Tricky five

Documentation

Made by: Illia Takhtamyshev

Contents

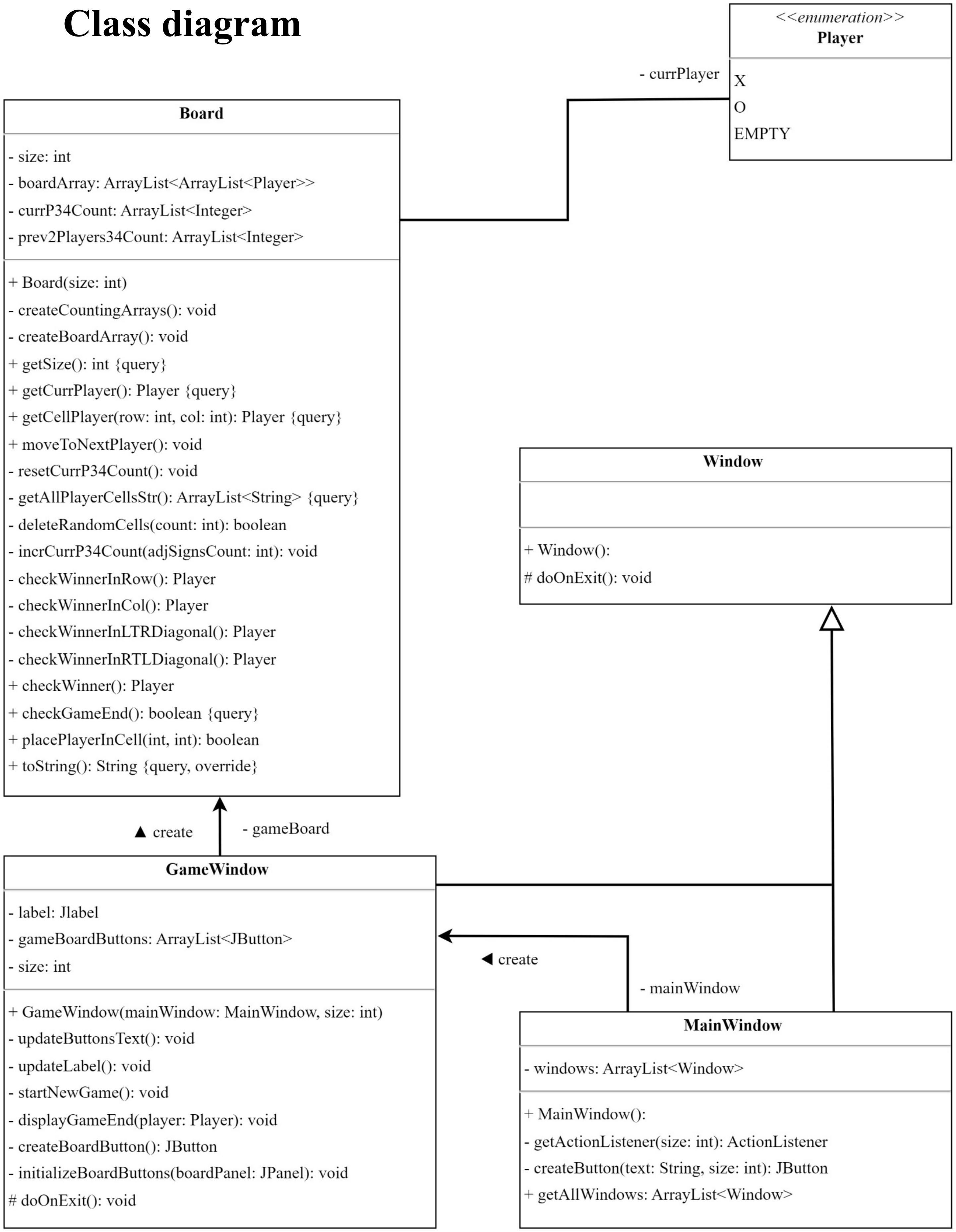
Task

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.

Class diagram


<<enumeration>>
Player
X
O
EMPTY
- currPlayer
Board
- size: int
- boardArray: ArrayList<ArrayList<Player>>
- currP34Count: ArrayList<Integer>
- prev2Players34Count: ArrayList<Integer>
+ Board(size: int)
- createCountingArrays(): void
- createBoardArray(): void
+ getSize(): int {query}
+ getCurrPlayer(): Player {query}
+ getCellPlayer(row: int, col: int): Player {query}
+ moveToNextPlayer(): void
- resetCurrP34Count(): void
- getAllPlayerCellsStr(): ArrayList<String> {query}
- deleteRandomCells(count: int): boolean
- incrCurrP34Count(adjSignsCount: int): void
- checkWinnerInRow(): Player
- checkWinnerInCol(): Player
- checkWinnerInLTRDiagonal(): Player
- checkWinnerInRTLDiagonal(): Player
+ checkWinner(): Player
+ checkGameEnd(): boolean {query}
+ placePlayerInCell(int, int): boolean
+ toString(): String {query, override}
Window
+ Window():
# doOnExit(): void
▲ create
- gameBoard
GameWindow
- label: Jlabel
- gameBoardButtons: ArrayList<JButton>
- size: int
+ GameWindow(mainWindow: MainWindow, size: int)
- updateButtonsText(): void
- updateLabel(): void
- startNewGame(): void
- displayGameEnd(player: Player): void
- createBoardButton(): JButton
- initializeBoardButtons(boardPanel: JPanel): void
# doOnExit(): void
◀ create
- mainWindow
MainWindow
- windows: ArrayList<Window>
+ MainWindow():
- getActionListener(size: int): ActionListener
- createButton(text: String, size: int): JButton
+ getAllWindows: ArrayList<Window>

