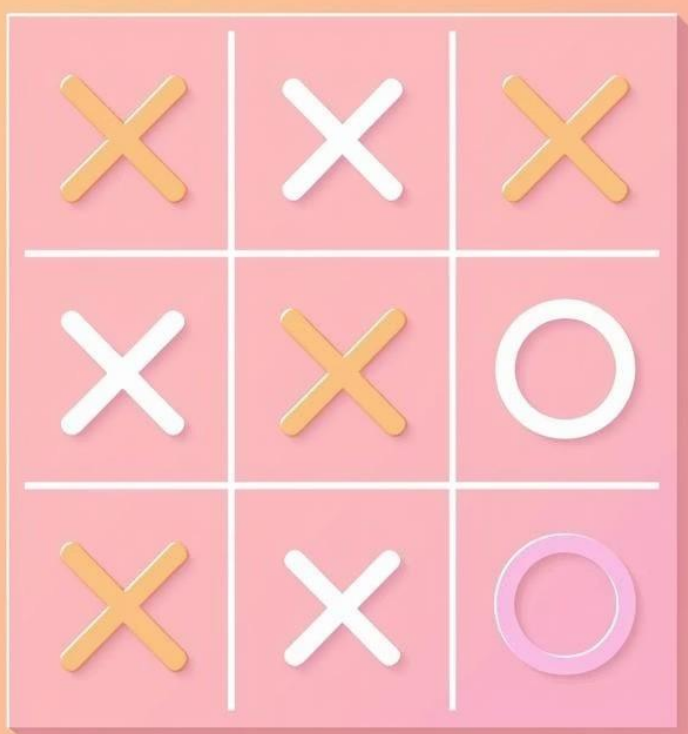


Building a Tic-Tac-Toe Game in Java Swing

This presentation explores the development of a Tic-Tac-Toe game using Java Swing, highlighting its core components and functionalities. We'll delve into the code structure, user interface design, and game logic that bring this classic game to life.

Chapter 1

Project Structure and Initialization

The TicTacToeGUI.java file serves as the main entry point for the application. It extends JFrame and implements ActionListener to handle user interactions. The constructor initializes the game window, sets up player names, and configures the game, settings, and score panels.

Java Swing

Utilizes Swing for GUI components.

Game Logic

Manages game state and player turns.

User Interface

Interactive and customizable UI elements.

Player Setup and Game Modes

The setupPlayers() method prompts the user to choose between "Single Player" and "Two Players" modes. In single-player mode, "Computer" is automatically assigned as Player O. Player names are customizable, with default values provided if no input is given.

1

Choose Game Mode

Option for single-player or two-player.

2

Player Naming

Customizable names for Player X and Player O.

3

AI Opponent

Computer plays as Player O in single-player mode.

Chapter 2

Configuring Game Settings

The initSettingsPanel() method creates a panel for game customization. Users can adjust the grid size (3x3, 4x4, 5x5), font size for game buttons, and the background color of the game board. These settings dynamically update the game panel.

- **Grid Size:** Adjustable from 3x3 to 5x5.
- **Font Size:** Options for button text size (40, 50, 60, 70).
- **Background Color:** A color chooser allows selection of the board's background.

Dynamic Game Board Initialization

The initGamePanel() method is responsible for creating and updating the Tic-Tac-Toe grid. It dynamically generates buttons based on the selected grid size, ensuring a responsive and customizable game experience.

01

Remove Existing Panel

Clears the previous game grid if it exists.

02

Calculate Total Cells

Determines the number of buttons needed based on gridSize .

03

Create Buttons

Initializes each button with default text, font, and background color.

04

Add Action Listeners

Attaches an action listener to each button for user interaction.

