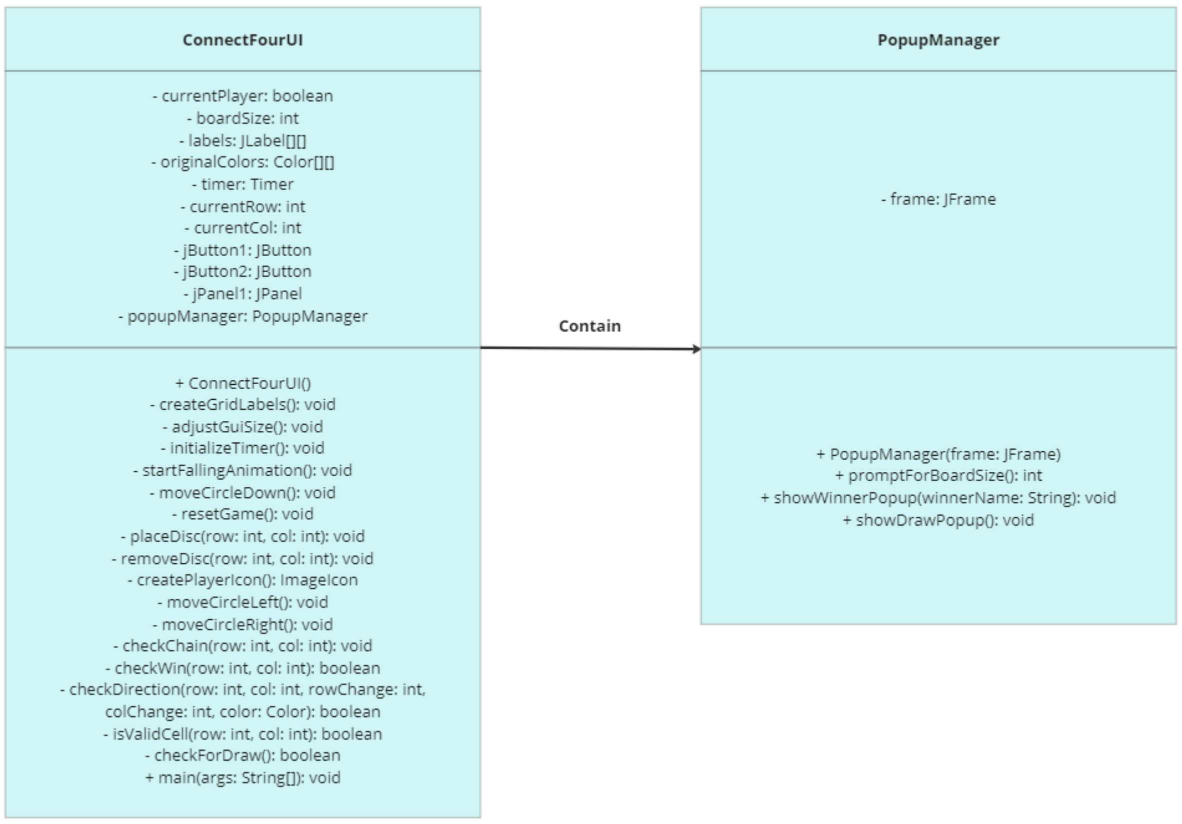
ConnectFour

Contents

Description

1. Connect Four
   Connect Four is a two-player game. The discs of the first player are marked with X, and the discs of the second player are marked with O. The players take turns dropping their disc from the top into a n-column, m-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. If the grid becomes full, the result is draw.
   Implement this game, and let the grid size be selectable (8×5, 10×6, 12×7). The game should recognize if it is ended, and it has to show the name of the winner in a message box (if the game is not ended with draw), and automatically begin a new game.