Short description of methods

Board

1. **Constructor “public Board(int size)”**

- Initializes game board with the given size and sets the current player to 'X'
- Creates two counting array lists for counting adjusted signs
- Creates two-dimensional array list as a gameboard

2. **Method “private void createCountingArrays()”**

- Initializes two counting array lists. “currP34Count” counts current 3 and 4 adjusted signs for current player. “prev2Players34Count” counts 3 and 4 adjusted signs for both “X” and “O”.
- It is called by the constructor

3. **Method “private void createBoardArray()”**

- Initializes two-dimensional array list with empty cells
- It is called by the constructor

4. **Method “public int getSize()”**

- Returns size of the game board

5. **Method “public Player getCurrPlayer()”**

- Returns current player sign “X” or “O”

6. **Method “public Player getCellPlayer(int row, int col)”**

- Returns player sign which is in the specified cell of the game board

7. **Method “public void moveToNextPlayer()”**

- Switches the current player to the next one “X” or “O”

8. **Method “private void resetCurrP34Count()”**

- Resets current count of adjusted signs for current player

9. **Method “private ArrayList<String> getAllPlayerCellsStr()”**

- Returns list of strings with all coordinates of cells where are the signs of the current player

10. **Method “private boolean deleteRandomCells(int count)”**

- Deletes a random cell from those cells where are the signs of the current player based on count if new adjusted signs appeared
- If the count is equal to 3, it deletes 1 cell. If it is 4, it deletes 2 cells

11. **Method “private void incrCurrP34Count(int adjSignsCount)”**

- Increments the current count of 3-adjacent or 4-adjacent signs for the current player

12. **Method “private Player checkWinnerInRow()”**

- Checks if the current player is a winner in rows on the game board (five adjacent signs)
- In cases of three or four adjacent signs in a row it calls incrCurrP34Count

13. **Method “private Player checkWinnerInCol()”**

- Checks if the current player is a winner in columns on the game board (five adjacent signs)
- In cases of three or four adjacent signs in a column it calls incrCurrP34Count

14. **Method “private Player checkWinnerInLTRDiagonal()”**

- Checks if the current player is a winner in left to right diagonals on the game board (five adjacent signs)
- In cases of three or four adjacent signs in a left to right diagonal it calls incrCurrP34Count

15. **Method “private Player checkWinnerInRTLDiagonal()”**

- Checks if the current player is a winner in right to left diagonals on the game board (five adjacent signs)
- In cases of three or four adjacent signs in a right to left diagonal it calls incrCurrP34Count

16. **Method “public Player checkWinner()”**

- Checks if the current player is a winner in rows, columns, left to right diagonals and right to left diagonals on the game board (five adjacent signs)
- Returns the sign of the current player if he is a winner or EMPTY if there is no winner

17. Method "public boolean checkGameEnd()"

- Checks if game ended because of draw. It is case when all cells are occupied by the players signs X or O
- Returns true if ended or false if not

18. Method "public boolean placePlayerInCell(int row, int col)"

- If the cell is not occupied by anyone, it places current player sign in the cell
- Returns true if the placement was successful and false if the cell was already occupied

19. Method "public String toString()"

- Returns string representation of the board

Player

It is an enum class which does not have any methods. It represents a group of constants: X, O and EMPTY.

Window

1. Constructor "public Window()"

- Sets up a window with title, size, overridden close operation and icon.
- Gives common structure for window implementations

2. Method "protected void doOnExit()"

- Is used to define an action which will be executed on window closing
- Disposes the window

MainWindow

1. Constructor "public MainWindow()"

- Creates main window which allows user to choose one of three sizes of the gameboard
- There are text field and three clickable buttons for choosing sizes

2. Method "private ActionListener getActionListener(int size)"

- Returns action listener

- Gives an opportunity to user to create a game board with the board size

3. **Method "private JButton createButton(String text, int size)"**

- Creates a button with a given text
- Allows user to initiate games with specified board sizes

GameWindow

1. **Constructor "public GameWindow(MainWindow mainWindow, int size)"**

- Creates a new game window with the given size and associates it with the main window
- Creates label, new game button and board buttons for the player

2. **Method "private void updateButtonsText()"**

- Updates buttons text based on the game board cells (X or O)
- Is used to display signs on the board
- It is called each time after the player places a new sign

3. **Method "private void updateLabel()"**

- Updates the label which indicates whose turn is
- Displays current player sign

4. **Method "private void startNewGame()"**

- Closes current window and opens a new one
- It manages transaction between game sessions
- Restarts the game

5. **Method "private void displayGameEnd(Player player)"**

- Shows message dialog which indicates the game end
- There are two possible cases: win and draw
- Starts the new game after showing the result

6. **Method "private JButton createBoardButton(int i, int j)"**

- Creates a board button for specific cell of the gameboard
- Defines the button behavior when it is clicked

