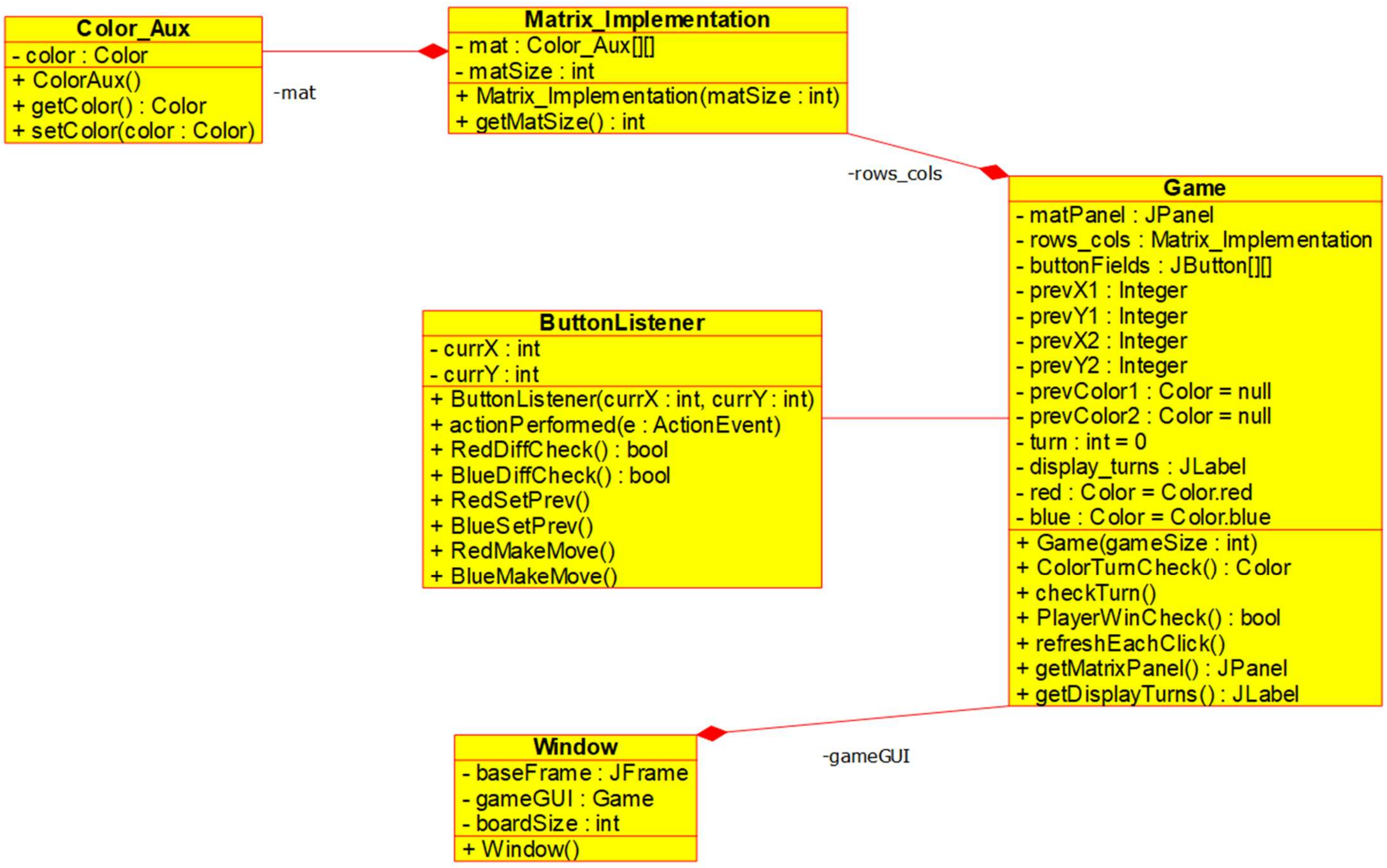
Munkhtenger Munkh-Aldar **2. assignment/7th. Task** **09 Nov 2022**
WSR292
WSR292@inf.elte.hu

Task

Break-through is a two-player game, played on a board consists of n x 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.

