Programming Technology 1

Adnan Adnan (NBDK4L)

Connecting Four – Task (1)

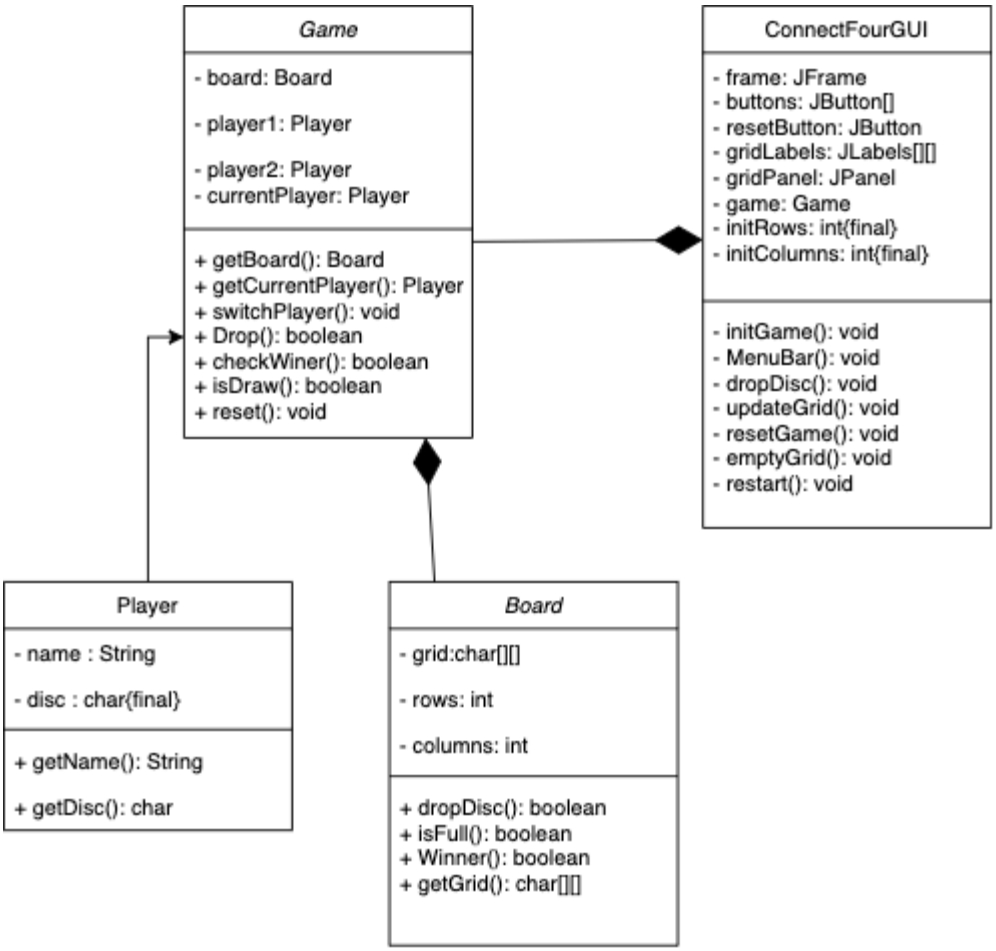
Description of the Task:

Connect Four is a two-player game where players aim to create a line of four consecutive discs (horizontally, vertically, or diagonally) on a vertically suspended grid. Players take turns dropping discs (Player 1 uses 'X' and Player 2 uses 'O') into columns on the grid, where each disc falls to the lowest available cell in the chosen column. The game can end in a win (when a player forms a line of four discs) or a draw (when the grid is full with no winning line). The grid size can be customized to 8x5, 10x6, or 12x7. After each game, the winner is displayed in a dialog box, and a new game begins.

Class Diagram:

The following diagram shows the relationships between the main classes of the game. This UML diagram illustrates how the classes are organized.

- **Player**: Represents each player in the game.
- **Board**: Manages the game grid and the placement of discs.
- **Game**: Handles game logic, player turns, and win/draw conditions.
- **ConnectFourGUI**: Provides a graphical interface, allowing players to interact with the game.



Method Descriptions:

Player Class

Represents a player in the game with attributes for the player's name and disc symbol.

- **getName()**: Returns the name of the player.
- **getDisc()**: Returns the disc symbol of the player.

Board Class

Defines the game grid, handles disc placement, and checks for win conditions.

