Programming Technology

Md Farid wefkhb 4th Task

Task -Description

Create a game, which is a variant of the well-known five-in-a-row game. The two players can play on a board consists of n x n fields. Players put their signs alternately (X and O) on the board. A sign can be put only onto a free field. The game ends, when the board is full, or a player won by having five adjacent signs in a row, column or diagonal. The program should show during the game who turns.

The trick in this variant is that if a player makes 3 adjacent signs (in a row, column or diagonal), then one of his signs is removed randomly (not necessary from this 3 signs). Similar happens, when the player makes 4 adjacent signs, but in this case two of his signs are removed.

Implement this game, and let the board size be selectable (6x6, 10x10, 14x14). The game should recognize if it is ended, and it has to show in a message box which player won (if the game is not ended with draw), and automatically begin a new game.

Evaluation of Task

I need to Create a two-player game where players place X and O on a selectable

n X n board (6x6, 10x10, or 14x14), aiming to align five consecutive signs in any direction. Unique twist: aligning 3 signs removes one random sign, and 4 signs removes two. Display turns, announce the winner or draw, and automatically restart after each game.

Plan to solve this task

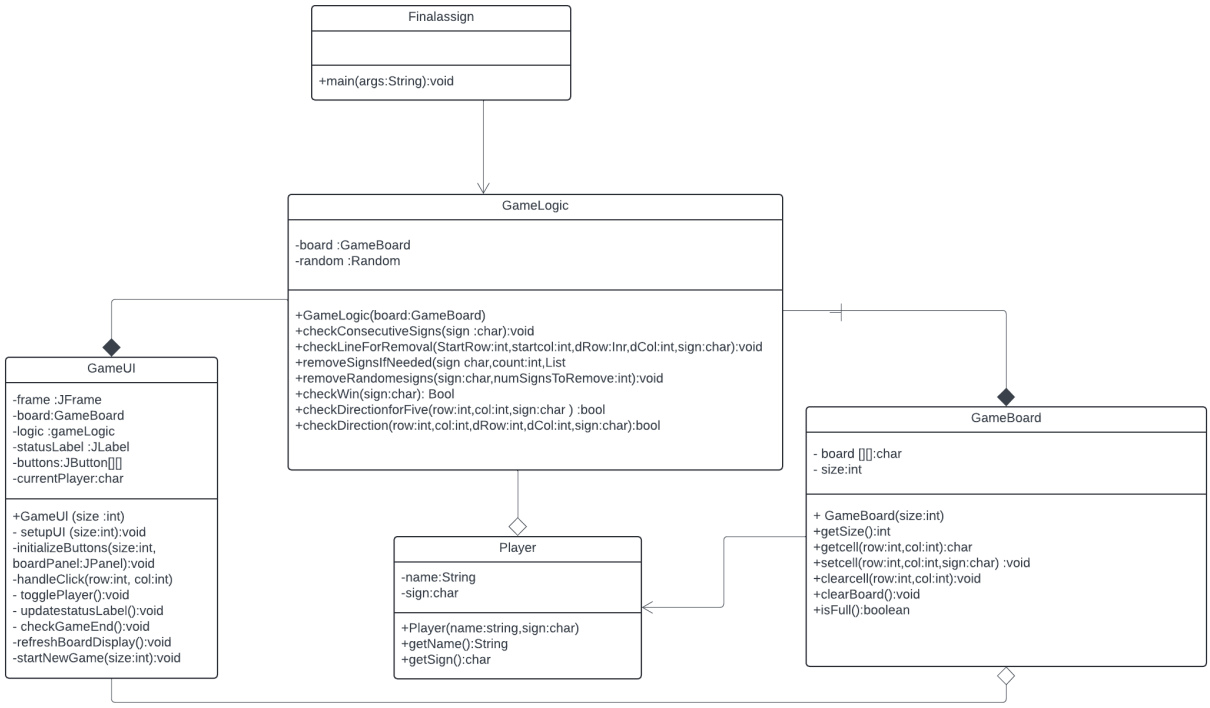
1. GameBoard : This class handles the game board's state, including setting, clearing, and printing cells, and checks for a full board. It also verifies if a player has won by having five consecutive signs in any direction.