Class Diagram


ConnectFourUI
- currentPlayer: boolean
- boardSize: int
- labels: JLabel[][]
- originalColors: Color[][]
- timer: Timer
- currentRow: int
- currentCol: int
- jButton1: JButton
- jButton2: JButton
- jPanel1: JPanel
- popupManager: PopupManager
+ ConnectFourUI()
- createGridLabels(): void
- adjustGuiSize(): void
- initializeTimer(): void
- startFallingAnimation(): void
- moveCircleDown(): void
- resetGame(): void
- placeDisc(row: int, col: int): void
- removeDisc(row: int, col: int): void
- createPlayerIcon(): ImageIcon
- moveCircleLeft(): void
- moveCircleRight(): void
- checkChain(row: int, col: int): void
- checkWin(row: int, col: int): boolean
- checkDirection(row: int, col: int, rowChange: int, colChange: int, color: Color): boolean
- isValidCell(row: int, col: int): boolean
- checkForDraw(): boolean
+ main(args: String[]): void
Contain
PopupManager
- frame: JFrame
+ PopupManager(frame: JFrame)
+ promptForBoardSize(): int
+ showWinnerPopup(winnerName: String): void
+ showDrawPopup(): void

Description of each method

Directory Structure:

```
project_root/
|
├── Source Packages/
|   └── com.mycompany.assign2/
|       ├── Assign2.java
|       ├── ConnectFourUI.java
|       └── PopupManager.java
|
├── Test Packages/
|   └── <default package>/
|       └── (test files for Assign2, ConnectFourUI, and Popup
|
├── Dependencies/
|   └── JDK 20 (Default)
|
└── Project Files/
    └── pom.xml
```

ConnectFourUI Class:

- `ConnectFourUI()`: The constructor initializes the game UI, including setting up the board size, creating grid labels, adjusting the GUI size, initializing the timer, and starting the falling animation.
- `createGridLabels()`: Sets up a grid layout for the game board and initializes the labels that represent the discs on the board.
- `adjustGuiSize()`: Adjusts the size of the GUI window based on the preferred size and margin.
- `initializeTimer()`: Sets up a timer that triggers actions at regular intervals; used for animations like moving the discs down.
- `startFallingAnimation()`: Begins the animation where the current player's disc starts at the top of the column, ready to be dropped.
- `moveCircleDown()`: Moves the current player's disc down one row on the game board.
- `resetGame()`: Clears the board and resets the game state to start a new game.
- `placeDisc(int row, int col)`: Places a disc at the specified row and column, updating the UI to reflect the disc's placement.
- `removeDisc(int row, int col)`: Removes a disc from the specified row and column, updating the UI to reflect the removal.
- `createPlayerIcon()`: Creates an icon representing the current player's disc.
- `moveCircleLeft()`: Moves the current player's disc one column to the left.
- `moveCircleRight()`: Moves the current player's disc one column to the right.
- `checkChain(int row, int col)`: Checks if the latest disc placement results in a win condition.
- `checkWin(int row, int col)`: Determines if the current player has won the game by checking for a sequence of four discs.
- `checkDirection(...)`: Helper method for `checkWin` that checks for four consecutive discs in a specific direction.

- `isValidCell(int row, int col)`: Determines if the specified cell coordinates are within the bounds of the game board.
- `checkForDraw()`: Checks if the game has ended in a draw with no more moves left.
- `main(String[] args)`: The main method that starts the game by creating an instance of `ConnectFourUI` and making it visible.

PopupManager Class:

- `PopupManager(JFrame frame)`: Constructor that sets the `JFrame` context for the popups.
- `promptForBoardSize()`: Displays a dialog to the user to select the board size and returns the user's selection.
- `showWinnerPopup(String winnerName)`: Displays a popup announcing the winner of the game and asks if the players want to start a new game.
- `showDrawPopup()`: Displays a popup announcing a draw and asks if the players want to start a new game.

The connections between the events and event handlers

Event Handlers:

- **`jButton1`**: Handles the "Move disc left" button click event.
- **`jButton2`**: Handles the "Move disc right" button click event.
- **`timer`**: Handles the event that is triggered after a delay to move the disc down.

