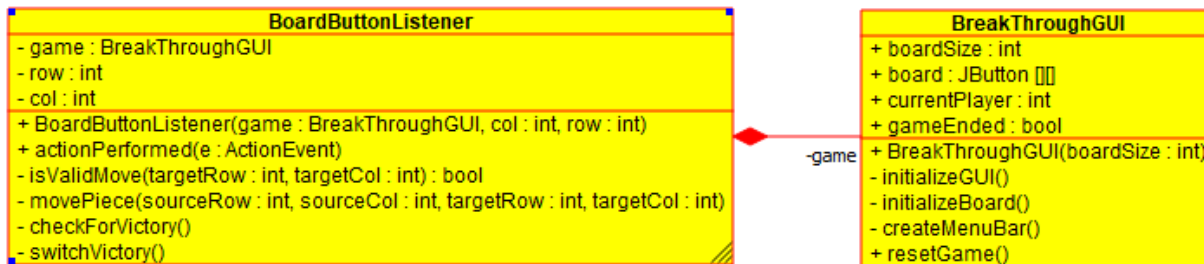


NAME: ARYAN SHAMS ANSARI

NEPTUN: GTPHX8

Break-through Break-through is a two-player game, played on a board consists of $n \times n$ fields. Each player has $2n$ dolls in two rows, placed on at the player's side initially (similarly to the chess game, but now every dolls of a player look like the same). A player can move his doll one step forward or one step diagonally forward (can't step backward). A player can beat a doll of his opponent by stepping diagonally forward onto it. A player wins when his doll reaches the opposite edge of the board. Implement this game, and let the board size be selectable (6x6, 8x8, 10x10). The game should recognize if it is ended, and it has to show in a message box which player won. After this, a new game should be started automatically.



BreakThroughGUI Class Methods:

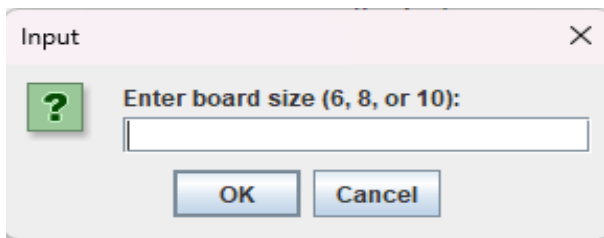
- **BreakThroughGUI(boardSize : int)** -> Simple constructor which takes an integer parameter "boardSize".
- **InitializeGUI()** -> It configures the JFrame with a title, size, close operation, layout manager, initializes the game board, creates a menu bar, and then makes the GUI visible to the user.
- **InitializeBoard()** -> Gives colors to the buttons on the board.
- **CreateMenuBar()** -> This method creates a menu bar with a "Game" menu that contains a "New Game" menu item. Clicking on "New Game" triggers the **resetGame()** method, and the menu bar is set for the main JFrame.
- **ResetGame()** -> Resets the game with some simple operations.

BoardButtonListener Class Methods:

- **BoardButtonListener(game: BreakThroughGUI, col : int, row : int)** -> Simple constructor with three parameters.
- **ActionPerformed (e : ActionEvent)** -> This method is overriding **actionPerformed** method in **ActionListener** class, the method features the movement in the game. By clicking on each button you move forward, forward-left, forward-right.
- **IsValidMove(targetRow : int, targetCol : int) : bool** -> the methods checks if the targeted button is inside the matrix of buttons.
- **movePiece(sourceRow : int, sourceCol : int, targetRow : int, targetCol : int)** -> This method is swapping the color of source button to target button.
- **CheckForVictory()** -> Checks if the Piece has reached the end of the board, (declares the winner).
- **SwitchVictory()** -> This method implements the logic for turn of each player.

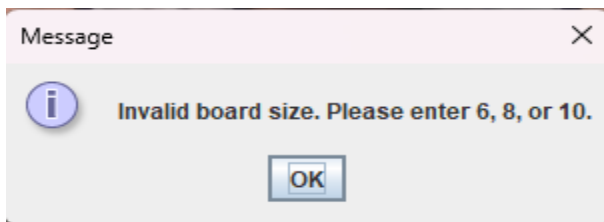
Testing:

When we start the game, we must input three specific numbers, if not we cannot proceed.



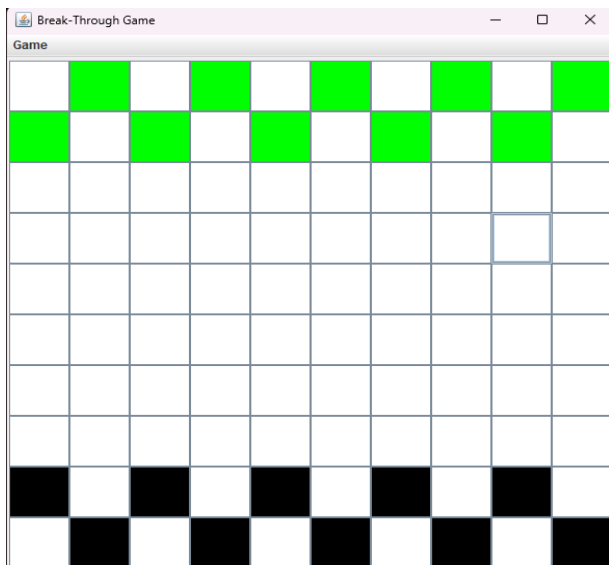
An input dialog box titled "Input" with a close button (X) in the top right corner. It contains a green square icon with a question mark on the left. The text "Enter board size (6, 8, or 10):" is followed by a text input field. Below the input field are two buttons: "OK" and "Cancel".

