Lecture 33 :

Build Tic Tac Toe Game

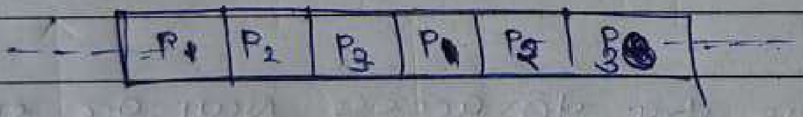
Requirements :-

① Size of board should be scalable.
② These are standard game rules & should be further extensible.
③ Allow an app notification for Tic-tac-toe moves, wins, draws etc.

UML Diagram

- create it using top-down approach.
- Start with Game class which will be our main class.

- we will use deque it would manage turns of players efficiently.

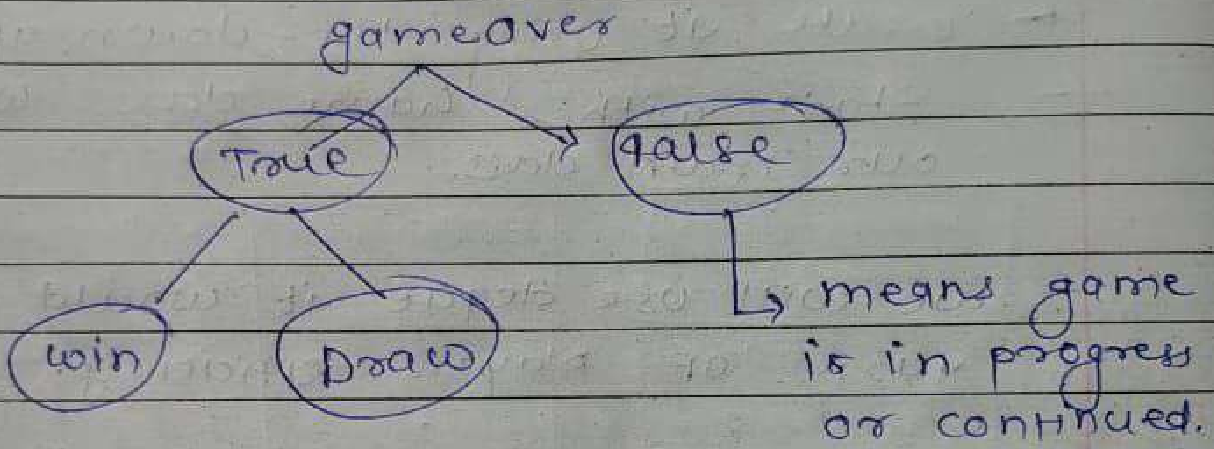


: This is How turn will manage.

① We will create player class.
② Now our last requirement is on App Notification (observer pattern).
③ On App notifications, player will get the notification such as logs, or alert msgs which will be visible to all players. like invalid move, draw, fail or win game.

④ our player class will Not be a observer class. but there will be a consoleNotifier class which will display Notification on screen.

⑤ Now there will be a class variable flag variable which will be help to decide the game is over or Not.



⑥ If we try to access Null or empty value in grid then we get Nullptrexception so we place '_' underscore in empty cell to avoid Nullptrexception.

⑦ · Board follow SRP (SingleResponsiblity principle) because it has only one responsiblity to manage board only.
Board doesn't have information about the rules.