7. **Method "private void initializeBoardButtons(JPanel boardPanel)"**

- Initializes board buttons on the panel

Connections between the events and event handlers

MainWindow

Event: Button Click

- Buttons: ”button6x6”, ”button10x10”, ”button14x14”
- Event Handler: “private ActionListener getActionListener(int size)”
    - Creates new GameWindow with the gameboard which has the specified size

GameWindow

Event: Restart Button Click

- Buttons: ”restartGameButton”
- Event Handler: “private void startNewGame()”
    - Creates new GameWindow with the gameboard which has the same size
    - Disposes current GameWindow

Event: Board Button Click

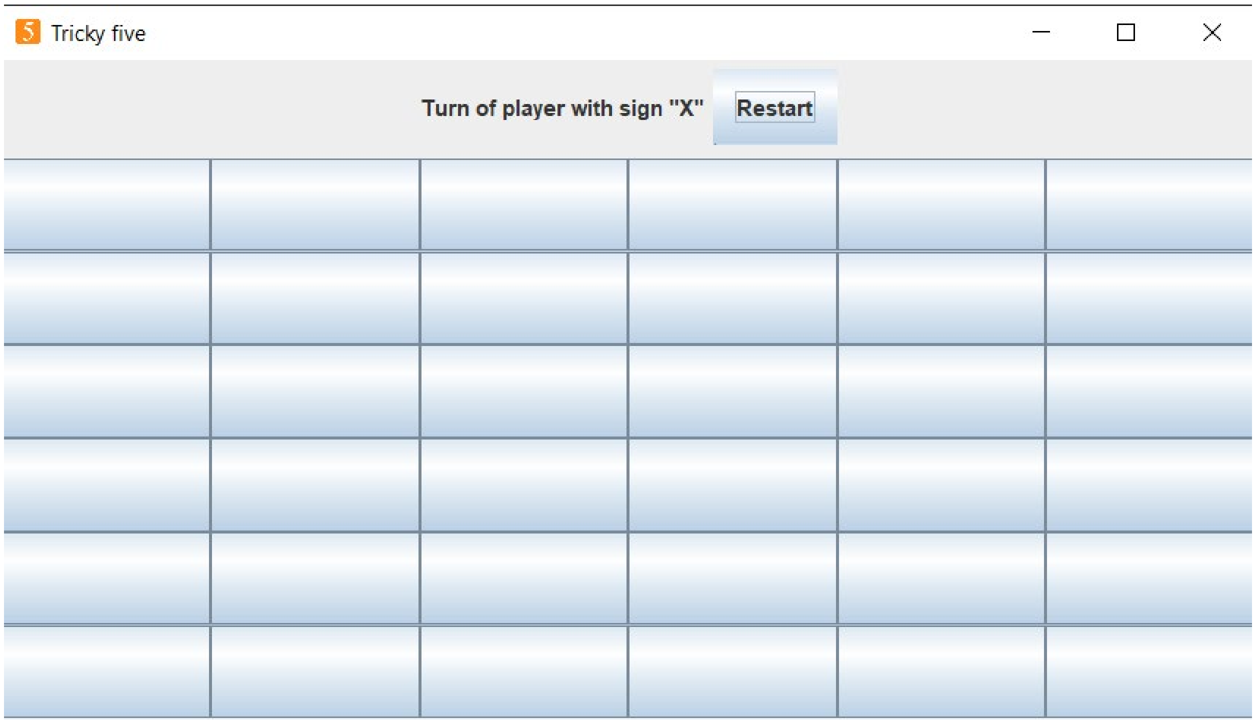
- Buttons: buttons created with
  “private JButton createBoardButton(int i, int j)"
- Event Handler: anonymous ActionListener
    - Calls placePlayerInCell(i, j) to place current player sign in a cell
    - Checks for winner using checkWinner()
    - Moves to next player X or O
    - Updates buttons text and label text
    - If there is a winner or a game end because of draw, it calls displayGameEnd(player)

Testing

1. Test clicking the size button and creating game board

Description:
When one of the three buttons is clicked, it is expected that new GameWindow will be created with the gameboard of specified size.

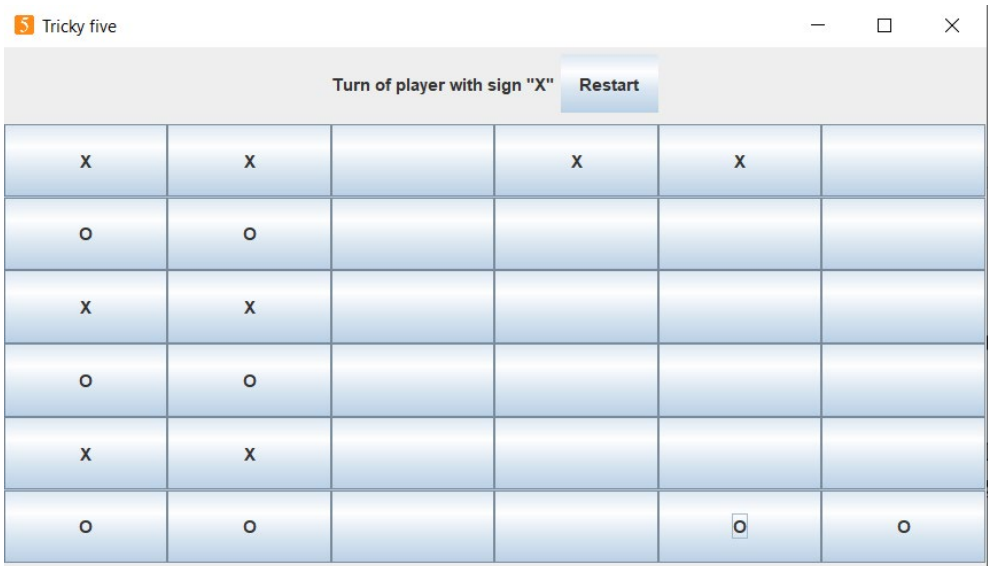
Result:
After clicking the ”button6x6”, ”button10x10”, ”button14x14” buttons, new window with the gameboard of specified size is created.



