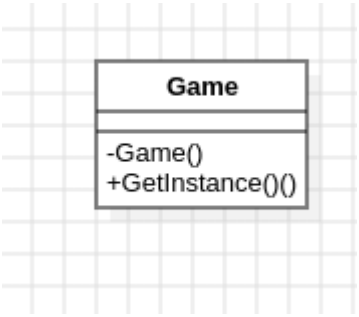
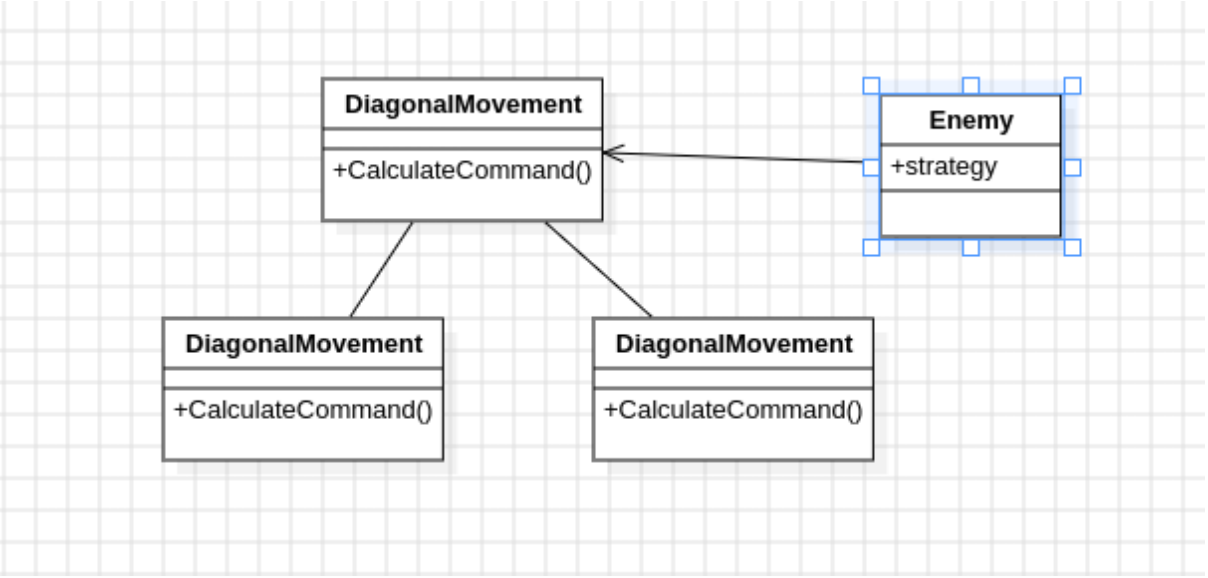
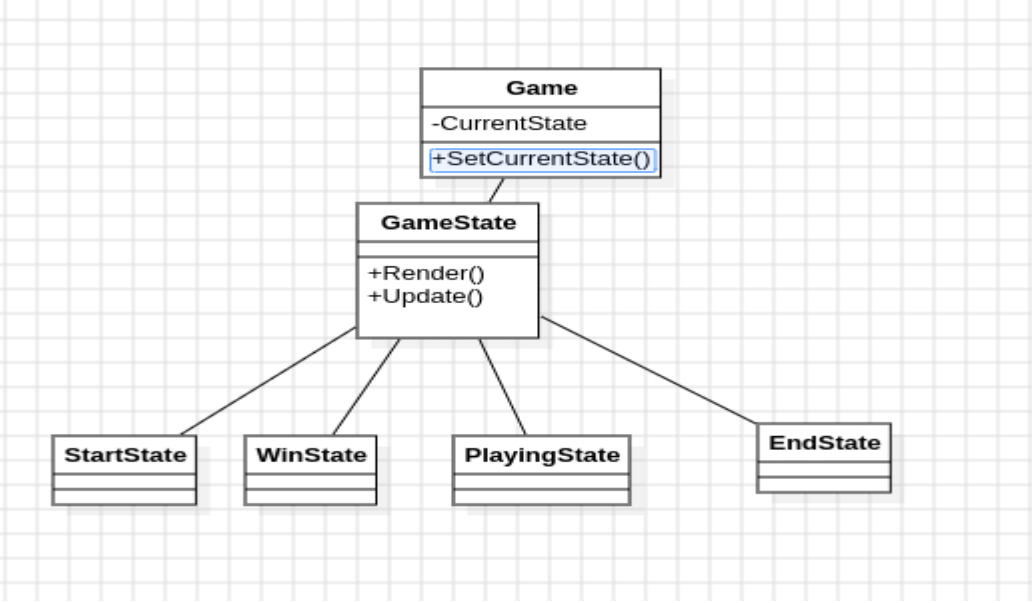
1.Singleton Design pattern is used for the game object to ensure that only one game is running at a time and there is no other windows that start popping up