2. GameLogic: The core game logic is managed here. This class checks for consecutive signs in rows, columns, and diagonals. It also handles the random removal of signs when three or four consecutive signs are placed. It interacts with `GameBoard` to update the board state and verify win conditions.

3. GameController: This class oversees game flow, including turn management and handling board interactions (placing/removing signs). It coordinates with `GameLogic` to check for consecutive sign placement, triggering removal rules, and determining the game's outcome. It also restarts the game after each round.

4. GameUI: This class manages the user interface using `JFrame` to display the game board and results. It listens for player actions (such as placing signs) and updates the board visually. It displays win or draw messages, restarts the game automatically after each round, and tracks the board's status during gameplay.

Remark :- for the Function detail prefer the java doc



White-box testing **:- is a software testing approach that examines the internal structure, logic, and code of a program. Testers focus on code paths, logic, and boundary cases to verify functionality and ensure every code segment performs as intended.**

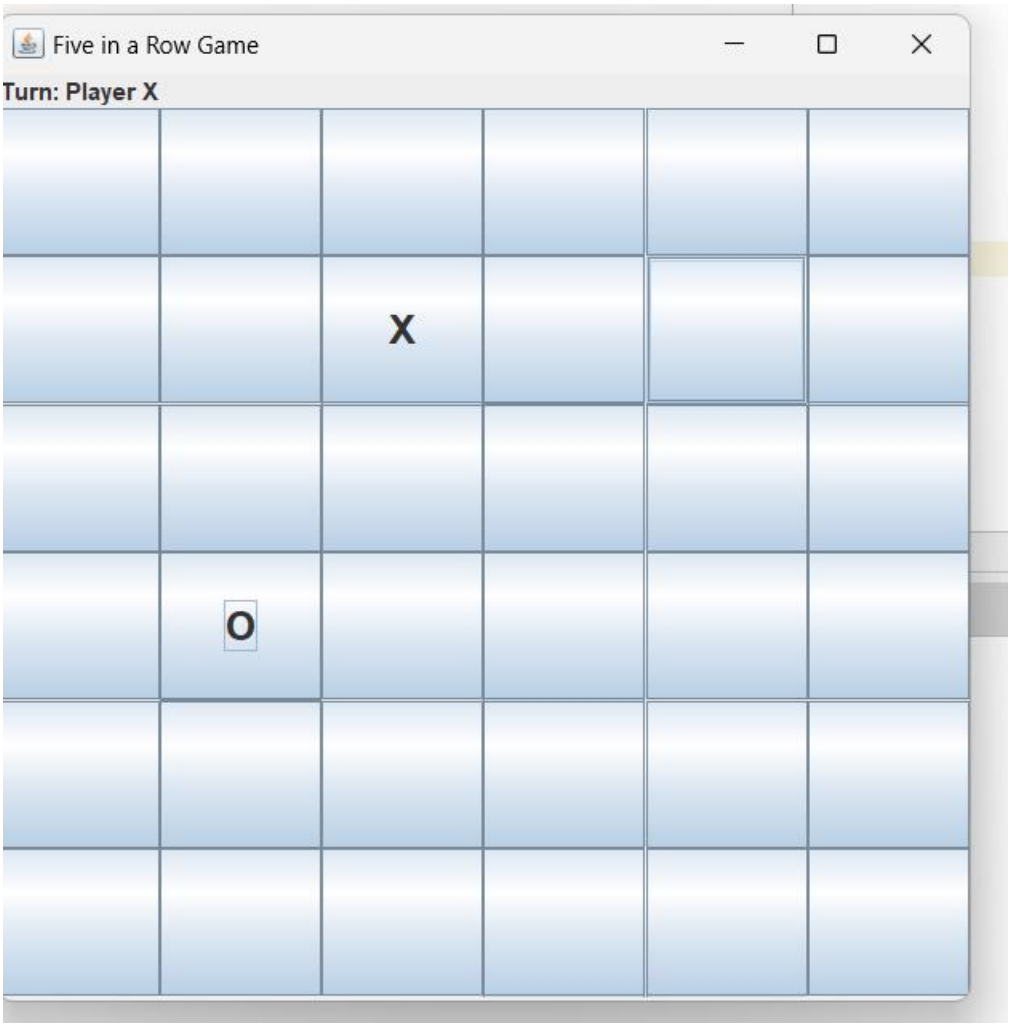
Black-box testing :- is a software testing technique that assesses the functionality of a program without knowledge of its internal code or logic. Testers evaluate input-output behavior to confirm that the software meets specified requirements and handles user scenarios correctly.