05

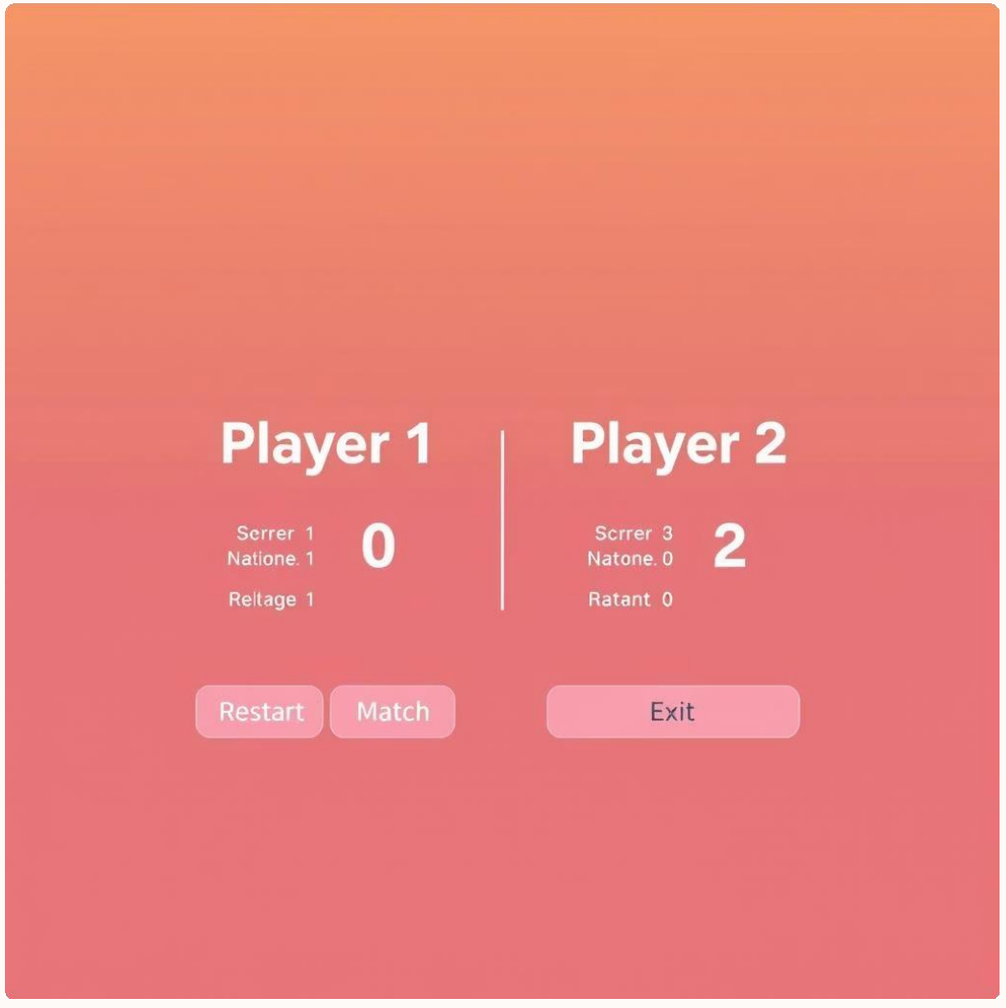
Revalidate and Repaint

Ensures the UI updates correctly after changes.

Chapter 3

Score Tracking and Game Controls

The initScorePanel() method sets up the display for player scores and game control buttons. It includes a score label that dynamically updates, along with buttons to reset the current game, restart the entire match, or exit the application.



- **Score Label:** Displays scores for Player X, Player O, and draws.
- **Reset Game Button:** Clears the board for a new round without affecting scores.
- **Restart Match Button:** Resets all scores and starts a new match.
- **Exit Button:** Terminates the application.

Updating UI Elements

The updateButtonFonts() and updateButtonColors() methods ensure that changes made in the settings panel are immediately reflected on the game board. The updateScoreLabel() method keeps the score display current after each game outcome.

Font Updates

Applies new font sizes to all game buttons.

Color Changes

Updates the background color of empty cells.

Score Refresh

Reflects current scores and draws on the label.

Chapter 4

Game Logic and AI

The core game logic resides in methods like checkGameStatus() and checkWin() , which determine win conditions and draws. In single-player mode, the makeAIMove() method provides a basic AI opponent that selects a random available cell.

Win Condition Check

Identifies winning rows, columns, and diagonals.

Draw Detection

Determines if the game ends in a draw.

Basic AI

Randomly selects an empty cell for the computer's move.

User Interaction and Game Flow

The actionPerformed() method handles button clicks, placing the current player's symbol on the board. It manages player turns, triggers AI moves in single-player mode, and updates the game status after each move.

1

Button Click

Registers user input on the game board.

2

Place Symbol

Marks the clicked cell with 'X' or 'O'.

3

Check Status

Evaluates for win, loss, or draw conditions.

4

Next Turn

Switches player or triggers AI move.

Conclusion and Future Enhancements

This Tic-Tac-Toe game provides a solid foundation for a GUI application in Java Swing. Future enhancements could include more sophisticated AI, different game modes (e.g., online multiplayer), and advanced UI themes.

Key Takeaways

- Modular design with separate panels.
- Dynamic UI updates based on settings.
- Basic game logic and AI implementation.

Next Steps

- Implement advanced AI algorithms.
- Add network multiplayer functionality.
- Introduce custom themes and animations.