UML Class Diagram

This diagram was made with Umbrello.



Description of each method

Game Class

- ***Constructer (gameSize):***
  Constructor for implementing the game panel with the given game size

- ***ColorTurnCheck():***
  Method for checking which player's turn and gives back color

- ***checkTurn():***
  Method for checking which player's turn and gives back player's name in string format

- ***PlayerWinCheck():***
  Method for checking if a player won by reaching end row of the other side and giving back logical value

- ***refreshEachClick():***
  Method for checking which player won and displays message dialog with winner's name

ButtonListener Class

- ***Constructor:***
  Constructs the class with given x and y of the Game class.

- ***actionPerformed (ActionEvent e):*** It checks three cases.
  1. We don't know where the doll will be, we only know its current position. In this case, we don't do anything.
  2. We know where the next position is. Here we will determine if the pebble wants to move forward or diagonally forward, then we calculate the move and affected button field positions.
  3. The move is not allowed when the player tries to move back, right side, left side, tries to skip fields, or tries to move on opposthe ite player's turn

- ***RedDiffCheck():***
  Method for checking if the red player played a different move

- ***BlueDiffCheck():***
  Method for checking if the blue player played a different move

- ***RedSetPrev():***
  Method for resetting previous red player's point to null and increment turn

- ***BlueSetPrev():***
  Method for resetting previous blue player's point to null and increment turn

- ***RedMakeMove():***
  If the red player made a valid move, then this method will make the previous move's icon and color into a new one.

- ***BlueMakeMove():***
  If the red player made a valid move, then this method will make the previous move's icon and color into a new one.

Window Class

- ***Constructor***: Constructs a window, the game panel, and the label.

Matrix_Implementation Class

- ***Constructor (matSize):***
  Constructor for Matrix_Implementation. Takes matrix size as argument and initializes matrix of that given size

- ***getMatSize():*** Getter for matrix size.

ColorAux Class

- ***Constructor:*** Empty constructer for ColorAux class

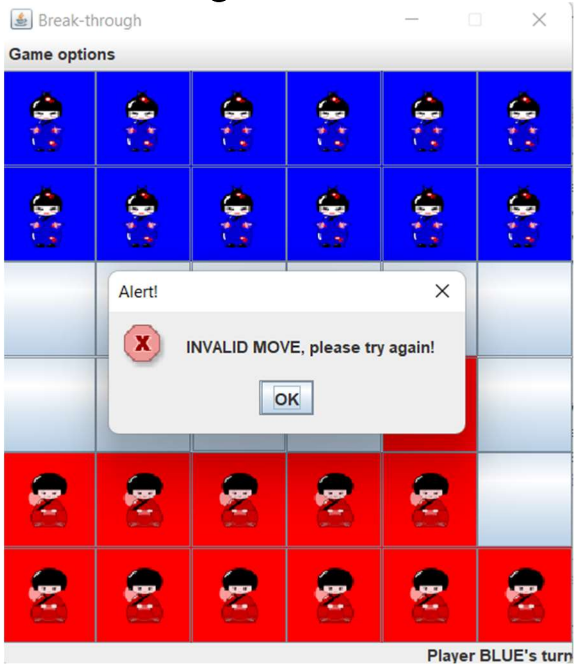
- ***getColor()***: getter for color

- ***setColor(Color color):*** setter for color

Tests

1. If a player tries to move backward, left, right or tries to skip a field

   Error message: INVALID MOVE, please try again!

