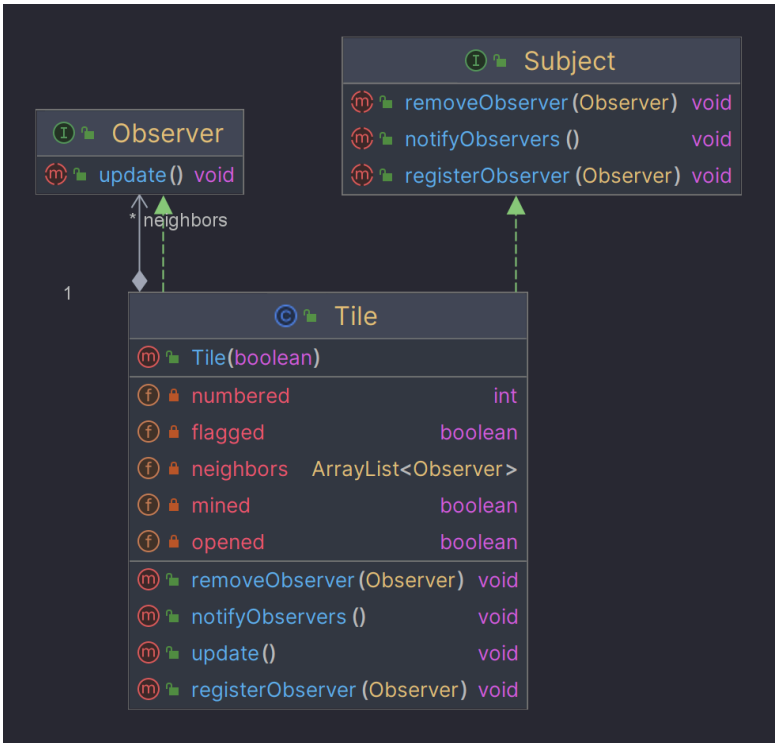
Minesweeper
Design Patterns

Observer

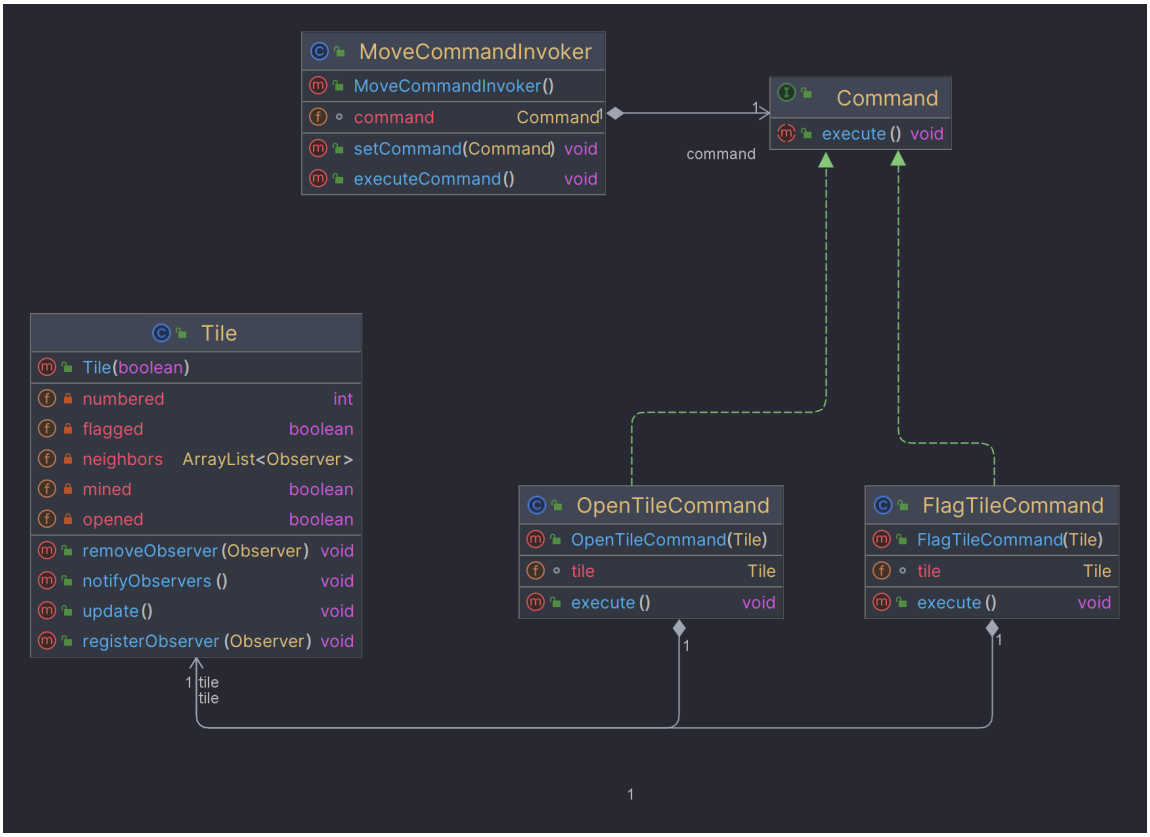


Description
The Observer pattern is used by making Tiles both a Subject and an Observer. Each Tile then holds an ArraryList of its neighbors. Thus when a single tile is updated we can update all its neighbors recursively if needed. Then once we update all tile neighbors recursively we can re render the game board to display the updates.

Advantages
- Can update tiles quickly and recursively when a single tile changes
- If we wish to add a new property to a file important to an update we only need to change the update method of Tile.

Drawbacks
- Having a Tile be both a subject and an observer may be hard to understand
- The update method me become very complex based on the number of details needed to be taken into consideration

Command



Description

The Command pattern is by creating 2 simple commands. The OpenTileCommand will open a tile and then update the tile with a number a mine if there was one or a blank and recursively show blank tiles that were its neighbors. The FlagTileCommand will mark a tile as flagged and update the visuals accordingly on the board. When we want to call a specific command we'll use the invoker and the setCommand method to choose which command we want then execute it.

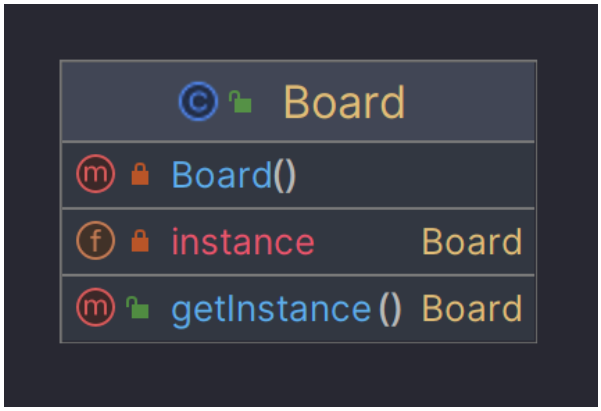
Advantages

- Delegates responsibility to the invoker then to a specific command when updating the board
- Can easily add new commands for new moves in the future

Drawbacks

- Currently we only have two commands but if we added more we would need to change the invoker to be able to hold many commands at once to avoid calling setCommand many times

Singleton



Description
The Singleton pattern is used to represent the board. It is implemented as a Double-checked Locking Singleton. Whenever we need the instance of the board we will use the static method getInstance. The Board will have a series of methods for playing the game as well not shown in the UML above that will delegate player actions to the command invoker.

Advantages

- Singleton is a good choice as we only need a single instance of a board throughout the whole application
- Since there is only a single instance we can't accidentally create another instance and perform actions on it erroneously

Drawbacks

- Since the board is a singleton it's a global instance of a class and anything is able to reach for it and modify it in a potentially malicious way