2. Test deleting random cells in case of 3 adjacent signs in a row

Description:
After creating new GameWindow place some signs one by one in two first columns for better demonstration of the random deletion and place 3 adjacent signs in a row on the right side. It is expected that one of the placed "X" signs will be deleted (including those which are in the row).

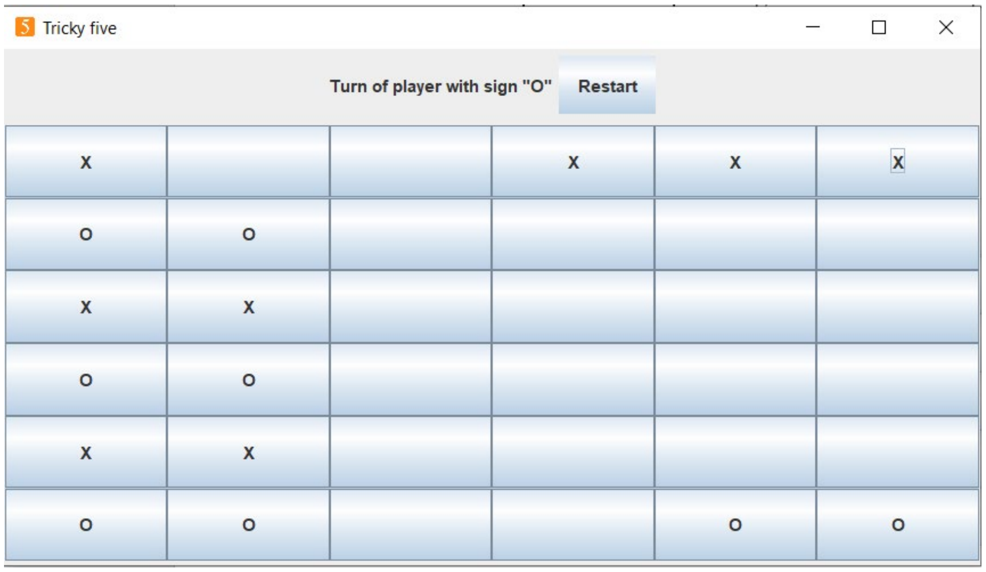
Result:
Before placing the (row 0, col 5) "X":



Turn of player with sign "X"

X	X		X	X	
O	O				
X	X				
O	O				
X	X				
O	O			O	O

After placing the (row 0, col 5) "X" (one of "X"s was deleted as expected):



Turn of player with sign "O"

X			X	X	X
O	O				
X	X				
O	O				
X	X				
O	O			O	O

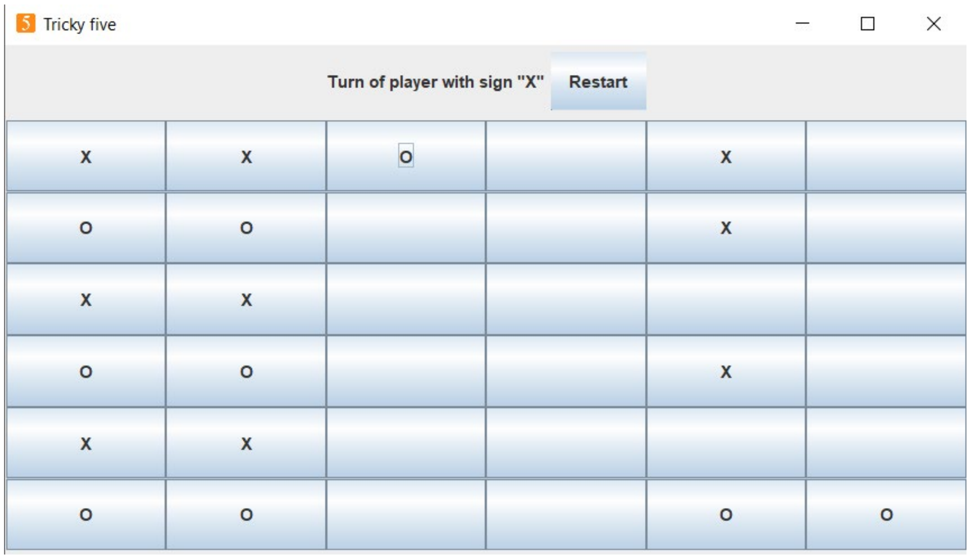
3. Test deleting random cells in case of 4 adjacent signs in a column

Description:

After creating new GameWindow place some signs one by one in two first columns for better demonstration of the random deletion and place 4 adjacent signs in a column on the right side. It is expected that two of the placed "X" signs will be deleted (including those which are in the column).