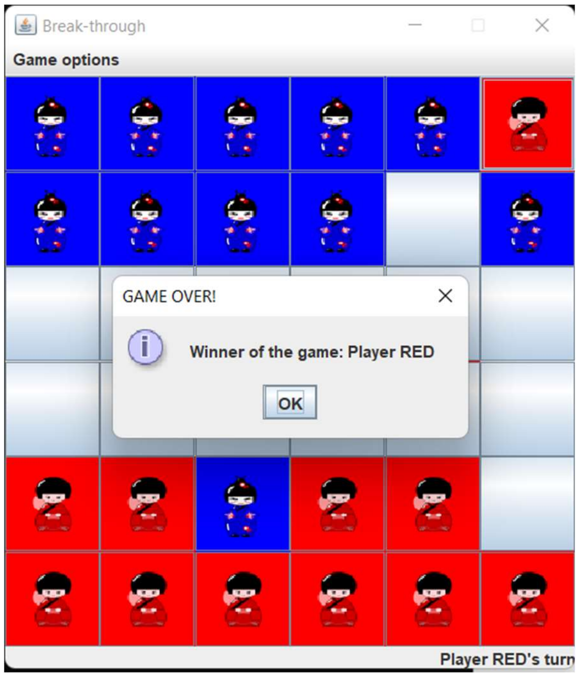


Alert!

INVALID MOVE, please try again!

Player BLUE's turn

2. Check if the game recognizes it has ended and outputs the winner

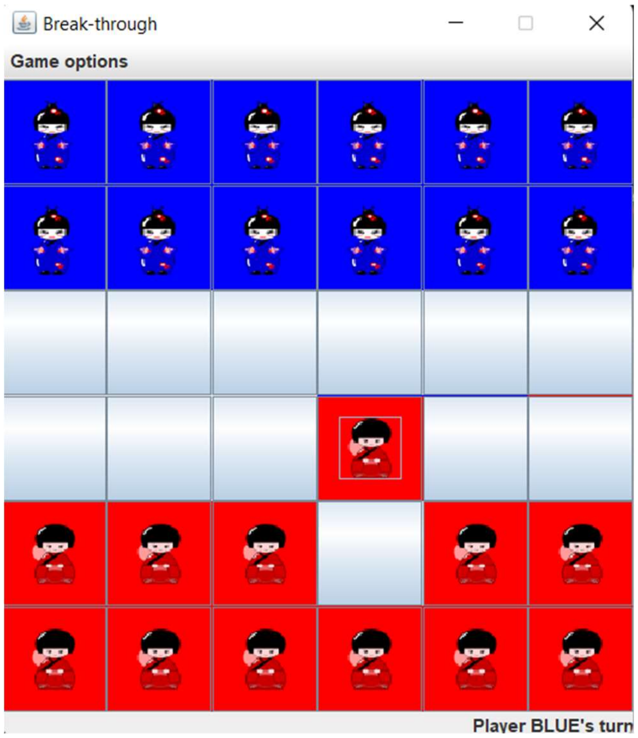


GAME OVER!