(8) Now we create rules class to create rules having some methods:-

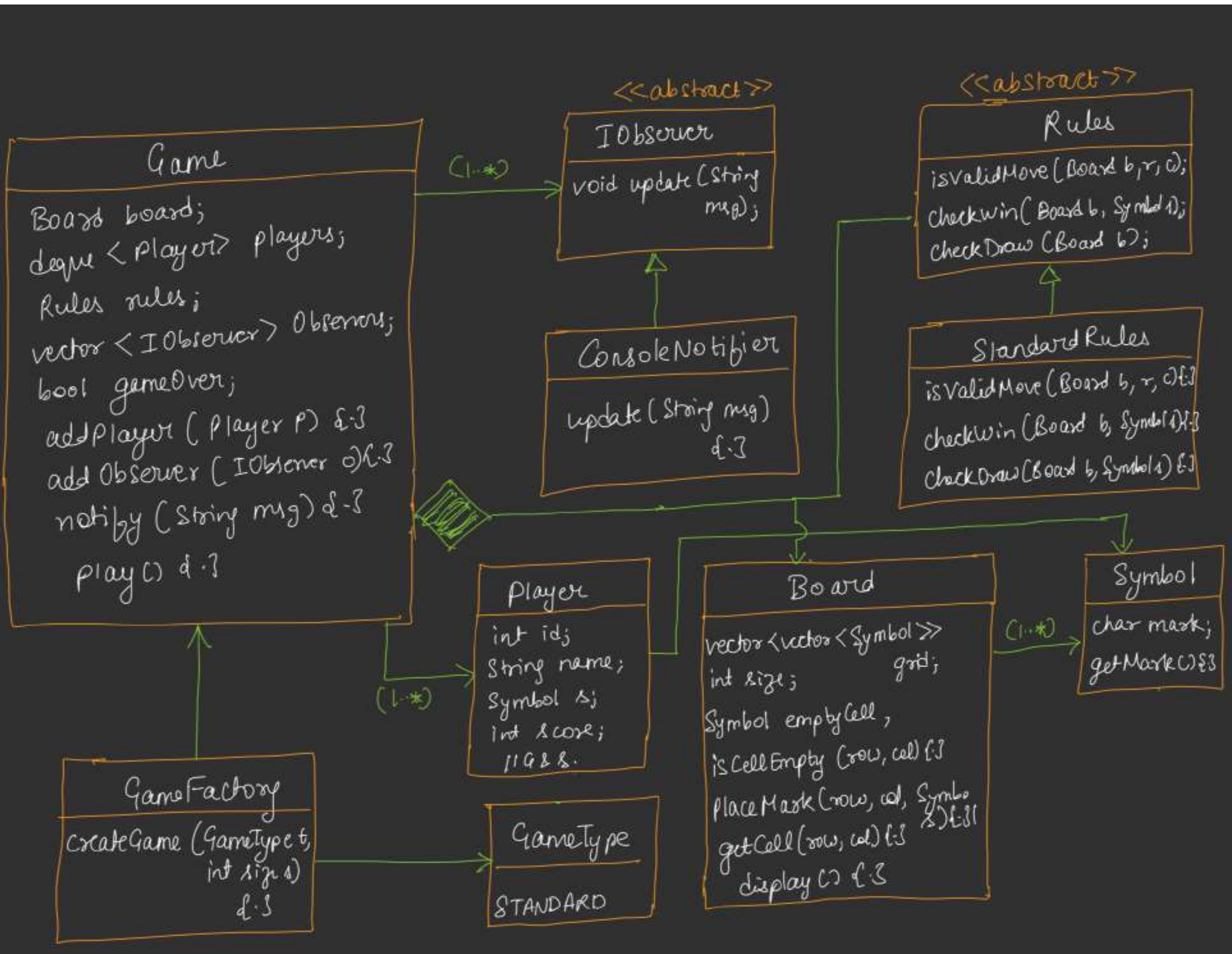
checkwin(Boardb, symbol s); → checks anyone win or Not.

checkDraw (Board b) → check game is draw or not.

isvalidmove(Board b, r, 1); → check whether move is valid or Not.

(9) Rule will be a abstract class and concrete class will be different rule strategy.

(10) Now we create players but we will Not use list to store the players becoz we haveto creat the scalable game and in future the No. of players can vary.


Game
Board board;
deque < player> players;
Rules rules;
vector < IObserver> Observers;
bool gameOver;
addPlayer (Player P) {-}
add Observer (IObserver o){-}
notify (String msg) {-}
play() {-}
(1..*)
<<abstract>>
IObserver
void update (String msg);
ConsoleNotifier
update (String msg) {-}
<<abstract>>
Rules
isValidMove (Board b, r, c);
Checkwin(Board b, Symbol s);
CheckDraw (Board b);
StandardRules
isValidMove (Board b, r, c){-}
checkwin (Board b, Symbol s){-}
CheckDraw(Board b, Symbol s) {-}
Player
int id;
String name;
Symbol s;
int score;
// G & S.
(1..*)
Board
vector < vector < Symbol >> grid;
int size;
Symbol emptyCell,
isCellEmpty (row, col) {-}
placeMark (row, col, Symbol s) {-}
getCell (row, col) {-}
display () {-}
(1..*)
Symbol
char mark;
getMark () {}
GameFactory
createGame (GameType t, int size) {-}
GameType
STANDARD