Testing

1 Invalid move :

When I try to place any symbol on the occupied cell then the game should reject that move .

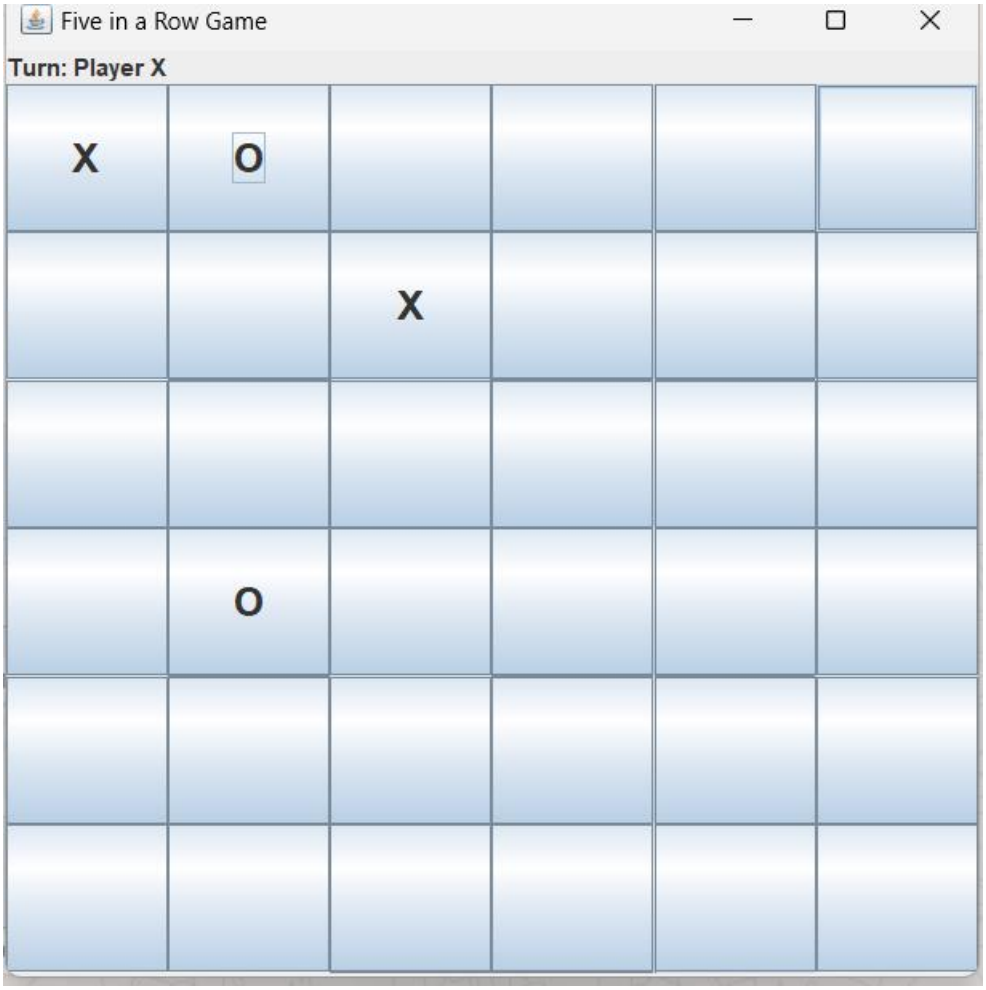
I can not replace the sign on the already filled cell.



I can not replace the sign on the above already filled cell .