For example, if we input 99, we get this error prompt.

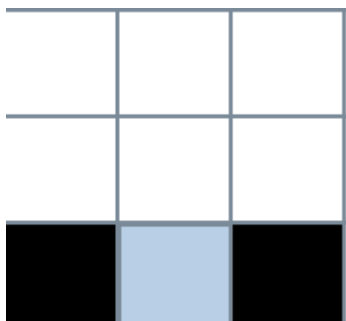


A message dialog box titled "Message" with a close button (X) in the top right corner. It contains a blue circular icon with an 'i' on the left. The text "Invalid board size. Please enter 6, 8, or 10." is displayed. Below the text is an "OK" button.

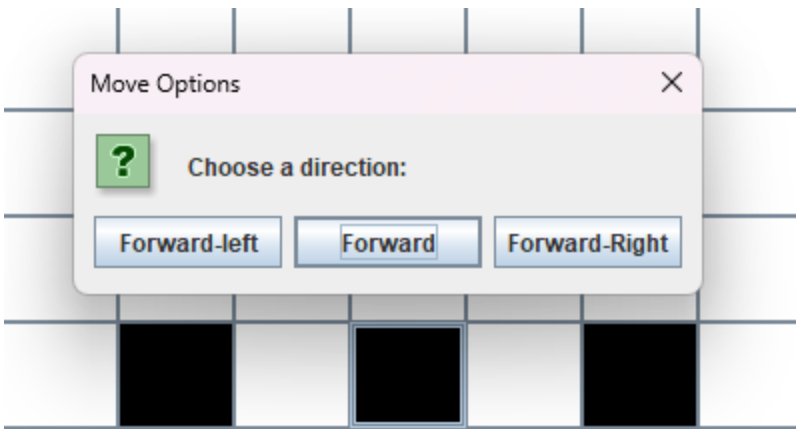
If 6, 8, 10 is entered the game starts.



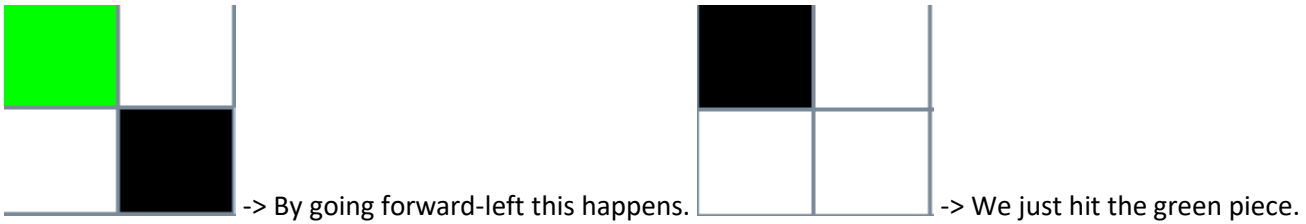
We can't select white fields.



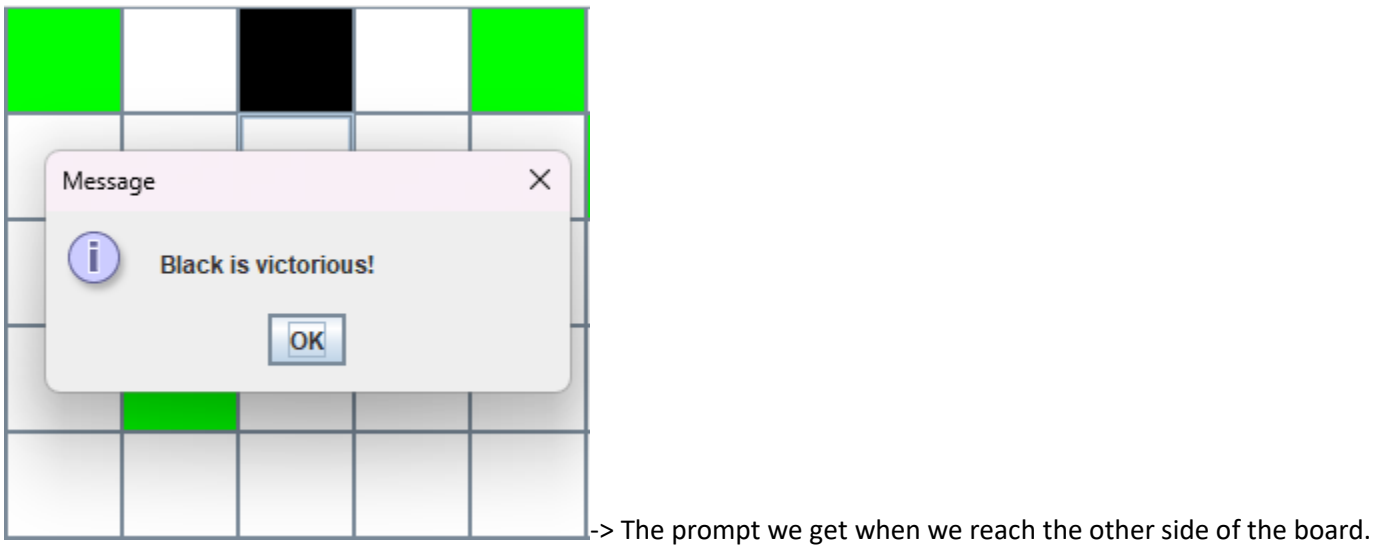
When we click on a colored field this is the prompt we get, the options are: Forward-left, forward, forward-right.



We can also hit other pieces of the opponent.



The goal of the game is to reach the other side of the board, the player who accomplishes this first is the winner.



We can also reset the game by clicking on the menu item (New Game).