Result:

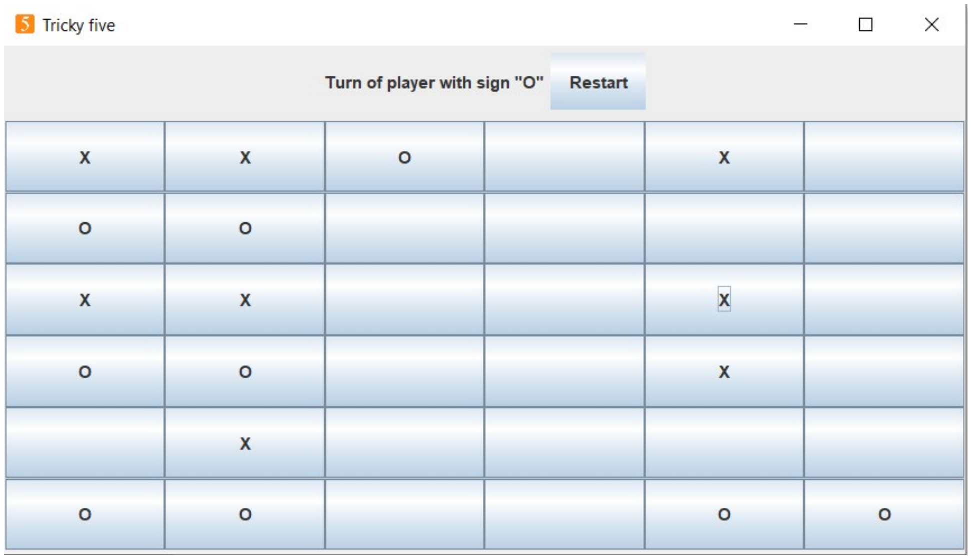
Before placing the (row 2, col 4) "X":



Turn of player with sign "X"

x	x	o		x	
o	o			x	
x	x				
o	o			x	
x	x				
o	o			o	o

After placing the (row 2, col 4) "X" (two of "X"s were deleted as expected):



Turn of player with sign "O"

x	x	o		x	
o	o				
x	x			x	
o	o			x	
	x				
o	o			o	o

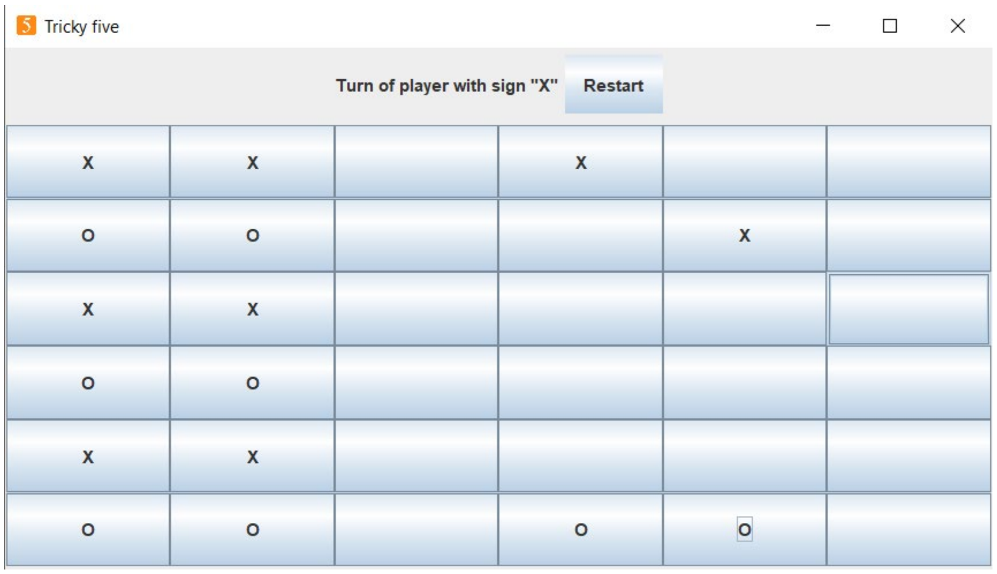
4. Test deleting random cells in case of 3 adjacent signs in a left to right diagonal

Description:

After creating new GameWindow place some signs one by one in two first columns for better demonstration of the random deletion and place 3 adjacent signs in a left to right diagonal on the right side of the board. It is expected that one of the placed "X" signs will be deleted (including those which are in the diagonal).

