Lab 4 TerribleTicTacToe

Task 1: find at least four design principles or patterns that this code design lies at odds with.
From the GRASP pattern:

1. **Controller** - Unclear controller position, as the class acts as a controller but has mixing different responsibilities within the class.
2. **Information expert** - this rule states that a class with most knowledge should also be responsible for performing that task , however with such a mixing of different roles this violates the information expert rule.
3. **Low coupling** - the classes have low coupling. Coupling refers to how well the elements have knowledge about each other, and you want this to be low otherwise there will be so-called dependencies which makes it harder to change any part without impacting another part.
4. **High cohesion** -Cohesion refers to that a class should only have a small set of responsibility, this makes it small and maintainable. Have too many different responsibilities: interaction, game logic and UI management this makes the code harder to maintain and understand. (Single responsibility principle).

Class diagram - The classes responsibilities

1. Board

Responsible for the moves and the games board's state.

2. Controller

Manages interactions between board and gameView, a typical form of a responsibility that a controller has in an MVC pattern, an intermediate class between the Board and Gameview. Handles the button clicking with the method handleMove() additionally updating and controls the game state, by using the methods board.isWin() and board.isFull() if there is a draw.

3. GameView

Handles the visuals meaning the user interface (GUI) for example the playerTurnLabel which shows the user which players turn it is, updating the status for updateButton which is responsible for showing the user the "X" or "O" and further alike.

4. Launcher

Essentially our “main” code that is responsible for starting every tictactoe game, by creating new instances and initializing the Controller by passing on the Board and GameView.

5. Player

A placeholder that is responsible for our players in the game.

6. RuleEngine

Responsible for the game rules, which is done by checking the last move made for example the isLineWin() is responsible for checking all of the rows and columns by iterating through all of them and if it finds that all of the elements belongs to the same player the method return a boolean type true for winning. The isDiagonalWin() will iterate and works similarly but this method() will iterate diagonally.