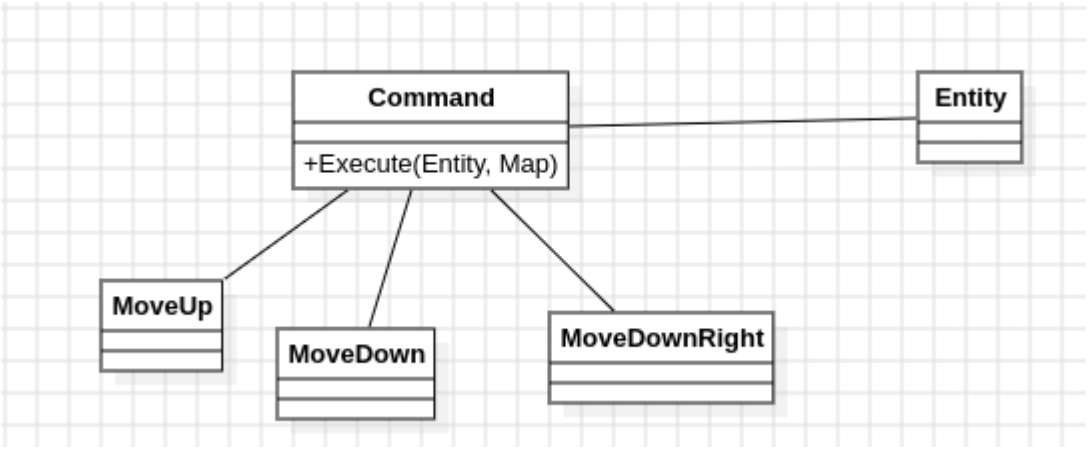
2.Strategy Design pattern is used for the way in which enemies move in the game, new strategies can made without modifying either the movement witch is controlled by the commands nor the enemy witch only decides how it wants to move



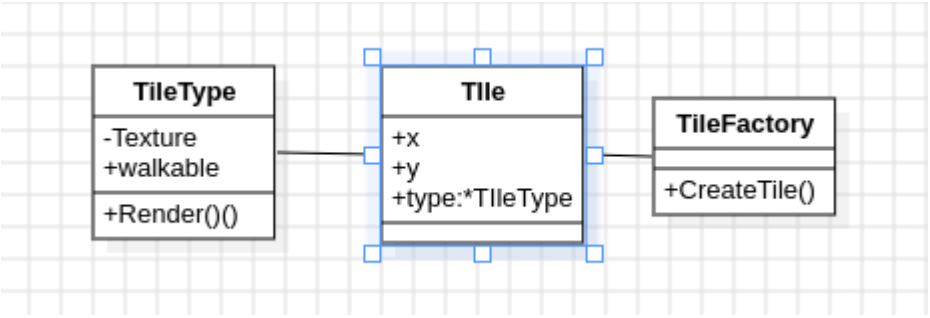
3.State Design pattern is used to manage the states of the game(start, playing, win, lose) without the need for modifying the game logic and making it easy to jump from



4.Command Design is used for the movement of the entities in the game(players and enemies) without the need to modify the entities classes and there can be added other commands that are not movement related.



5,6. The Flyweight Factory is a class responsible for creating and managing shared Flyweight objects. It ensures that no duplicate flyweights are created if a requested object already exists, it returns the existing one instead of creating a new one.



7.Observer Design Pattern is implemented with the player as the subject and the moveUI as the observer. In this implementation, the moveUI observes the player's state (position, health, inventory, etc.) and updates accordingly without the player needing to know about the UI specifics. When the player moves, takes damage, or interacts with the environment, it notifies its observers (including moveUI), which then update their displays based on the new player state. This decouples the player logic from the display logic, allowing either to change independently without affecting the other.