Tic-Tac-Toe Console App Documentation

Overview

This is a simple console-based Tic-Tac-Toe game implemented in Dart. The game allows two players to play alternately, placing their markers (X or O) on a 3x3 grid. The goal is to either get three of your markers in a row, column, or diagonal or force a draw if no one wins.

The game features input validation, clear error messages, and the ability to restart after the game ends.

Features

1. **Game Board:**
    - The board is a 3x3 grid.
    - Displays the current state of the board after each move.
2. **Player Input:**
    - Players alternate turns, entering a number between 1 and 9 to place their marker.
    - Input is validated to ensure:
        - It is within the range 1-9.
        - The chosen cell is not already occupied.
3. **Winning Conditions:**
    - The game checks for a win after each move. A win occurs if a player has three consecutive markers in:
        - Any row.
        - Any column.
        - Either diagonal.
4. **Draw Condition:**
    - The game is declared a draw if the board is full and no player has achieved a winning condition.
5. **Restart Option:**
    - After the game ends, players can choose to restart or exit the game.
6. **Code Structure:**

- Modularized functions for readability and reusability:
  - printBoard: Displays the game board.
  - checkWinner: Checks for winning conditions.
  - isDraw: Checks if the game ends in a draw.
  - playGame: Handles the core game loop.
- Main game logic is wrapped in the playGame function, allowing for easy restarts.

How to Run

1. Install Dart from dart.dev.
2. Open a terminal and navigate to the folder containing the Dart file.
3. Run the application using:

```
dart run
```

How to Play

1. Two players take turns to play. Player 1 is X, and Player 2 is O.
2. On each turn:
   - The board is displayed.

The current player is prompted to input a number (1-9) corresponding to the desired cell:

```
 1  |  2  |  3
----|-----|----
 4  |  5  |  6
----|-----|----
 7  |  8  |  9
```

   - For example, entering **5** places the marker in the center cell.
3. If the input is invalid (out of range or cell already occupied), an error message is displayed, and the player is prompted to try again.
4. The game checks for a winner or draw after each move:
   - If a winner is found, the game ends and declares the winner.
   - If the board is full and no one wins, the game declares a draw.

5. After the game ends, players can restart or exit.

Code Structure

1. Main Function

Handles the overall flow of the game:

- Loops to allow restarting the game.
- Calls the playGame function to start the game.

2. playGame Function

Encapsulates the core game logic:

- Initializes the game board.
- Alternates player turns.
- Validates inputs.
- Checks for winners or draws after each move.

3. printBoard Function

Displays the current state of the game board in a readable format.

4. checkWinner Function

Checks if the current player has achieved a winning condition:

- Rows, columns, or diagonals.

5. isDraw Function

Checks if the game is a draw:

- Returns true if all cells are filled and no player has won.

Code Example

Here's a snippet of how the board is displayed:

```
void printBoard(List<List<String>> board) {
  for (int i = 0; i < 3; i++) {
    print(' ${board[i][0]} | ${board[i][1]} | ${board[i][2]} ');
    if (i < 2) print('---|---|---');
  }
}
```

Example output:

```
X   | O  |
---|---|---
 O  | X  |
---|---|---
    |    | X
```

Extensions

- **Bonus AI Opponent:** You could enhance the game by implementing a simple AI opponent to play against the human player.
- **Player Marker Selection:** Allow players to choose their markers (X or O) at the start of the game.
- **Graphical UI:** Use Flutter to create a graphical version of the game.

Conclusion

This Tic-Tac-Toe console application is a beginner-friendly project that demonstrates the fundamentals of Dart programming, including:

- Lists and loops.
- User input handling.
- Input validation.
- Game logic implementation.