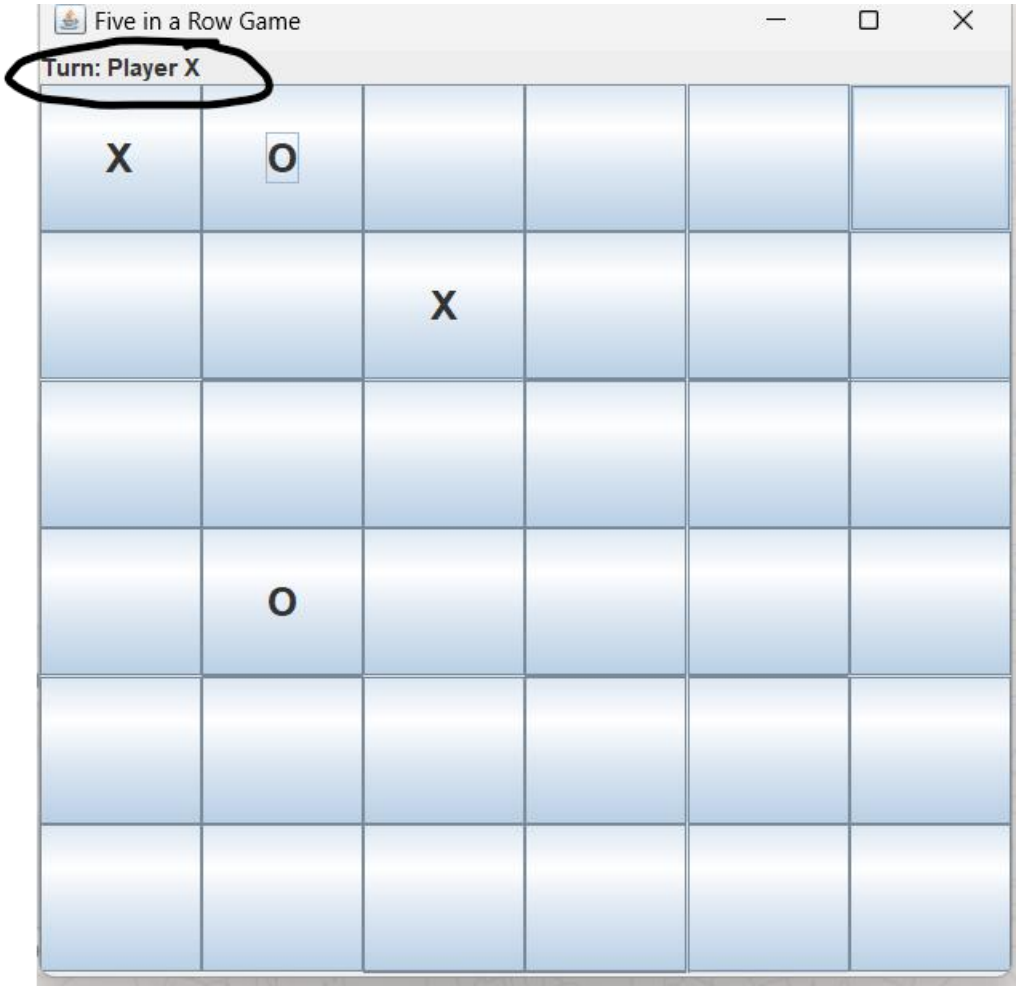
2 Valid move :