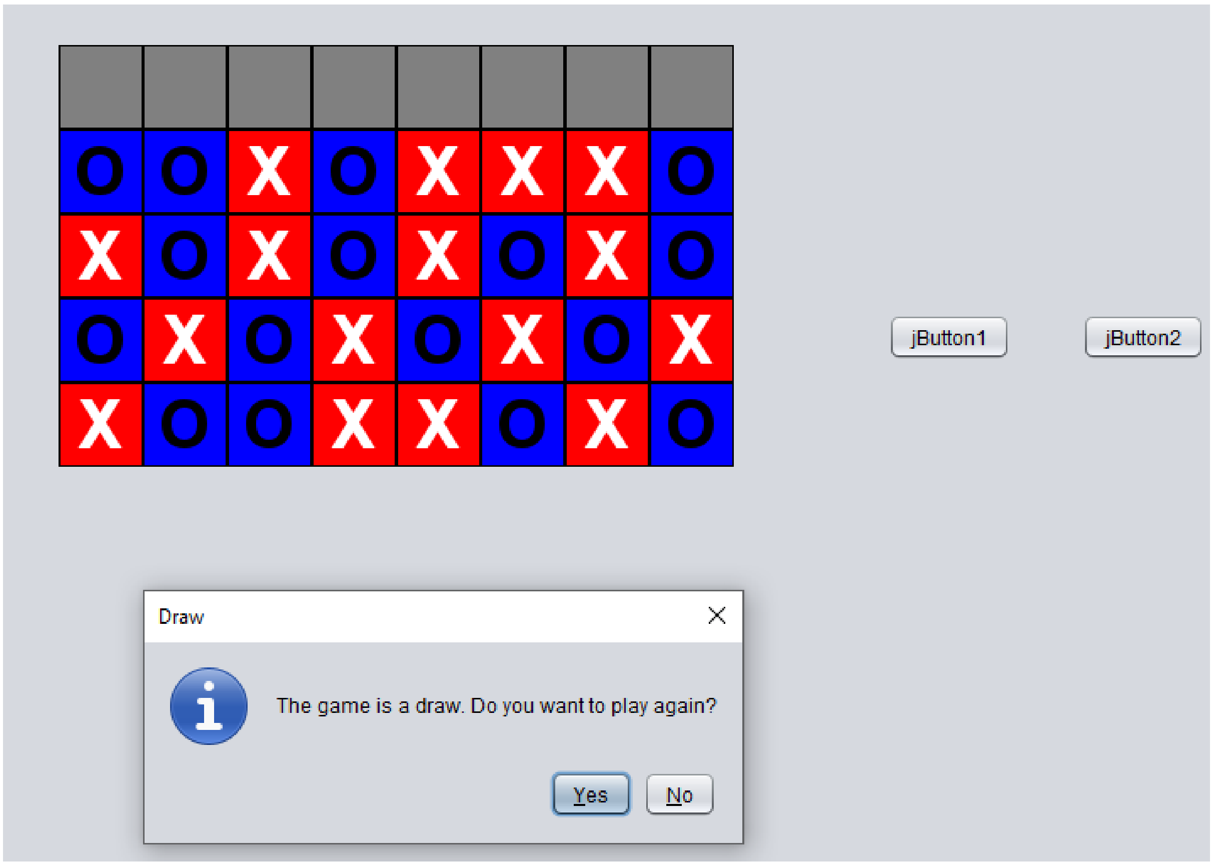
Connections Between Events and Event Handlers:

- **`jButton1`** click event connects to the **`jButton1ActionPerformed`** method.
- **`jButton2`** click event connects to the **`jButton2ActionPerformed`** method.
- **`timer`** event connects to the anonymous **`ActionListener`** class defined within the **`initializeTimer`** method, specifically to the **`actionPerformed`** method of that **`ActionListener`**.

Testing

1. **Draw Test Case:**

   - **Objective:** To verify that the game correctly identifies a draw condition when the board is filled without any player achieving the winning condition.
   - **Description:** This test will fill the board in a pattern that ensures no four discs are aligned vertically, horizontally, or diagonally. After placing the last disc, the test will assert that the game recognizes the draw state and offers the correct response, such as displaying a draw message to the user.