- **dropDisc()**: Attempts to place the disc in the lowest available row within the specified column.
- **isFull()**: Checks if the board is completely filled.
- **Winner()**: Checks if the specified disc has a four-in-a-row alignment (horizontal, vertical, or diagonal).
- **getGrid()**: Returns the current state of the grid as a 2D character array.

Game Class

Manages players, the board, and game flow.

- **getBoard()**: Returns the Board object.
- **getCurrentPlayer()**: Returns the current player.
- **switchPlayer()**: Switches the current player to the other player.
- **Drop()**: Attempts to drop the current player's disc in the specified column and returns success status.
- **checkWinner()**: Checks if the current player has won.
- **isDraw()**: Checks if the board is full, resulting in a draw.
- **reset()**: Resets the board and player states for a new game.

ConnectFourGUI Class

Implements the graphical interface, allowing player interaction with the game.

- **initGame(int rows, int columns)**: Sets up the frame, buttons, and grid based on specified board size.
- **MenuBar()**: Configures the menu bar with options for selecting board size and exiting.
- **dropDisc()**: Handles the drop action, updates the grid, checks for a win or draw, and switches players.
- **updateGrid()**: Updates the GUI grid to reflect the game board's current state.
- **resetGame()**: Resets the game board and GUI.
- **emptyGrid()**: Clears all discs from the GUI grid.
- **restart()**: Restarts the game with a new board size.

Event and Event handlers:

Drop Button Event (dropDisc()):

- Each column has a "Drop" button associated with it, and clicking this button triggers the dropDisc() method.
- The method attempts to drop a disc in the specified column for the current player by calling game.Drop().
- If the column has space, it updates the grid (updateGrid()), checks if there's a winner (game.checkWinner()), or checks if the game is a draw (game.isDraw()).
- If either a winner or a draw is found, a message is displayed, and the game is reset (resetGame()); otherwise, the player switches.

Reset Button Event (resetGame()):

- This button triggers the resetGame() method, which resets the game state by calling game.reset() and clears the GUI grid (emptyGrid()).
- The game resets without changing the board dimensions, providing a fresh game on the current grid size.

Menu Options for Board Size (MenuBar):

- Three menu items in the "Game" menu allow selecting different board sizes (8x5, 10x6, and 12x7).
- Each menu item triggers the restart() method with specific dimensions.
- This method closes the current game window () and reinitializes the game (initGame()) with the specified board size.

Exit Menu Item (MenuBar):

- The "Exit" menu item has an action listener that calls System.exit(0) to close the game window and exit the program.

Connections Between Events and Handlers

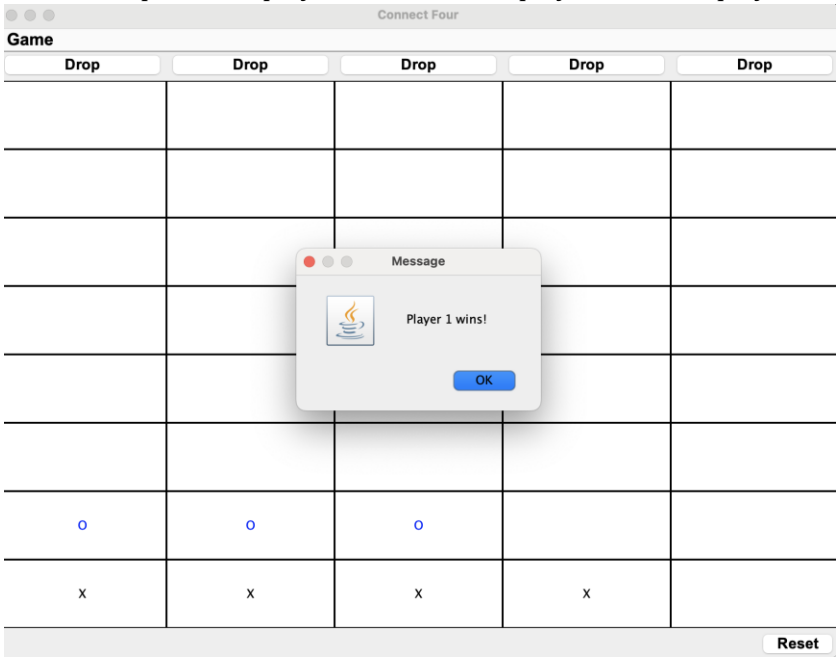
- **GUI Buttons and Menu Items**:
    - Each "Drop" button directly calls the dropDisc(i) method, connecting the button click to dropping a disc in the corresponding column.
    - The "Reset" button is linked to the resetGame() method to restart the game with the same dimensions.
    - The board size menu items trigger restart() with specific dimensions, allowing dynamic changes to board size.

