The **XO_Game** package contains a JavaFX application that implements the classic game of Tic-Tac-Toe, also known as X-O Game. This documentation will explain the purpose and functionality of the different components and methods present in the code.

Package and Class

Package: com.example.XO_Game

Class: HelloApplication

Class Overview

The **HelloApplication** class extends the **Application** class provided by JavaFX. It serves as the main entry point for the game and contains the necessary methods and logic to create, manage, and display the game interface.

Class Variables

- **count**: An integer array used to keep track of the number of moves made in the game. Initialized with a single element **1**.

Class Methods

1. **createButtons(): Button[][]**
    - Creates and initializes a 2D array of JavaFX **Button** objects representing the game board.
    - Returns the created 2D array.
2. **setLayout(arr: Button[][]): void**
    - Configures the layout properties (size) of the buttons in the game board.
    - Takes the 2D array of buttons as a parameter.
3. **addButtonsToPane(arr: Button[][], pane: GridPane): void**
    - Adds the buttons from the 2D array to a **GridPane** container.
    - Takes the 2D array of buttons and the **GridPane** as parameters.
4. **btnClicked(btn: Button, counter: int[], arr: Button[][]): void**
    - Defines the action to be performed when a button is clicked by the player.
    - Takes a button, the counter array, and the 2D array of buttons as parameters.
    - Checks the button's text and updates it to either "X" or "O" based on the current turn.
    - Modifies the button's style based on the player's turn.

- Increments the move counter.
- Checks for a win or draw condition and displays the appropriate alert.

5. **manageButtons(arr: Button[][]): void**
    - Sets up the button click event handling for all buttons in the game board.
    - Takes the 2D array of buttons as a parameter.
6. **checkColumn(arr: Button[][]): boolean**
    - Checks if there is a winning condition in any of the columns.
    - Takes the 2D array of buttons as a parameter.
    - Returns **true** if a winning condition is found; otherwise, returns **false**.
7. **checkRow(arr: Button[][]): boolean**
    - Checks if there is a winning condition in any of the rows.
    - Takes the 2D array of buttons as a parameter.
    - Returns **true** if a winning condition is found; otherwise, returns **false**.
8. **checkDiagonal(arr: Button[][]): boolean**
    - Checks if there is a winning condition in any of the diagonals.
    - Takes the 2D array of buttons as a parameter.
    - Returns **true** if a winning condition is found; otherwise, returns **false**.
9. **checkWin(arr: Button[][]): boolean**
    - Checks if there is a winning condition in the game board.
    - Takes the 2D array of buttons as a parameter.
    - Returns **true** if a winning condition is found; otherwise, returns **false**.
10. **winAlert(arr: Button[][]): void**
    - Displays an alert dialog indicating that a player has won the game.
    - Takes the 2D array of buttons as a parameter.
    - Shows an alert box with the message "Congratulations" and an image of a cup.
11. **drawAlert(arr: Button[][]): void**
    - Displays an alert dialog indicating that the game ended in a draw.
    - Takes the 2D array of buttons as a parameter.

- Shows an alert box with the message "Try Again" and an image of "game over".

12. **reset(arr: Button[][]): void**
    - Resets the game board and move counter to their initial states.
    - Takes the 2D array of buttons as a parameter.
    - Sets all button texts to " ".
    - Resets button styles to their default values.
    - Resets the move counter to **1**.
13. **start(stage: Stage): void**
    - Overrides the **start** method from the **Application** class.
    - Configures and displays the game window.
    - Takes a **Stage** object as a parameter.
14. **main(args: String[]): void**
    - The main entry point for the application.
    - Launches the JavaFX application.

Class Execution Flow

When the **main** method is called, it launches the JavaFX application by calling the **launch** method. This, in turn, calls the **start** method, which sets up and displays the game window. The game window contains a **GridPane** that holds the buttons representing the game board.

The **start** method creates the necessary buttons and sets up their layout and event handling using the helper methods defined in the class. It also configures the window's size, title, icon, and visibility.

When a button is clicked, the corresponding **btnClicked** method is invoked, which updates the button's text, style, and increments the move counter. It then checks for a win or draw condition using the appropriate methods. If a win or draw is detected, the corresponding alert dialog is displayed, and the game board is reset.

The game continues until a win or draw occurs, at which point the player can start a new game by closing the alert dialog or by closing and reopening the application window.

UML Diagram:

HelloApplication
- **count**: int[]
+ **createButtons**(): Button[][] + **setLayout**(arr: Button[][]): void

+ **addButtonsToPane**(arr: Button[][], pane: GridPane): void
+ **btnClicked**(btn: Button, counter: int[], arr: Button[][]): void
+ **manageButtons**(arr: Button[][]): void
+ **checkColumn**(arr: Button[][]): boolean
+ **checkRow**(arr: Button[][]): boolean
+ **checkDiagonal**(arr: Button[][]): boolean
+ **checkWin**(arr: Button[][]): boolean
+ **winAlert**(arr: Button[][]): void
+ **drawAlert**(arr: Button[][]): void
+ **reset**(arr: Button[][]): void
+ **start**(stage: Stage): void
+ **main**(args: String[]): void