I can place the sign on those cells those are Empty only



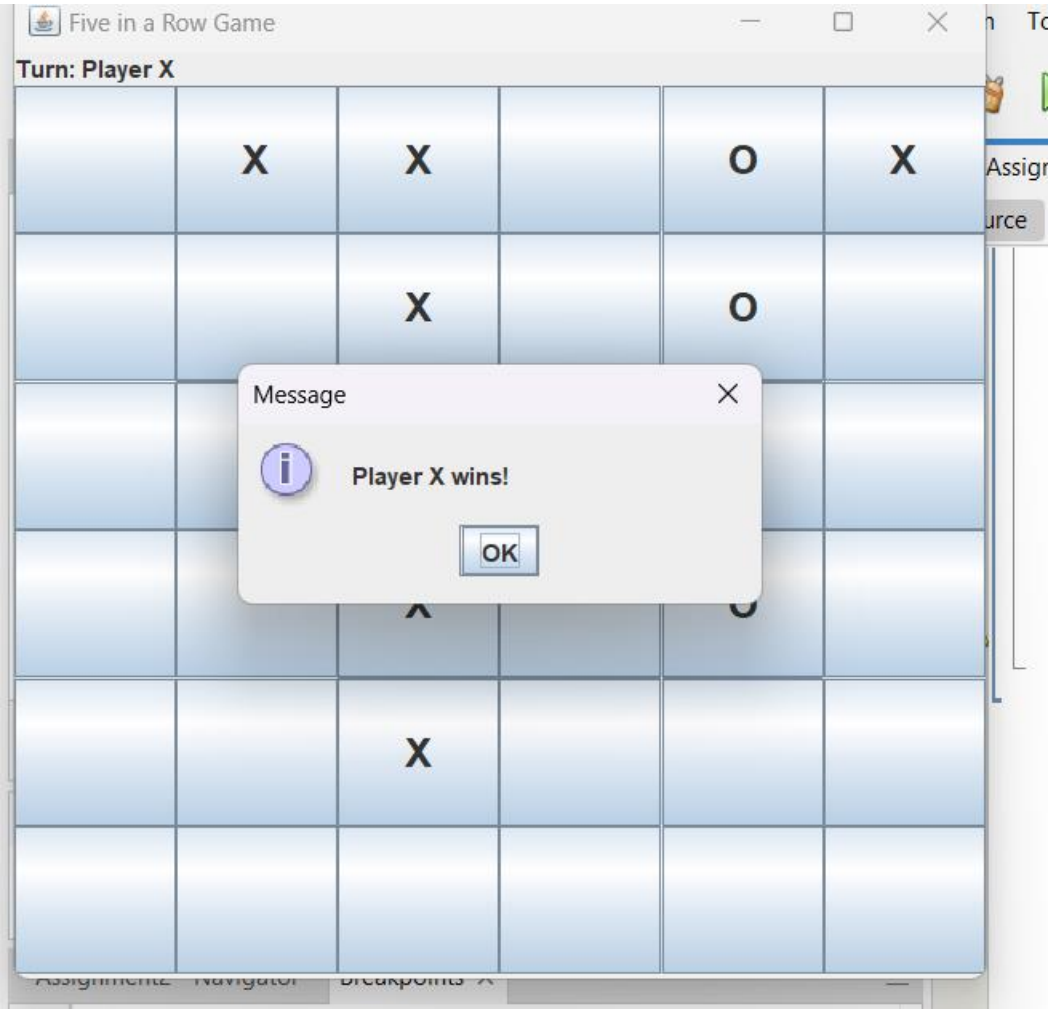
3 Player turn :a single player can have play only one time then other player should play his turn ,