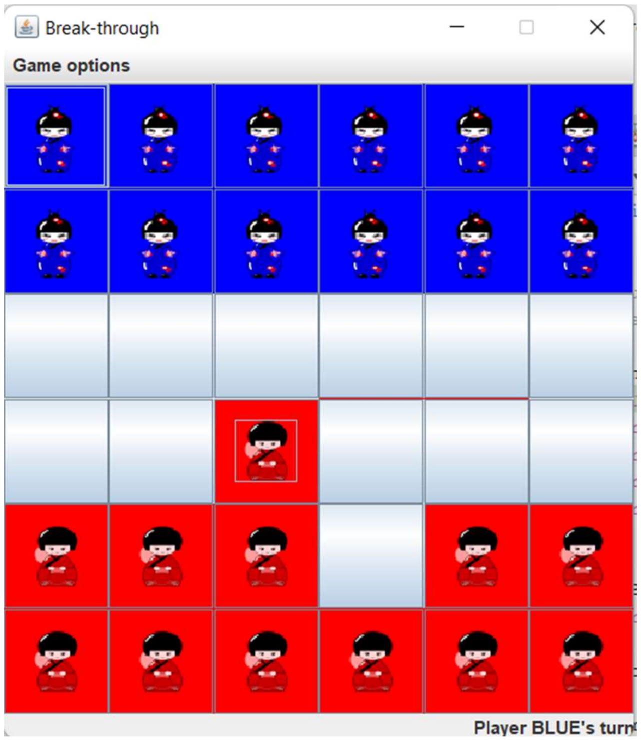
Winner of the game: Player RED

Player RED's turn

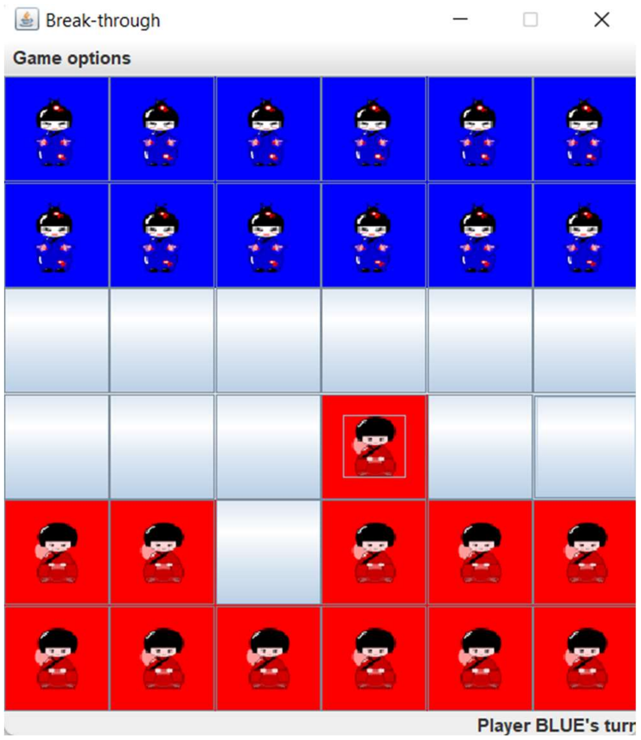
3. Check if the red player can move 1 forward



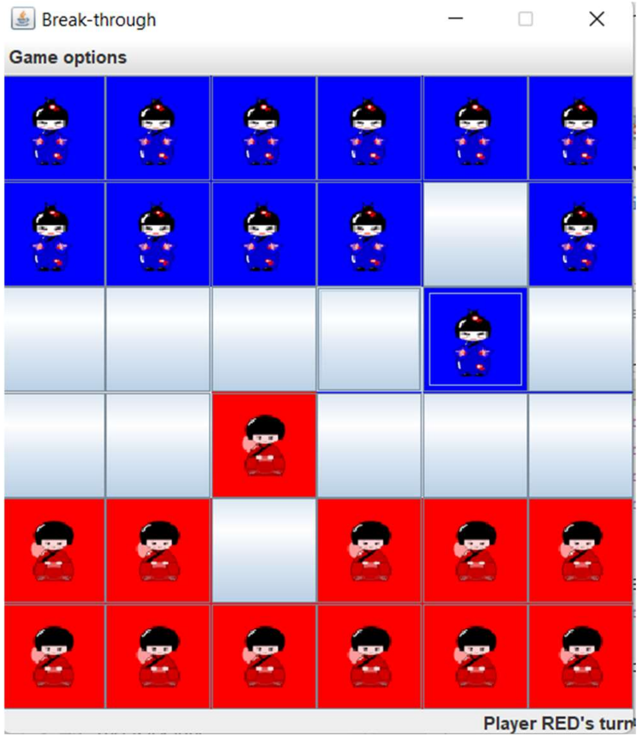
4. Check if the red player can move 1 unit left diagonal forward