- **Game Logic and GUI Updates**:
    - After each "Drop" button action, updateGrid() is called to visually reflect the board's state.
    - If a game-ending condition (win or draw) is met, it prompts a reset or restart, ensuring the board reflects the new game state immediately.

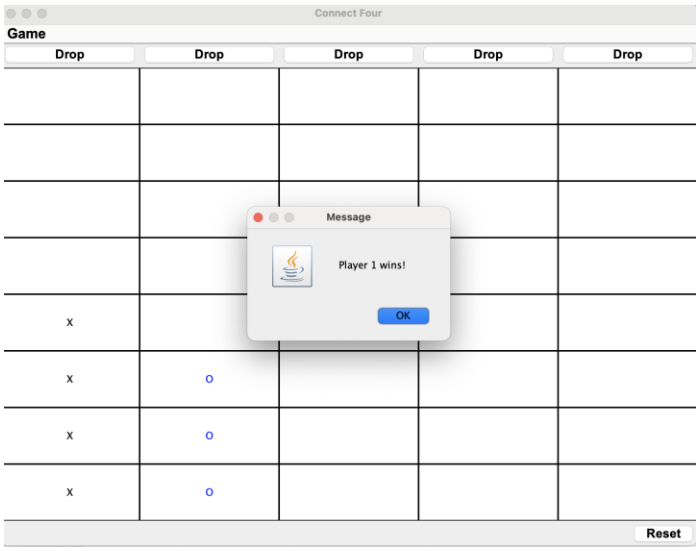
Testing:

- **White Box Testing**: Tests were conducted to validate all logical conditions, including edge cases where the board is nearly full or has complex winning patterns.

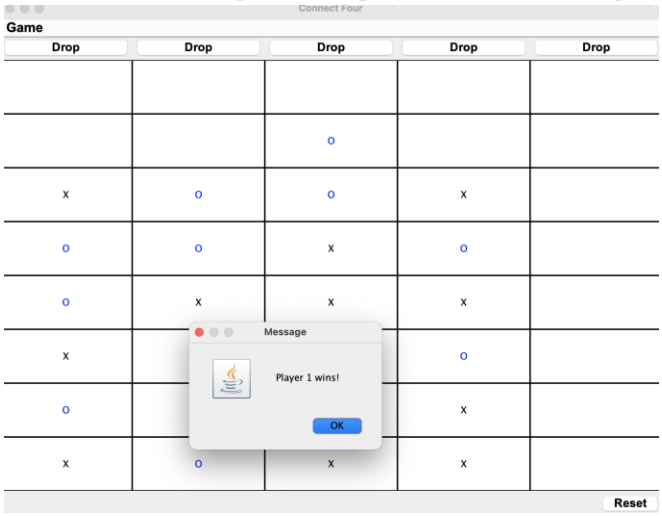
1. Horizontal Winning Condition
AS A user
I WANT TO run the application
GIVEN the board grid is visible
WHEN the user clicks drop button over respective column on the board to drop disc (e.g player1 drops disc X)
THEN the disc drop and the player switches and play so on until player1 wins horizontally.



2. Vertical Winning Condition
AS A user
I WANT TO run the application
GIVEN the board grid is visible
WHEN the user clicks drop button over respective column on the board to drop disc (e.g player1 drops disc X)
THEN the disc drop and the player switches and play so on until player1 wins vertically.