Result:

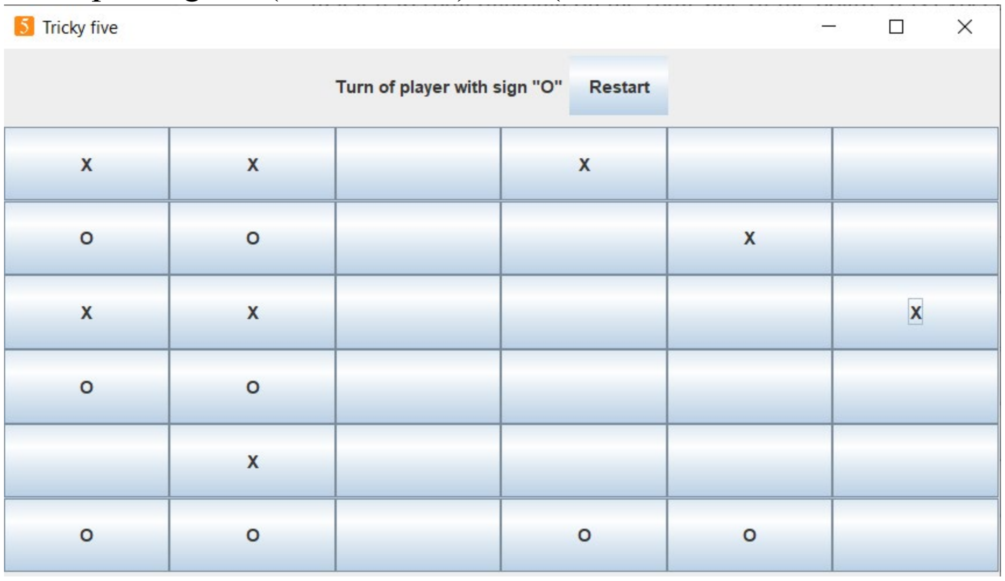
Before placing the (row 2, col 5) "X":



Turn of player with sign "X"

X	X		X		
O	O			X	
X	X				
O	O				
X	X				
O	O		O	O	

After placing the (row 2, col 5) "X" (one of "X"s was deleted as expected):



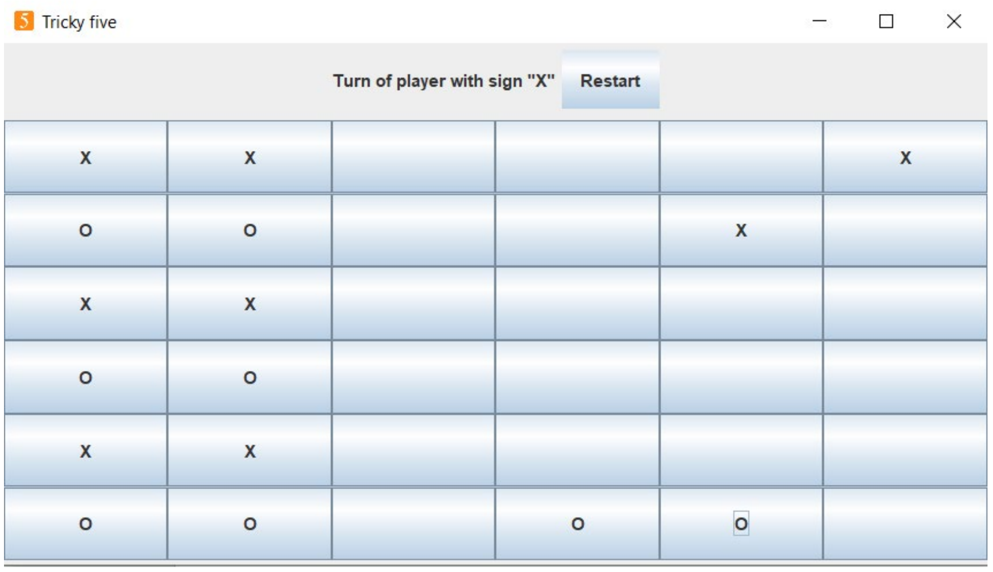
Turn of player with sign "O"

X	X		X		
O	O			X	
X	X				X
O	O				
	X				
O	O		O	O	

5. Test deleting random cells in case of 3 adjacent signs in a right to left diagonal

Description:
After creating new GameWindow place some signs one by one in two first columns for better demonstration of the random deletion and place 3 adjacent signs in a right to left diagonal on the right side of the board. It is expected that one of the placed “X” signs will be deleted (including those which are in the diagonal).

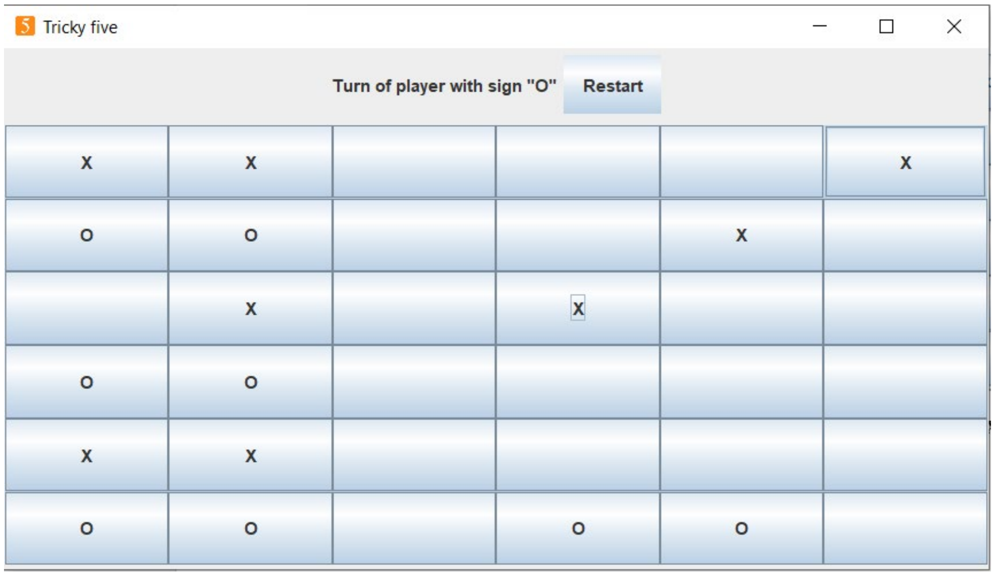
Result:
Before placing the (row 2, col 3) “X”:


Tricky five

Turn of player with sign "X" Restart

X	X				X
O	O			X	
X	X				
O	O				
X	X				
O	O		O	O	

After placing the (row 2, col 3) “X” (one of “X”s was deleted as expected):


Tricky five

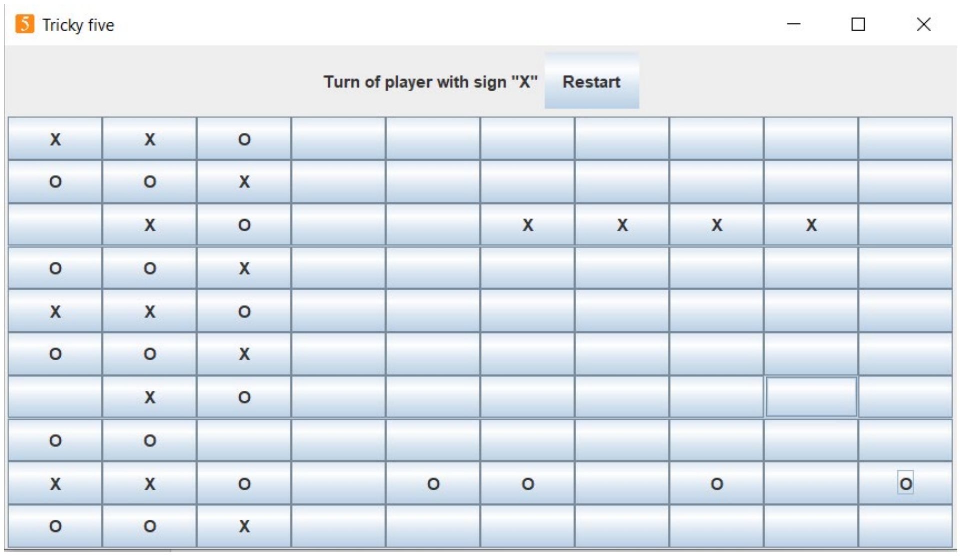
Turn of player with sign "O" Restart

X	X				X
O	O			X	
	X		X		
O	O				
X	X				
O	O		O	O	

6. Test win in case of 5 adjacent signs in a row

Description:
After creating new GameWindow place some signs one by one in three first columns for better win chances because of the random deletion and place 5 adjacent signs in a row on the right side of the board. It is expected that after placing the 5th sign, win will be counted and game end screen be showed.

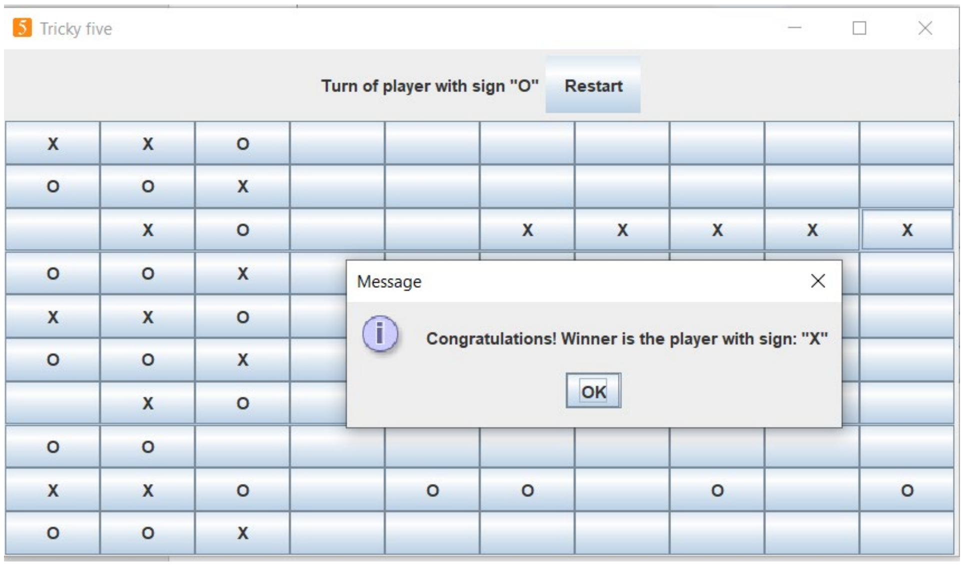
Result:
Before placing the (row 2, col 9) "X":



Turn of player with sign "X"

X	X	O							
O	O	X							
	X	O			X	X	X	X	
O	O	X							
X	X	O							
O	O	X							
	X	O							
O	O								
X	X	O		O	O		O		O
O	O	X							

After placing the (row 2, col 3) "X" (game end screen showed as expected):



