Lab 4

1.

The 4 design principles/patterns:

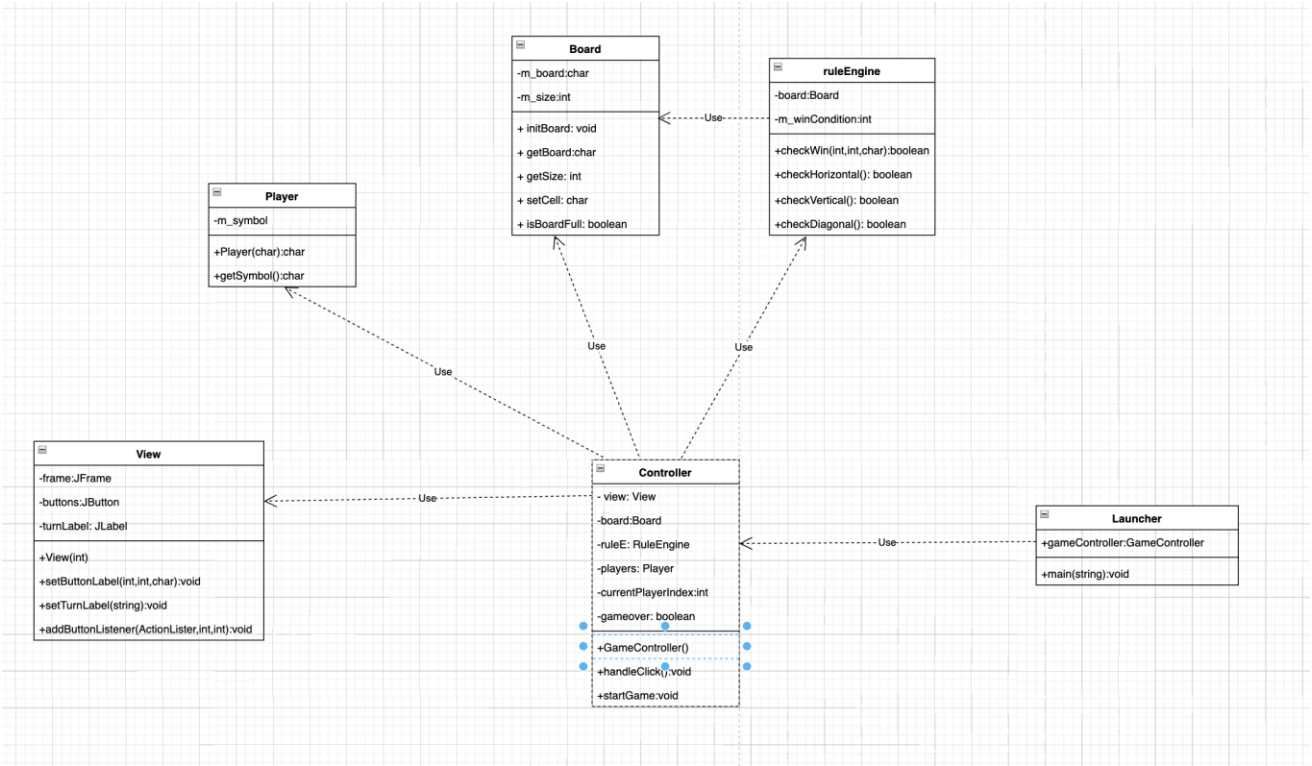
Single Responsibility Principle (SRP): The code violates SRP by handling multiple responsibilities in a single class, including game logic, GUI creation, and event handling.

Don't Repeat Yourself (DRY): The code contains duplicated logic in button action listeners, resulting in redundancy and making it prone to errors.

Composite Pattern: The code manually manages individual GUI components, which becomes cumbersome as the interface complexity increases.

Observer Pattern: The code relies on a boolean flag (gameover) to track the game state without providing a mechanism to notify interested parties when the state changes.

2. Class diagram:



3. Responsibilities:

Board: Initilizes and manages a board. Is also checks if the board is full.

Controller: sets up the necessary components, including creating instances of the Board, Players, RuleEngine, and View. It is also responsible for starting the game with a function. It It interprets player moves, checks for win conditions, and updates the board accordingly. It also communicates with the View to update the game's display.

Launcher: Initiates the game and creates instance of controller. Calls function from controller to start the game

Player: Represents a player in the game. Gives the player a symbol to play with.

View: Creates and manages the games GUI, It displays the gameboard and updates it based on player moves and game events.

RuleEngine: Checks for win conditions, it communicated with the controller to notify about game outcomes.