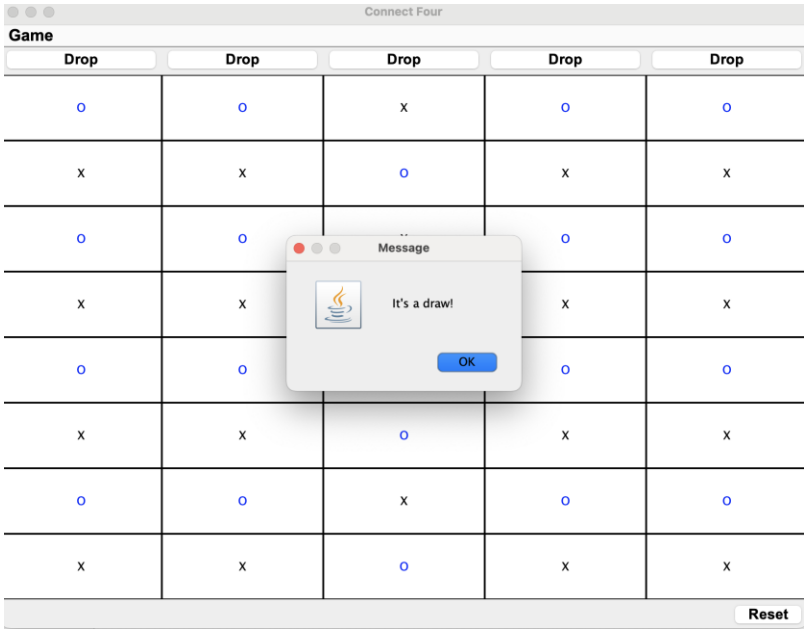


3. Diagonal Winning Condition
AS A user
I WANT TO run the application
GIVEN the board grid is visible
WHEN the user clicks drop button over respective column on the board to drop disc (e.g player1 drops disc X)
THEN the disc drop and the player switches and play so on until player1 wins diagonally.



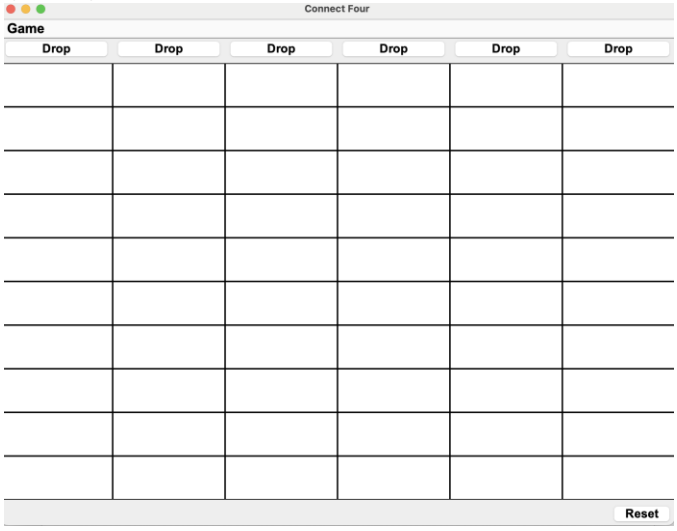
- **Input Tests**: Simulate disc placements to test winning and draw conditions, as well as game reset and board resizing.

1. Draw

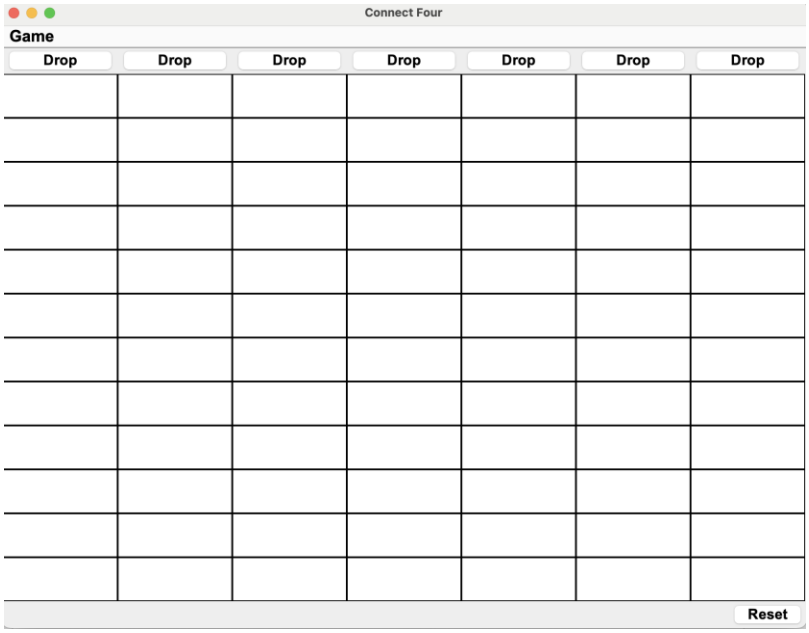


2. Board Resizing

(10 x 6)



(12 x 7)


Connect Four
Game
Drop
Drop
Drop
Drop
Drop
Drop
Drop
Reset