight: **45%**
- Individual/Group: **Group**
- Due: **Friday 23 May (Week 12)**
- Project demo: **Week 13 workshops**

1. Overview

In this assignment, you will engage in a non-trivial object-oriented software design and implementation project. To simulate real-world software development practices, you will collaborate in a team (refer to Section 3) to design and implement an object-oriented software solution that satisfies the specified requirements (refer to Section 2).

The primary objective of this project is to provide hands-on experience in object-oriented software development by utilising object modeling with design diagrams and applying software design principles and patterns. You will demonstrate your understanding by creating and documenting an object-oriented design and implementing it in C# for a practical, real-world problem.

While you can talk about the project with tutors and peers outside of your team, the work you submit must be the combined efforts of your team members only. Copying the work of others is not tolerated at QUT and submissions will be checked for code plagiarism against online repositories. In addition, the use generative AI tools such as ChatGPT, including programming/coding assistants such as CoPilot, is strictly forbidden for all assessment tasks in IFN584 and will be considered an act of academic misconduct.

Both the group's achievement (35%) and your individual contribution (10%) will be assessed (refer to Section 4). You will be required to demonstrate your project during the Week 13 workshops (refer to Section 5).

2. Description

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

- Numerical Tic-Tac-Toe (as specified in Assignment 1).
- Notakto, also known as Neutral or Impartial Tic-Tac-Toe (Wikipedia). Two players take turns placing the same piece (e.g. an `X`) on a finite number of 3×3 board (in this project, we only use **three** boards). The game ends when all three boards contain a three-in-a-row of `X`s, at which point the player to have made the last move loses the game.
- Gomoku, also known as Five in a Row (Wikipedia). Two players take turns placing two types of pieces (e.g. an `X` and an `O`) on a 15×15 board (you can use a smaller board if that helps the UI). The winner is the first player to form an unbroken line of five pieces of their colour horizontally, vertically, or diagonally.

To demonstrate the feasibility and effectiveness of the design, you **must implement the games that correspond to your design** using C# on **.NET 8** (refer to Section 4.1.2).

2.1. 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 **two modes of play**, including:

- Human vs Human
- Computer vs Human

With human players, the program checks the validity of moves as they are entered. With computer players, the program makes a move that immediately win the game; otherwise, if no immediate winning move is available, the computer player randomly selects a valid move.

A game can be **saved and restored** from any state of play, which is stored in a save file. Upon loading a save file, the game resumes from the exact position it was saved, preserving game modes (HvH or HvC) as well.

During a game, human players can **undo and redo any number of turns** (meaning the entire history of moves is tracked). If you're not familiar with these operations, it's a good idea to check out some online board games, like chess.com or lichess.org. Additionally, if you load a saved state from a file, you can perform undo operations right away. Redo operations become available after you've made new moves.

The program should provide a simple **in-game help** menu system to guide users with the available commands. Additionally, it can provide some examples of commands that may not be immediately apparent to users.

2.2. 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 8** 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.

3. Teamwork

You will work in teams of **4 members** and distribute the workload fairly among your team members. All team members are expected to contribute equally and meaningfully to this group project.

You should have regular meetings to discuss your progress and any issues you encounter. This increases discussion opportunities and often leads to exploring ideas that may not have otherwise occurred to you as an individual. However, as in any relationship, working together may become difficult if group members have very different expectations regarding the assignment. Some difficulties may be avoided by discussing expectations with team mates before starting work together. For example, you may want to discuss the following points with your team-mate:

- What are you hoping to achieve for this unit/assessment item?
- How much time do you have available to work on this, when and where?
- What are your skills or strengths related to the tasks?

There may be other points that are also important to you. If you discuss these honestly before you start, then you know where you stand, and that can help to make your work together more effective.

You are required to register your team on the IFN584 Canvas (from the "People" section, select the "Groups" tab) as early as possible, but no later than **Friday of Week 9**. It is highly recommended to form your own teams, as this allows you to collaborate with individuals who share similar goals and expectations.

You can use the Canvas discussion forums to find team members and you may form a team with students from other workshops. If you are unable to find a team or register your team on the IFN584 Canvas by Friday of Week 9, the teaching team will assign you to an available team. Please note that this may not align with your preferences or expectations.

Only under special circumstances, a team of fewer than 4 members may be accepted. However, you must contact the unit coordinator to obtain approval for your team size. Note that the size of your team will not affect the assignment requirements or the grade you receive.

4. Submission

The assignment must be submitted via the IFN584 Canvas website under the "Assignments" section. The marking criteria/rubric for the assignment are detailed on the submission page. You are encouraged to use these criteria to self-assess your work before submission.

