Tic Tac Toe Documentation

Made by:-

Saikat Kar {AD1-13 | 231191101154}

Aditya Kumar Singh {AD1-15 | 231191101007}

Thiramdasu Yaswanth Krishna {AD1 | 49|231191101190}

Syed Naveed {AD1-50 | 231191101187}

Key Concepts Used in the Program

1. Object-Oriented Programming (OOP) Concepts

- Encapsulation:

- The class `TicTacToe` encapsulates the game's state and behavior. The data members like `frame`, `buttons`, `currentPlayer`, and `random` are private, ensuring that they cannot be accessed directly from outside the class.

- The methods `initializeButtons()`, `makeComputerMove()`, `getNextPlayer()`, `hasWon()`, and `isBoardFull()` operate on these data members to manipulate the game state.

- Abstraction:

- The program hides the complex implementation details of the Tic Tac Toe game behind simple methods. For example, the `makeComputerMove()` method abstracts the logic of making a computer move.

- Inheritance:

- This concept is demonstrated through the `ButtonListener` class which extends the `ActionListener` interface to define custom behavior for button clicks.

- Polymorphism:

- The use of the `ActionListener` interface allows for polymorphism. The `actionPerformed` method is overridden in the `ButtonListener` class to define specific behavior when a button is clicked.

2. Loops

- For Loops:

- Used in the `initializeButtons()` method to initialize the 3x3 grid of buttons.

- Used in the `hasWon()` method to check if a player has won by iterating over rows and columns.

3. Conditions

- If-Else Statements:

- Used in the `actionPerformed()` method to check the state of the clicked button and determine the game's outcome.

- Used in the `makeComputerMove()` method to determine the computer's next move and check for a win or a draw.

- Used in the `hasWon()` method to check the winning conditions for rows, columns, and diagonals.

- Used in the `isBoardFull()` method to check if the board is full.

Packages and Methods

1. **`javax.swing` Package**

- `JFrame`

- `setDefaultCloseOperation(int operation)`: Sets the operation that will happen by default when the user initiates a "close" on this frame.
- `setLayout(LayoutManager manager)`: Sets the layout manager for this container.
- `add(Component comp)`: Adds the specified component to the end of this container.
- `setSize(int width, int height)`: Sets the size of the frame.
- `setVisible(boolean b)`: Shows or hides the frame depending on the value of parameter `b`.
- `dispose()`: Releases all of the native screen resources used by this window.

- `JButton`

- `setText(String text)`: Sets the button's text.
- `setFont(Font font)`: Sets the button's font.

  - `addActionListener(ActionListener l)`: Adds the specified action listener to receive action events from this button.
  - `getText()`: Returns the button's text.

- `JOptionPane`

  - `showMessageDialog(Component parentComponent, Object message)`: Shows a message dialog with the specified message.

2. `java.awt` Package

- `GridLayout`

  - Constructor `GridLayout(int rows, int cols)`: Creates a grid layout with the specified number of rows and columns.

- `Font`

  - Constructor `Font(String name, int style, int size)`: Creates a new font with the specified name, style, and size.

3. `java.awt.event` Package

- `ActionEvent`

- `getSource()`: Returns the object on which the event initially occurred.

- `ActionListener`

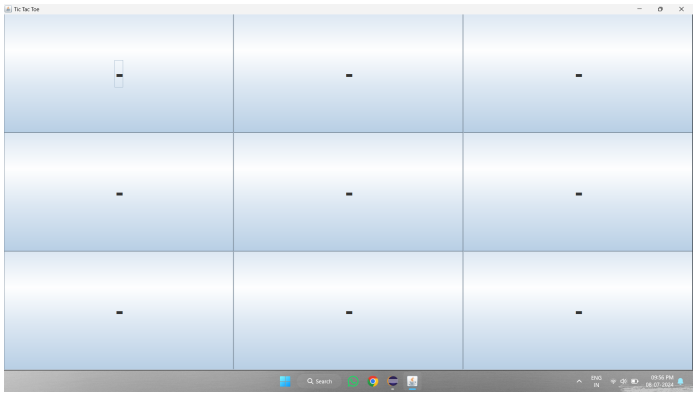
- `actionPerformed(ActionEvent e)`: Invoked when an action occurs.

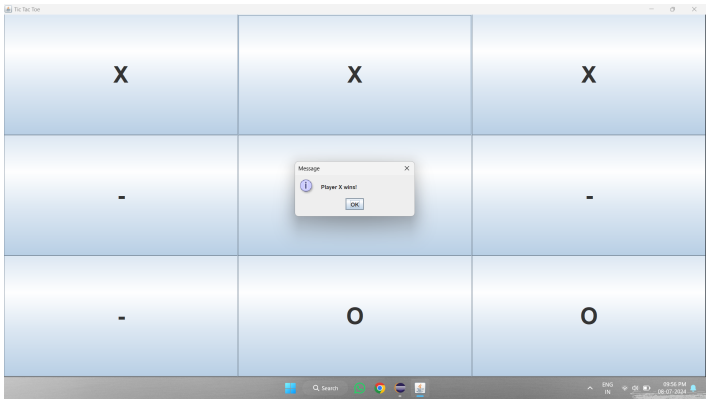
4. `java.util` Package

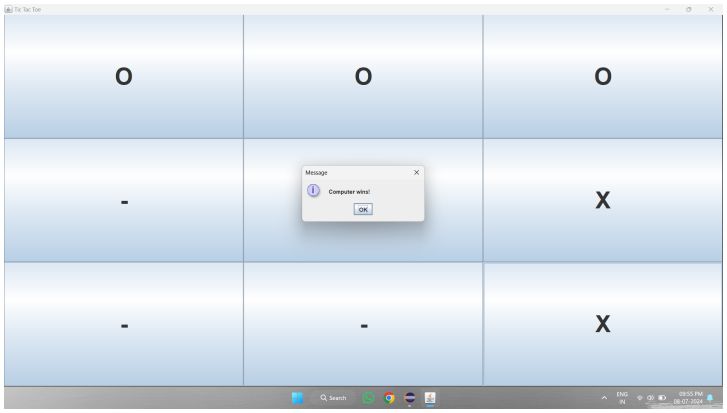
- `Random`

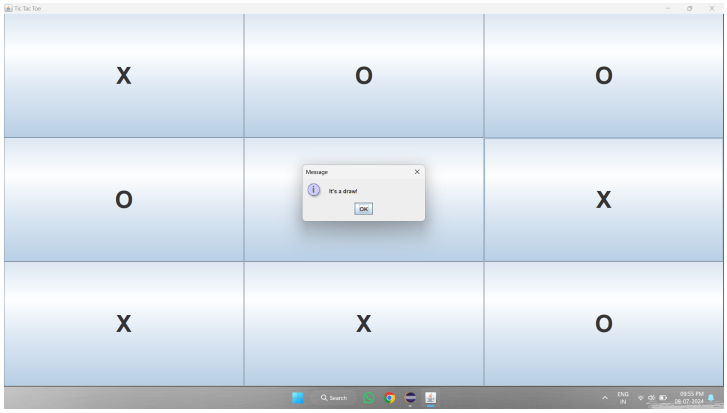
- `nextInt(int bound)`: Returns a random integer between 0 (inclusive) and the specified value (exclusive).

Output


Tic Tac Toe
-
-
-
-
-
-
-
-
-


Tic Tac Toe
X
X
X
-
Message
Player X wins!
OK
-
-
O
O


Tic Tac Toe
O
O
O
-
Message
Computer wins!
OK
X
-
-
X


Tic Tac Toe
X
O
O
O
Message
It's a draw!
OK
X
X
X
O