A single player can not place same sign continuously twice in the board



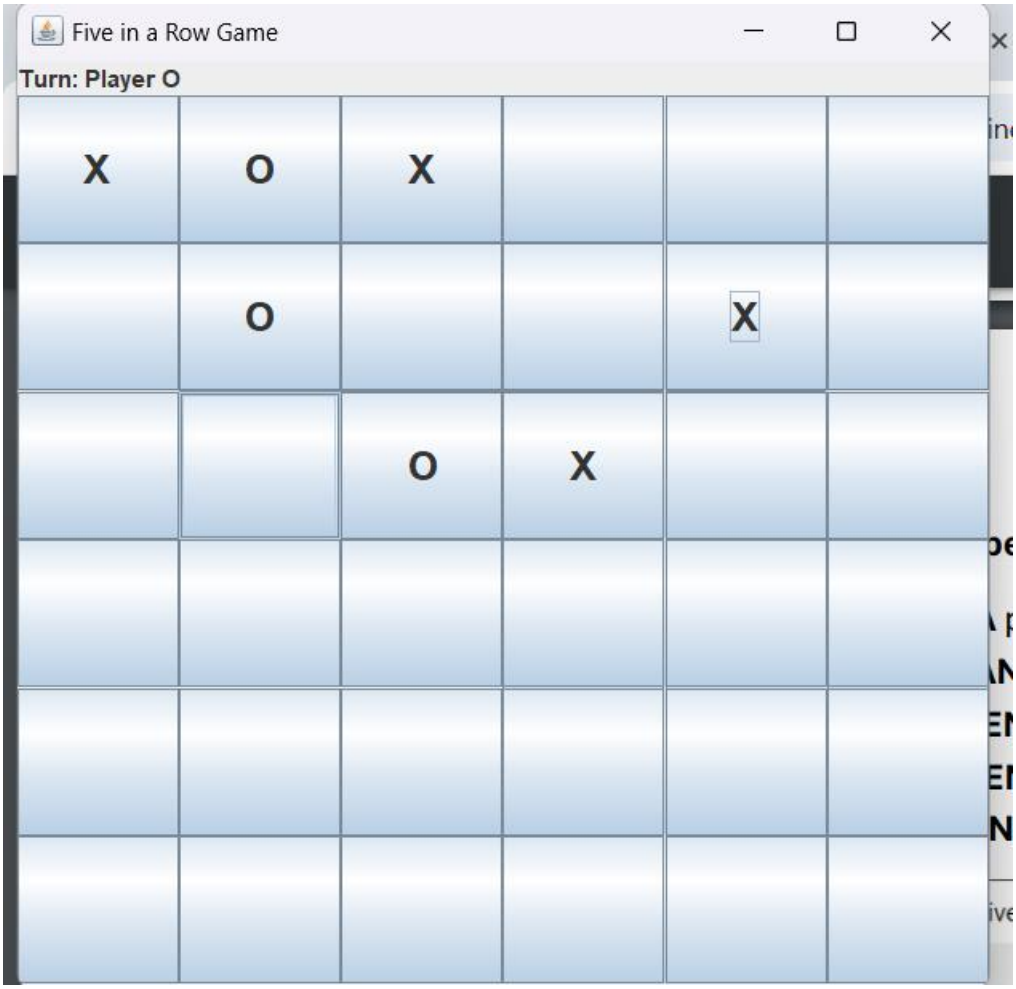
Afte the one player turn the next player should play his turn

4 game win condition :- the user will win the game if he make continous 5 sign row column or diogonal wise

