Design Patterns Task 1: Implement a Game: Connect Four

The goal for the exercise is to be able to apply design patterns and come up with readable and maintainable software architecture and design. For this assignment, you will have to develop an application in groups of two. The application is a simple and well-known board game. For this application, you should come up with a software architecture and design and also implement this. Afterward, this implementation will be shared with other students who have to write a review about this. As the last step, each group will present their project, focusing on the implemented design patterns and the core ideas of the software architecture.

Constraints

- **Group size**: 2 Students.
- **Time Effort**: approx. 20h-30h per student.
- **Programming Language**: You can freely decide on the language or environment but use a common programming language. Other students must be able to read your source code to write their reviews. Execution is not needed for the review.
- **Graphical User Interface**: The game has to have a simple graphical user interface. Just a console application is not enough.
- **Design Patterns**: The implementation should involve at least four design patterns (either just applied or self-implemented if reasonable).
- **Submission**: The project must be uploaded as a .zip-file into the TeachCenter, with a maximum file size of 10 MB.
- **Deadline**: 18.11.2021 23:59

Project Requirements

Topic: Connect Four

You have to implement the board game Connect Four (“Vier Gewinnt” in German). For this, you have to develop a software architecture and design for the game and implement a simple graphical user interface with which the user can interact and make moves. You also have to implement the game rules. The design must also involve at least four design patterns, but most likely, you will use more anyways.

Requirements:

- The game should be **playable**. It should be possible to start from a fresh board, and there must be a winning condition when the game ends.
- You should implement a (very simple) **graphical user interface**. It should display the current state of the board and allow the user to make moves. Depending on the programming language and the platform, this can look completely different (e.g., Webpage, Desktop App, DirectX, iOS, Android, …). You don’t have to put too much effort into graphics – the exercise is about programming, not graphic design or game design.
- A user should be able to make **only valid moves**. There must be a possibility to make moves (either textual, using drag and drop, or other means – you can decide this by yourself).

- Users **alternate** between moves. White begins, black continues, and then they take turns on every move.
- It should be possible to **undo** a move. Also, multiple undo should be possible (undoing the moves of both players) until the start of the game.
- There should be the possibility of **redo**. Also, multiple redo should be possible (redoing or “replaying” the moves of both players) until the last move a player did was redone again.
- It should be possible to **save** the state of a game in a file, load it again, and continue playing.
- By reloading and undoing all steps, a complete step-by-step **replay** of a game should be possible.
- If a player decides to deviate from previous steps during such a replay, the game should continue normally, like in a regular play. In this case, further redo is not possible (the redo-chain is broken).

Optional:

- Implement a **computer opponent**. It does not have to be very intelligent but should only make valid moves.

Important for the grading and the reviews:
Other students will review your application and try to understand the architecture and design. They will also search for design patterns. Therefore, the better understandable your software architecture is, and the better readable your code is, the easier it will be for others to understand. Write your code with the fact in mind that at least three other students from the course will have to read and understand it.

The grading will consider the reviews as hints, but the grade will be based on our own assessment, judging the following aspects. The peer reviews will be more used towards the grading of the reviewer as evidence that the student has tried to understand the other projects.

Grading aspects:
- Is the architecture reasonable and extensible?
- Does the design incorporate design patterns in a meaningful way?
- Is the code readable and maintainable?