4.1. Group submission (35%)

You must submit **two separate files**: **one PDF file** of the design report and **one ZIP file** containing all your C# project files. Only one of your team members needs to make this submission (i.e., not everyone in the team needs to submit). Each team will receive a single grade for the group submission.

4.1.1. Design report

You should only submit one PDF document with no more than 12 pages in length (A4 page size with 2cm in margins on all sides; 12-point Times New Roman or 11-point Arial font or something equivalent, and in single space).

Your report should include:

- Executive summary (up to 2 pages)
  - A list of all team members (full names, student numbers and emails) along with your team number.
  - A declaration of **contributions of each of your team members**. Please note that any team member who didn't contribute substantially to the project will receive zero mark for this assignment.
  - A statement of completion, clearly **declaring the requirements that have and haven't been implemented**. This declaration must be complete and accurate, in order to receive marks for the "Fulfilment of requirements" in the marking criteria.
- Design documents (up to 5 pages)
  - An **overview of the overall design** and a brief summary of the design process, including the changes or improvements you made compared to Assignment 1. Ensure that your design diagrams are clean, readable, and effectively illustrate your software design based on the project requirements.
  - A single **class diagram** that includes all classes in the software, clearly showing their relationships, attributes, and methods;
  - Two **object diagrams**, each illustrating a snapshot of the program's memory at a specific and meaningful moment during execution, with clear labels for the scenarios depicted;
  - Two **sequence diagrams**, each representing a significant scenario and showcasing different key events during the software's execution, with the flow of interactions.
- Design patterns and principles (up to 3 pages)
  - Identification and a brief explanation of the design patterns and principles used in the design. For each pattern or principle, **clearly identify the participating classes and key operations** from the design diagrams, and provide a justification for their inclusion in your design. Ensure that the explanation highlights how these patterns and principles contribute to the extensibility, reusability, and maintainability of the framework.
- Implementation details (up to 2 pages)
  - A concise guide on **how to execute your program**, including any necessary setup or dependencies.
  - A brief summary of **external libraries or frameworks used**, listing the classes/interfaces (e.g., from the Collections library) that are reused, without requiring detailed explanations.

Remember that clarity and simplicity are key to a successful design. Strive to make your design neat, concise, and self-explanatory. Use explanations sparingly, focusing on areas where your intentions might not be immediately clear.

4.1.2. Implementation source code

You must submit a working implementation including full C# project source code for .NET 8. You should zip the whole folder containing the project file (`.csproj`) and upload this ZIP file onto the IFN584 Canvas website.

Your class implementations must strictly adhere to the documented class designs in your design report. This means that the classes in the source code must precisely match the classes defined in the class diagram, including their attributes, methods, and relationships with other classes. Unfaithful class implementations will receive zero for implementation.

You must make sure that your submitted code can be compiled and run properly with .NET 8. The submitted project files will be compiled and tested on QUT lab computers with .NET 8. Uncompilable or inexecutable source code cannot be marked and will receive zero for implementation. To confirm the version of .NET on the computer, simply open a terminal and run the following command:

```
dotnet --version
```

To check that your project code can be compiled and executed on .NET 8, open a terminal in the folder containing the project file (`.csproj`) and run the following commands:

```
dotnet clean
dotnet run
```

4.2. Individual contribution (10%)

To ensure fairness and clarity in evaluating individual contributions while aligning with the group submission grade, we will use the peer review system in Canvas. You will be required to submit a peer review on each of your three team members' contribution to the project.

You can access the assigned peer review link from the group project assignment page on Canvas. The link will become available once the group project is submitted. This ensures students have the final project to review.

You can use the provided rubric to evaluate each team member's contributions to the project. Consider aspects such as participation, quality of work, communication, collaboration, and timeliness. Additionally, you should provide brief comments to justify your assessment. The peer review process is anonymous, and the results will be used to adjust individual contributions based on the group submission grade. Note that the peer review grade will be capped by the group submission grade. For instance, if the group submission received 50% of the total points, the maximum peer review score for any individual will be 5 out of 10.

5. Demonstration

Before your submissions can be graded, you will be required to demonstrate your project during the Week 13 workshops. Each team will be allocated a 10-15 minute time slot for their demonstration:

- Teams 1-10 will demonstrate during the Tuesday workshop.
- Teams 11-20 will demonstrate during the Wednesday workshop.

If you have a preference for your demonstration time, please select the appropriate team when registering your team on Canvas. Attendance at your scheduled demonstration time is mandatory, and rescheduling will not be possible due to limited time slots.

During the demonstration, you will need to:

- Provide a brief explanation of your design and implementation.
- Demonstrate the functionality of your project, showcasing how it meets the requirements outlined in this document.
- Highlight how the design patterns and principles used in your design are reflected in your implementation.
- Answer questions from peers and tutors. Please reserve a couple minutes at the end of your demonstration.