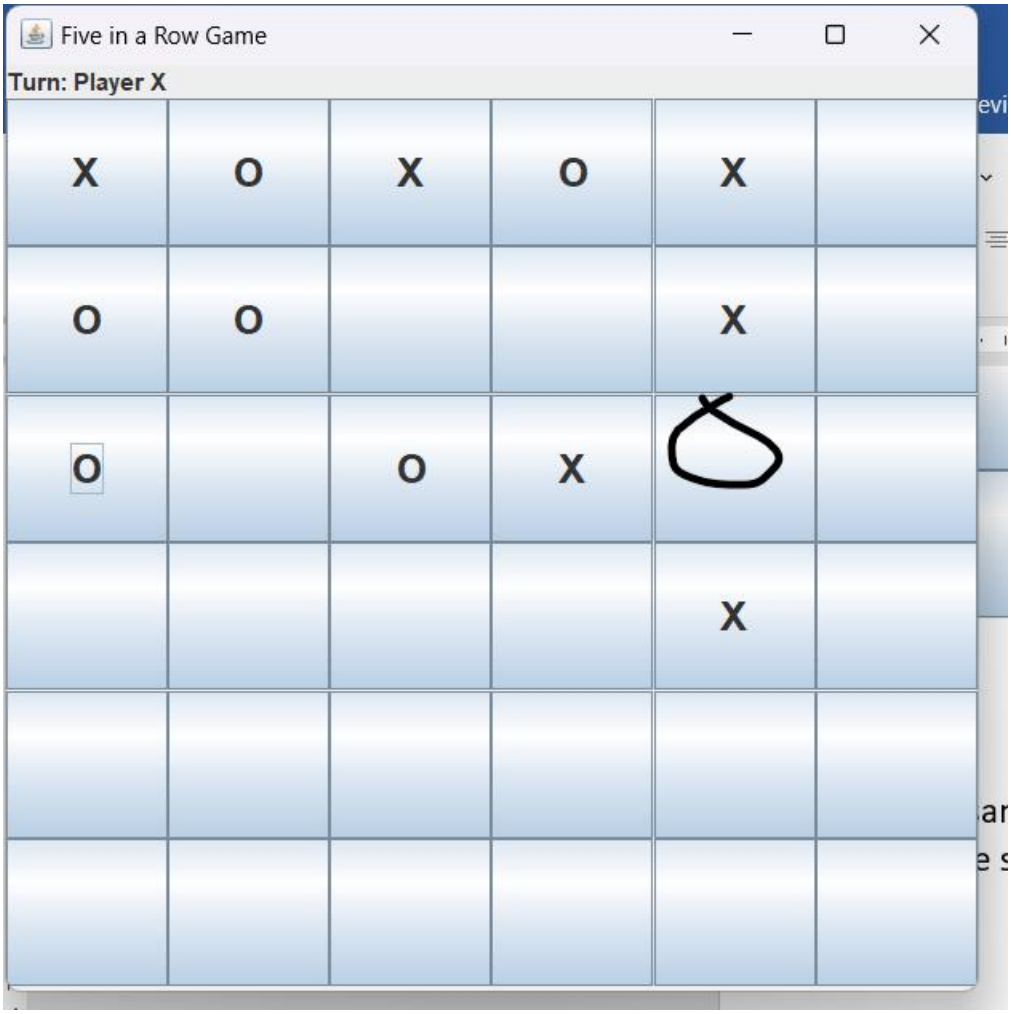


5 when same sign 3 times in a row ,column or diognal wise :

In this situtation I will remove the same sign randomly from anywhere from the board

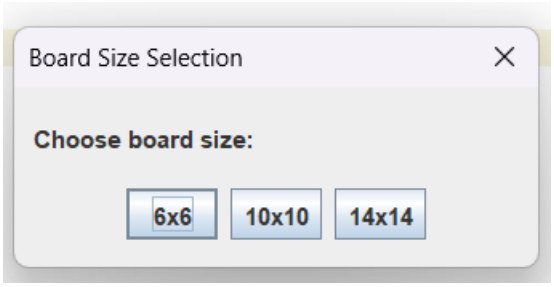


6 if same sign appear 4 times in a row column or diagonalwise :- then remove the same sign 2 times from anywhere from the board