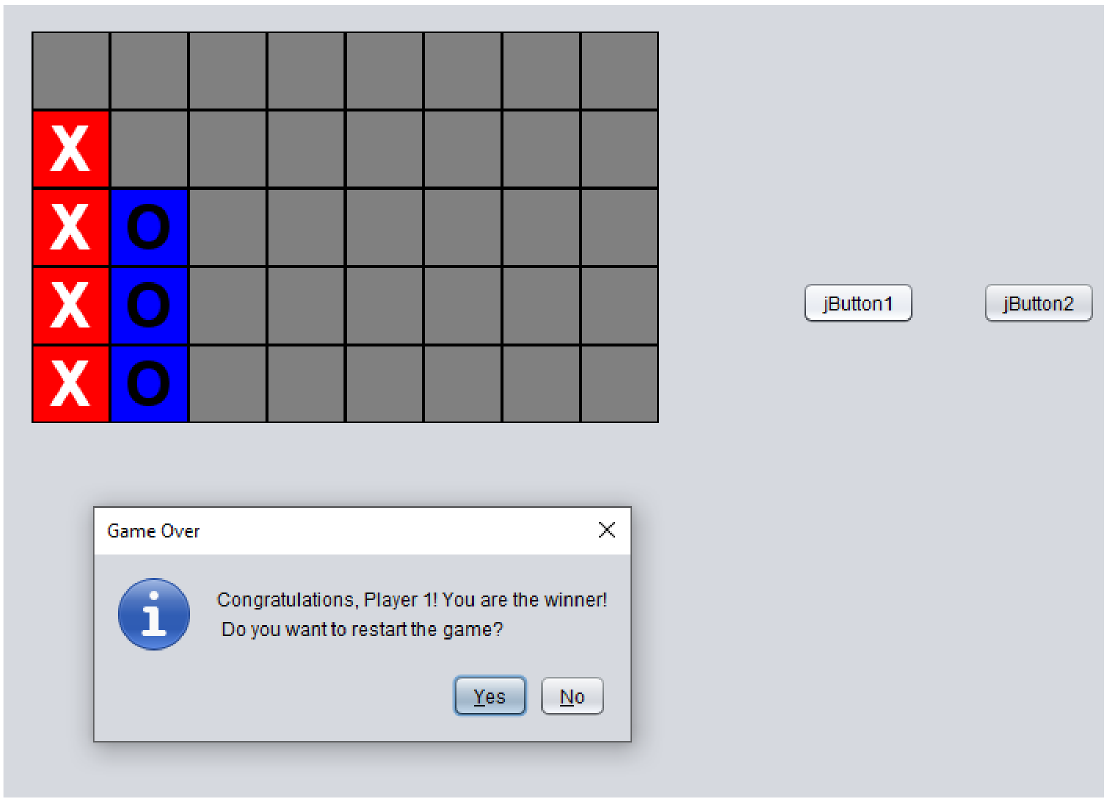
The game is a draw. Do you want to play again?

1. **Check Winner for Vertical, Horizontal, and Diagonal Test Cases:**

   - **Objective:** To ensure the game accurately determines the winner for all possible winning conditions: vertical, horizontal, and diagonal alignments of discs.

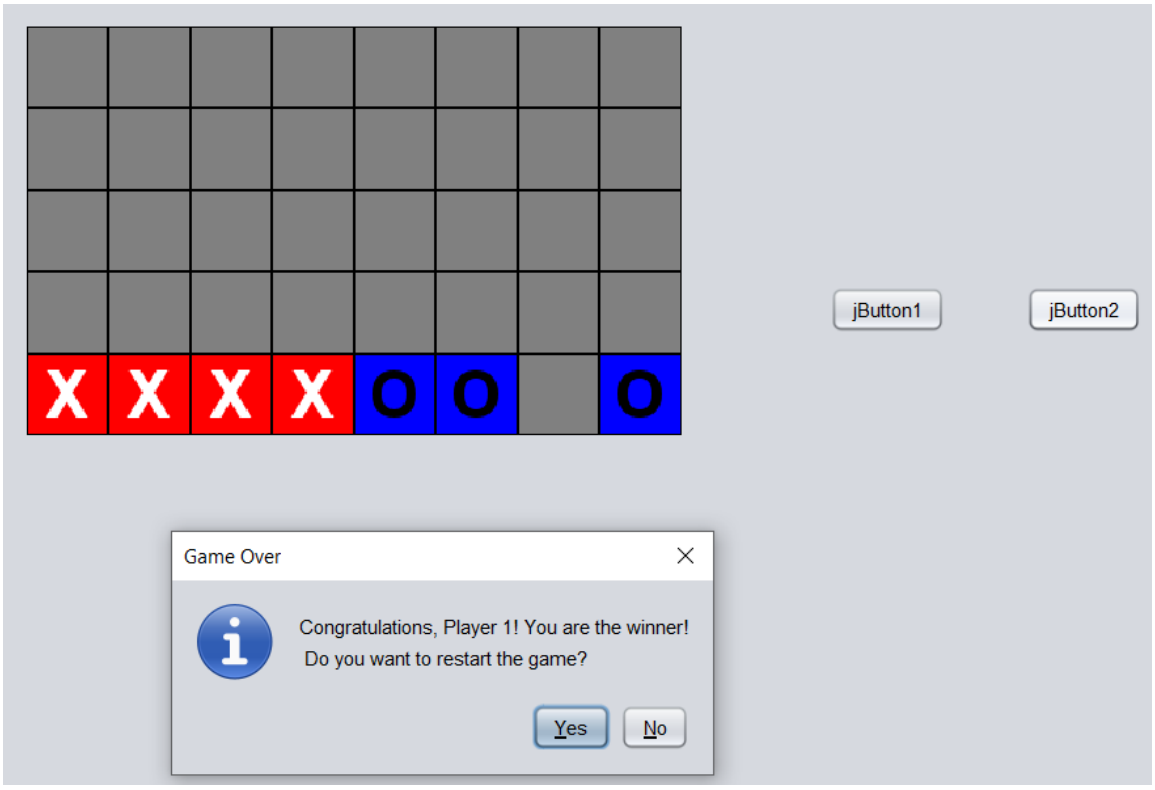
- **Description:** This set of tests will programmatically place discs to form a winning line in three directions - vertical, horizontal, and diagonal. Each test will execute moves that lead to a win state and then assert that the game identifies the winner correctly and displays the appropriate winning message.