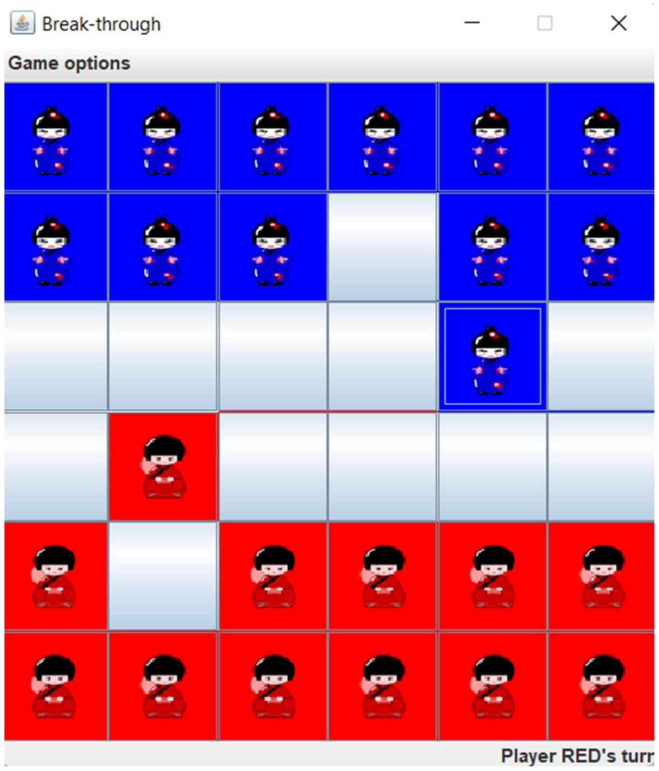
5. Check if the red player can move 1 unit right diagonal forward



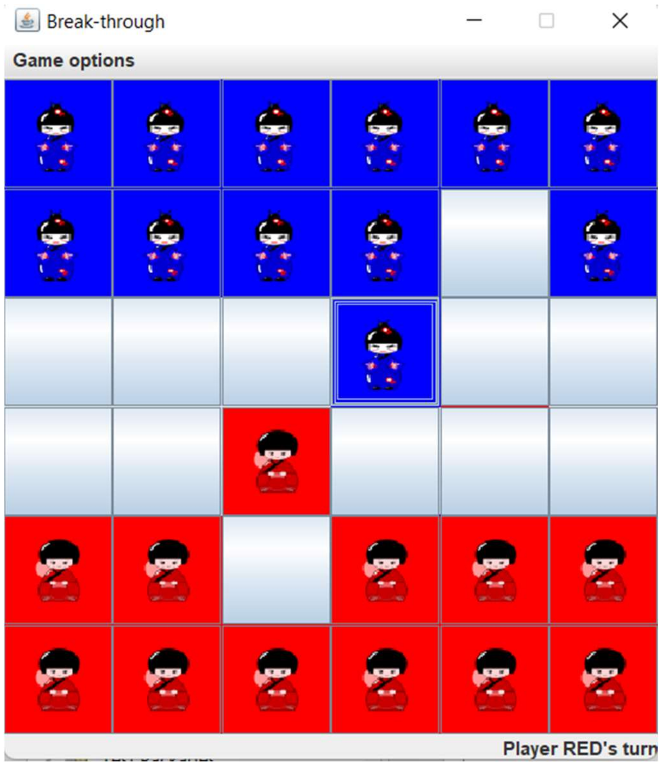
6. Check if the blue player can move 1 unit forward