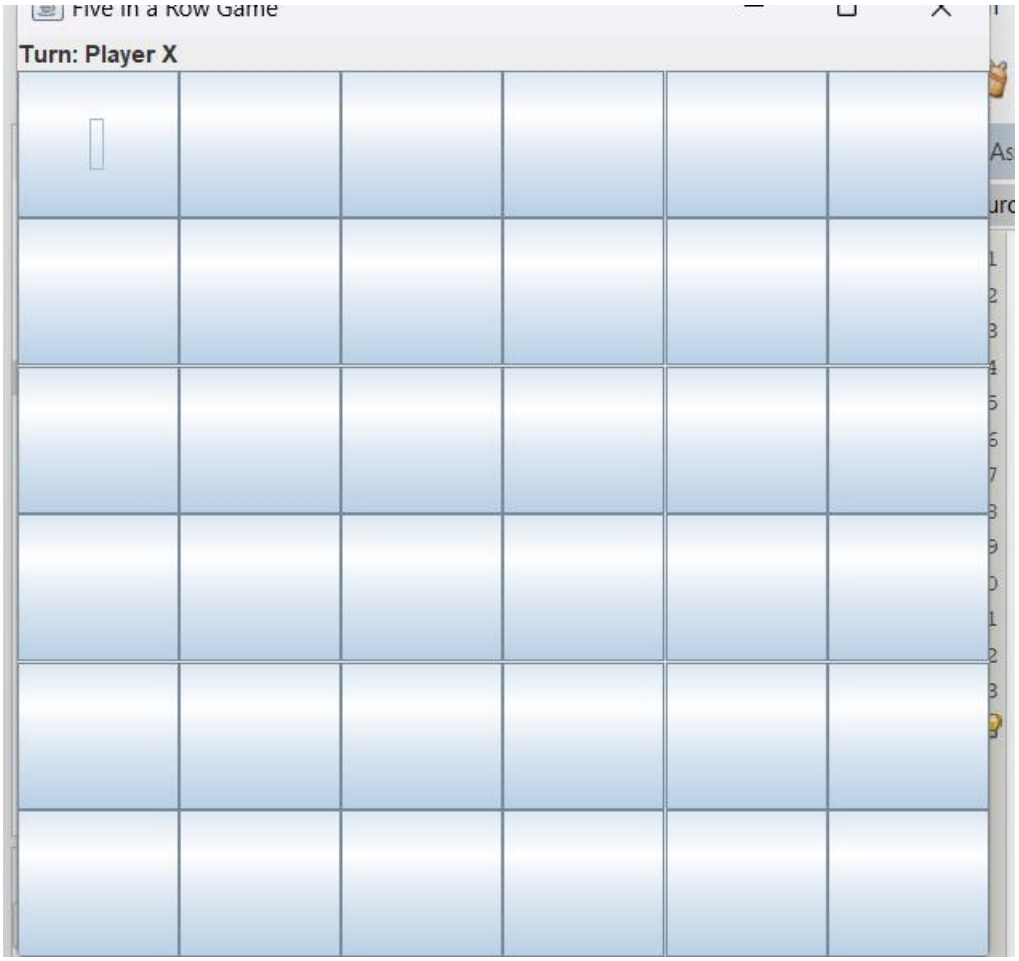


7: change the board size

If I click on the selected board size then the board size should change



8 restart the game :when player win the game then I can restart the game with new empty board



white box :

Certainly! Here’s an overview of the seven white box tests you implemented:

1. Test `checkConsecutiveSigns` - Three in a Row (Row Test):

- Objective: Verify that when there are three consecutive 'X' signs in a row, exactly one random 'X' sign is removed.

- Setup: Initialized a 6x6 game board and set three consecutive 'X' signs in the first row.

- Execution: Called `checkConsecutiveSigns('X')`.

- Expected Result: One of the three 'X' signs in that row is removed.

2. Test `checkConsecutiveSigns` - Four in a Row (Column Test):

- Objective: Ensure that when four consecutive 'O' signs appear in a column, exactly two random 'O' signs are removed.

- Setup: Initialized a 6x6 game board with four consecutive 'O' signs in the first column.

- Execution: Called `checkConsecutiveSigns('O')`.

- Expected Result: Two of the four 'O' signs in that column are removed.

3. Test `checkWin` - Five in a Row (Diagonal):

- Objective: Confirm that `checkWin` returns true when there are five consecutive 'X' signs in a diagonal.

- Setup: Initialized a 6x6 game board with five consecutive 'X' signs diagonally from (0,0) to (4,4).

- Execution: Called `checkWin('X')`.

- Expected Result: Method returns `true`, indicating a win for player 'X'.

4. Test `checkWin` - No Win Condition:

- Objective: Ensure that `checkWin` returns false when there are no five consecutive signs in any direction.

- Setup: Initialized a 6x6 game board with random placements of 'X' and 'O' signs without any five consecutive signs.

- Execution: Called `checkWin('X')`.

- Expected Result: Method returns `false`.

5. Test `checkDirection` - Horizontal Five in a Row:

- Objective: Verify that `checkDirection` correctly detects five consecutive 'O' signs horizontally.

- Setup: Initialized a 6x6 game board with five consecutive 'O' signs in the second row.

- Execution: Called `checkDirectionForFive(1, 0, 'O')`.

- Expected Result: Method returns `true`.

6. Test `removeRandomSigns` - Boundary Removal:

- Objective: Test behavior when attempting to remove more signs than exist for a player's symbol.

- Setup:Initialized a 6x6 game board with only one 'X' sign at position (2,2).

- Execution: Called `removeRandomSigns('X', 2)`.

- Expected Result: Only the single 'X' sign at (2,2) is removed without errors.

7. Test `refreshBoardDisplay` - Verify Button Updates:

- Objective: Ensure that `refreshBoardDisplay` accurately updates button text to reflect the game board's state.

- Setup: Initialized a 6x6 game board with 'X' signs in the first row, then updated some to 'O' signs.

- Execution :Called `refreshBoardDisplay`.

- Expected Result: Buttons display the updated state, showing 'O' instead of 'X' where changes occurred.