Turn of player with sign "O"

X	X	O							
O	O	X							
	X	O			X	X	X	X	X
O	O	X							
X	X	O							
O	O	X							
	X	O							
O	O								
X	X	O		O	O		O		O
O	O	X							

Congratulations! Winner is the player with sign: "X"

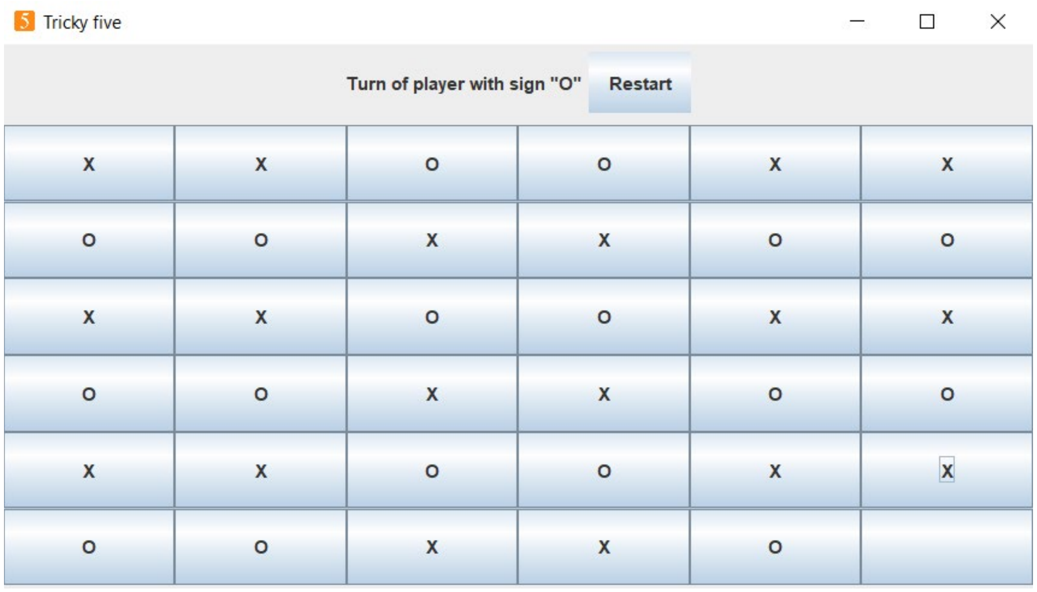
7. Test draw in case of full board

Description:

After creating new GameWindow place signs one by one in two first columns then change order for next two and continue until end for making the board full of signs without the random deletion. It is expected that after placing the last sign, draw will be counted and game end screen be showed.

Result:

Before placing the (row 5, col 5) “O”:

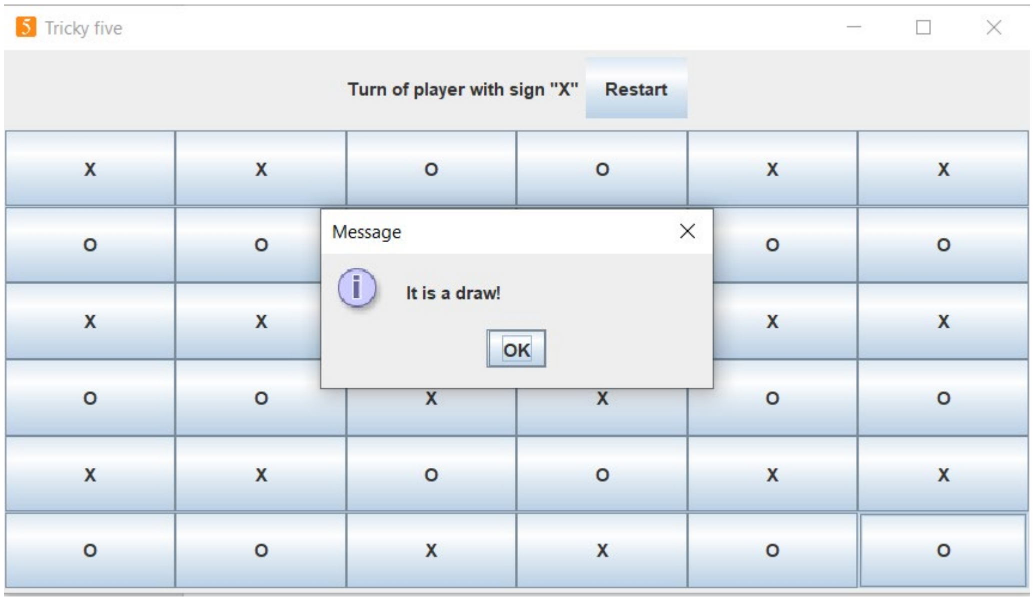


Tricky five

Turn of player with sign "O" Restart

X	X	O	O	X	X
O	O	X	X	O	O
X	X	O	O	X	X
O	O	X	X	O	O
X	X	O	O	X	X
O	O	X	X	O	

After placing the (row 5, col 5) “O” (game end screen showed as expected):



Tricky five

Turn of player with sign "X" Restart

X	X	O	O	X	X
O	O			O	O
X	X			X	X
O	O	X	X	O	O
X	X	O	O	X	X
O	O	X	X	O	O

Message

It is a draw!

OK