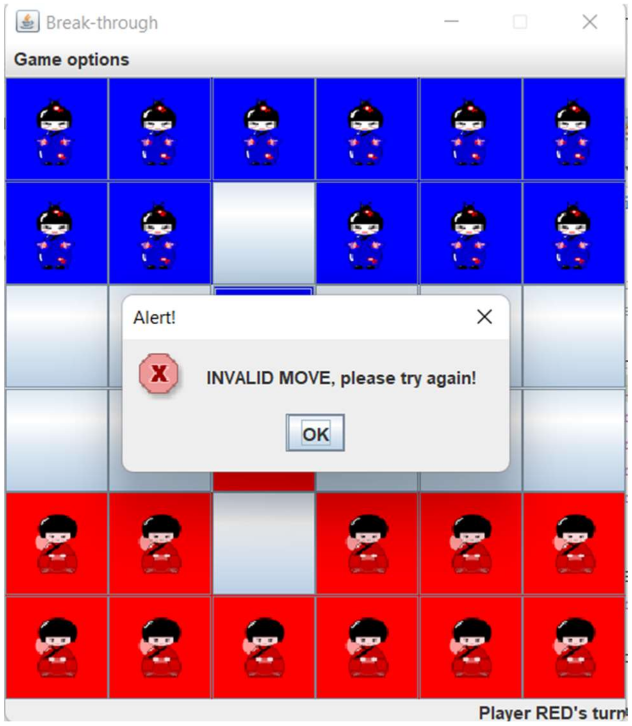
7. Check if the blue player can move 1 unit right diagonal forward



8. Check if the blue player can move 1 unit left diagonal forward



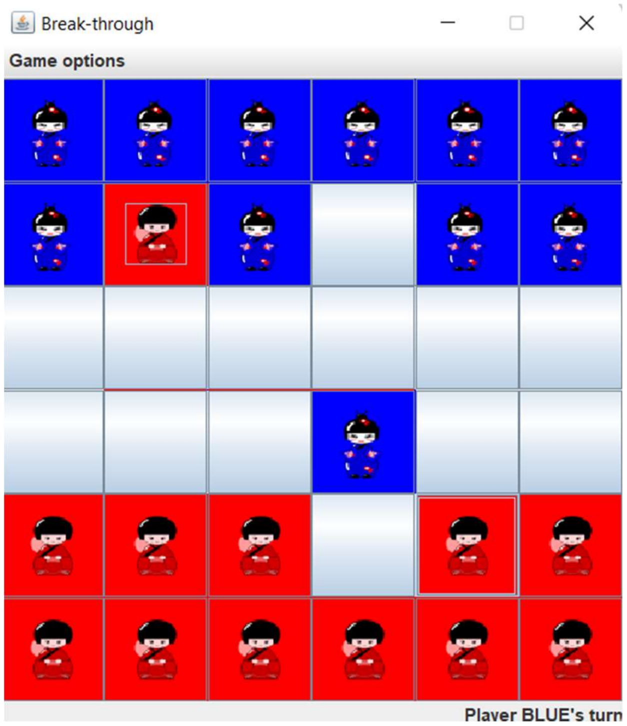
9. Check if moving forward when there is a doll in the front



INVALID MOVE, please try again!