Vertical



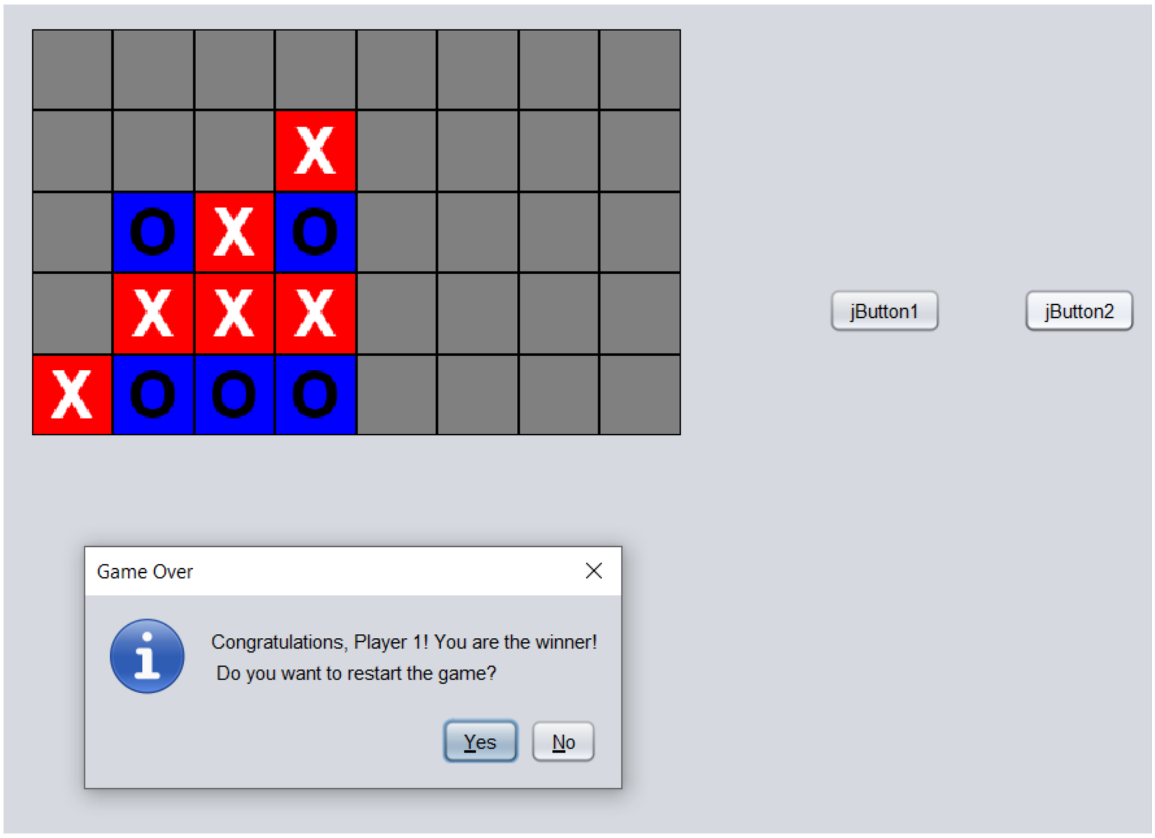
Congratulations, Player 1! You are the winner!
Do you want to restart the game?

