MAY 26, 2022

CONNECT FOUR GAME

DOCUMENTATION

YASH KASHYAP
21MCMC04

ADARSH LAMA
21MCMC32

Introduction:

The Connect Four Game is a Two-player game. The two sides select a colour each and fill the board to connect 4 boxes continuously. The first player to match the consecutive boxes with same colour horizontally, vertically or diagonally wins the game. We provide support for 2 colours only (Blue and Red) on a grid of 64 boxes (8 x 8).

Design of the Classes:

There are 2 major entities in the game – the board and the players. Thus, we have 2 classes `Board` and `Player`. The `Board` class manages the 8 x 8 grid and all the functions associated with it that help proceed the game. The `Player` class is to contain the information about the players that are playing the game. Furthermore, we have the main class `ConnectFour` where we fire up the GUI for the game.

Analysis of the Classes:

- **The `Board` class:**
  - Class Variables:
    - `numRows | int | private`:
      - Number of Rows in the grid
    - `numColumns | int | private`:
      - Number of Columns in the grid
    - `isWinnerDeclared | boolean | private`:
      - 0: not declared
      - 1: declared
      - 2: draw
    - `board[][] | String | private`:
      - The game board with inputs according to user-play.
  - Constructors:
    - `Board()`
      - Default constructor.
      - Initializes, `numRows` and `numColumns` with 8 and `board[][]` with empty String.
  - Class Methods:
    - `isVerticalConnected(int r, int c) | boolean | private:`
      - param :: `int r` :: row of the current cell
      - param :: `int c` :: column of the current cell
      - Checks whether the column containing the current cell has met the Connect 4 criteria.
    - `isHorizontalConnected(int r, int c) | boolean | private:`
      - param :: `int r` :: row of the current cell
      - param :: `int c` :: column of the current cell
      - Checks whether the row containing the current cell has met the Connect 4 criteria.
    - `isLeftDiagonalConnected(int r, int c) | boolean | private:`
      - param :: `int r` :: row of the current cell
      - param :: `int c` :: column of the current cell
      - Checks whether the left diagonal containing the current cell has met the Connect 4 criteria.

- `isRightDiagonalConnected(int r, int c) | boolean | private:`
  - param :: `int r` :: row of the current cell
  - param :: `int c` :: column of the current cell
  - Checks whether the right diagonal containing the current cell has met the Connect 4 criteria.
- `isConnected(int r, int c) | boolean | public:`
  - param :: `int r` :: row of the current cell
  - param :: `int c` :: column of the current cell
  - Checks whether the current cell completes the Connect 4 criteria. Runs checks through above mentioned `isVerticalConnected`, `isHorizontalConnected`, `isLeftDiagonalConnected`, `isRightDiagonalConnected` methods.
- `getResult() | int | public:`
  - returns `isWinnerDeclared`
- `getBoard() | String[][] | public:`
  - returns `board[][]`
- `pushMove(String color, int column) | void | public:`
  - param :: `String color` :: color of the player that makes the move
  - param :: `int column` :: column in which the move is pushed
  - Makes a move and updates the `isWinnerDeclared` variable.
- `showBoard() | void | public:`
  - displayed `board[][]`

➢ The `Player` class:

❖ Class Variables:

- `username | String | private:`
  - username of the Player
- `color | String | private:`
  - color choice for the Player (Blue / Red)

❖ Constructors:

- `Player():`
  - initializes `username` with empty String.
  - Initializes `color` with empty String.

❖ Class Methods:

- `setUsername(String username) | void | public:`
  - param :: `String username` :: the username to be set.
  - Sets the `username` variable of the class.
- `setColor(String color) | void | public:`
  - param :: `String color` :: the color to be set.
  - Sets the `color` variable of the class.
- `getUsername() | String | public:`
  - returns `username`
- `getColor() | String | public:`
  - returns `color`

➢ The `ConnectFour` class:

❖ The main class, `ConnectFour`, implements the `ActionListener` interface and builds up a GUI for the game.

❖ Initializes class variables required for setting up the UI.

- ❖ Performs different actions on the click of different buttons.
- ❖ How the game proceeds:
    - Initially user needs to enter the player information i.e. username and color, for each player one by one.
    - Player A begins the game, always.
    - The push buttons in the board provokes the `actionPerformed` method to push the color corresponding to the player to the respective column.
    - The label above the board provides the necessary information as the game proceeds.
    - As soon as a winner is declared, the game stops.
    - To replay the game user needs to rerun the program.

Dependencies:

- ➢ java.awt
- ➢ javax.swing

How to run:

- ➢ Make sure you have JDK installed in your PC.
- ➢ Unzip "ConnectFourGUI.rar"
- ➢ Open command prompt in the project folder.
- ➢ Run command "`java ConnectFour`"

Conclusion:

The classes have been designed in such a way that the UI once fired up creates an object of the `Board` class and 2 objects of `Player` class. The grid is managed by the `Board` class and as the game proceeds the changes in the matrix are reflected on the grid in the UI. The game stops as soon as a winner is declared.