Horizontal



Congratulations, Player 1! You are the winner!
Do you want to restart the game?

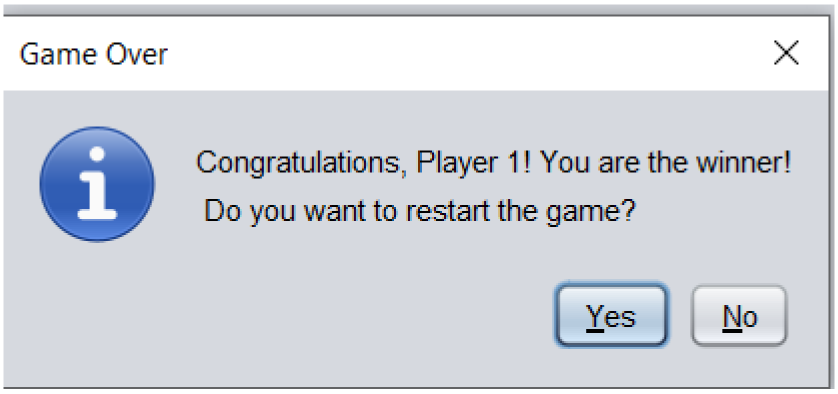
Diagonal



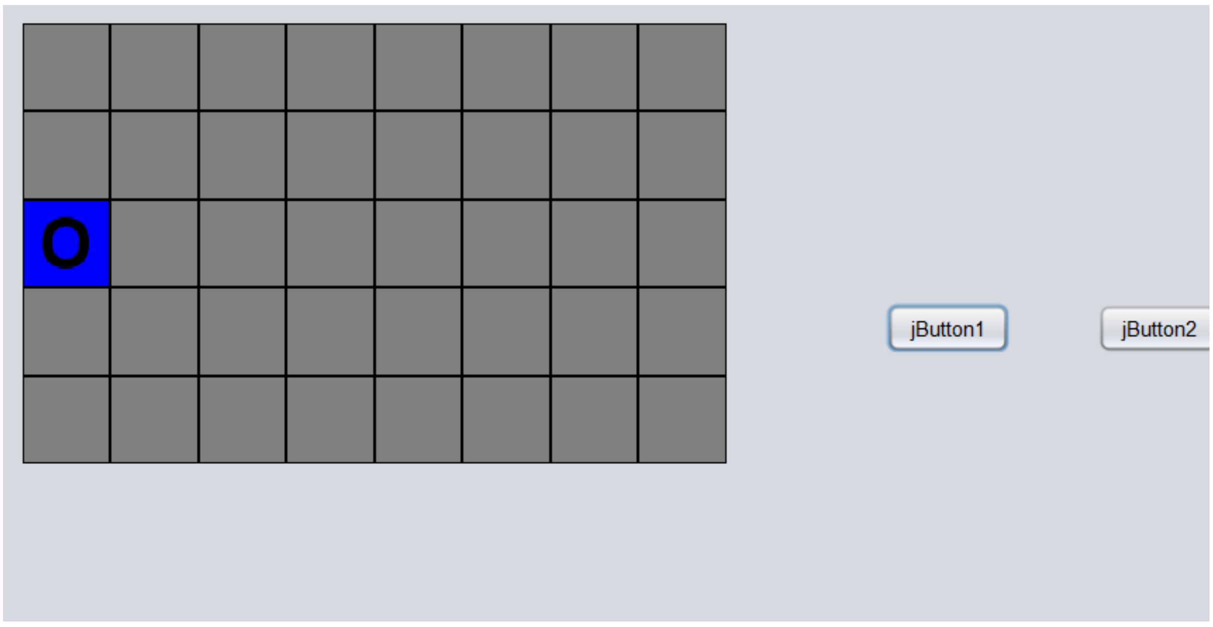
Congratulations, Player 1! You are the winner!
Do you want to restart the game?

2. **Automatic New Game Start Test Case:**

- **Objective:** To confirm that a new game automatically starts after the previous game has concluded with a win, loss, or draw.
- **Description:** This test will simulate the end of a game and then verify that the game board is reset to its initial state without user intervention, indicating the commencement of a new game. The test checks if the board is cleared, the player turns are reset, and the UI is ready to accept the first move of a new game.


Game Over

Congratulations, Player 1! You are the winner!
Do you want to restart the game?


O
jButton1
jButton2