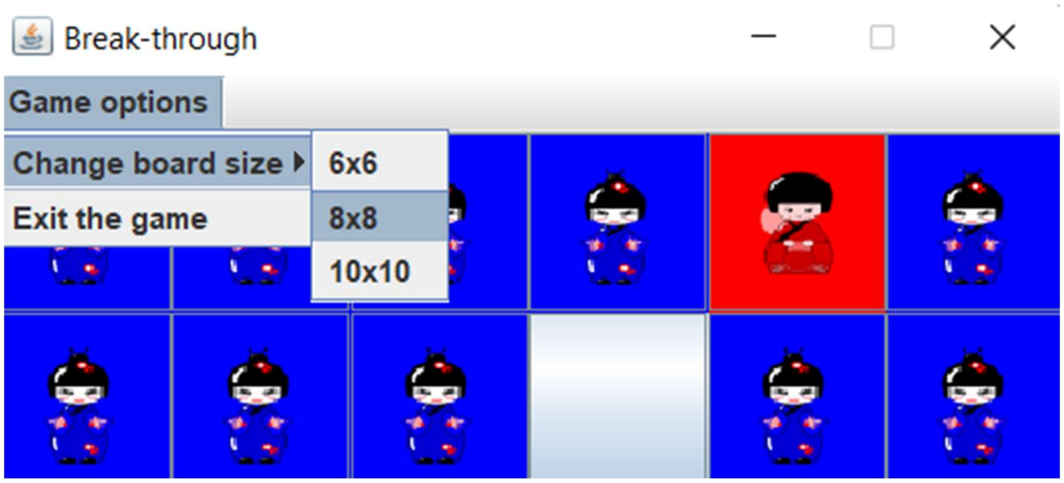
Player RED's turn

10. Check if one player's doll can beat opposite player's doll by moving diagonally forward

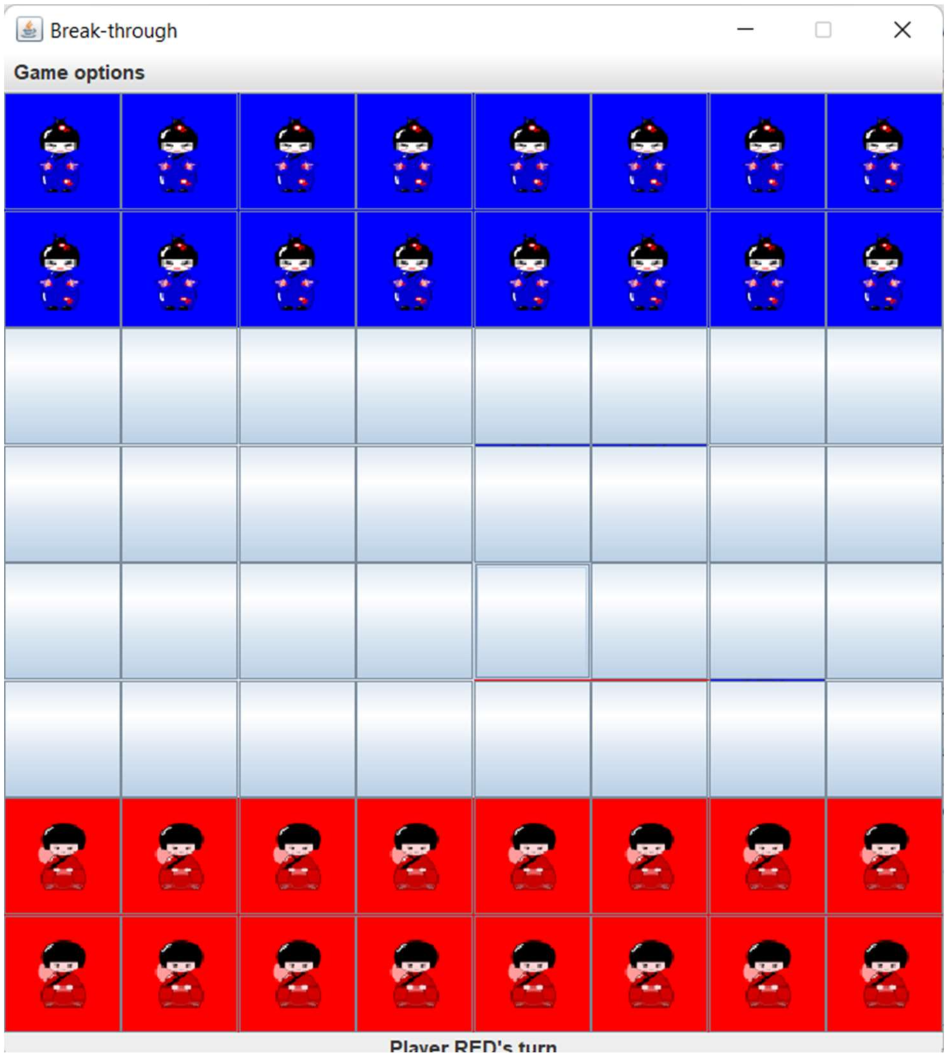


Player BLUE's turn

11. Board size menu options are functioning

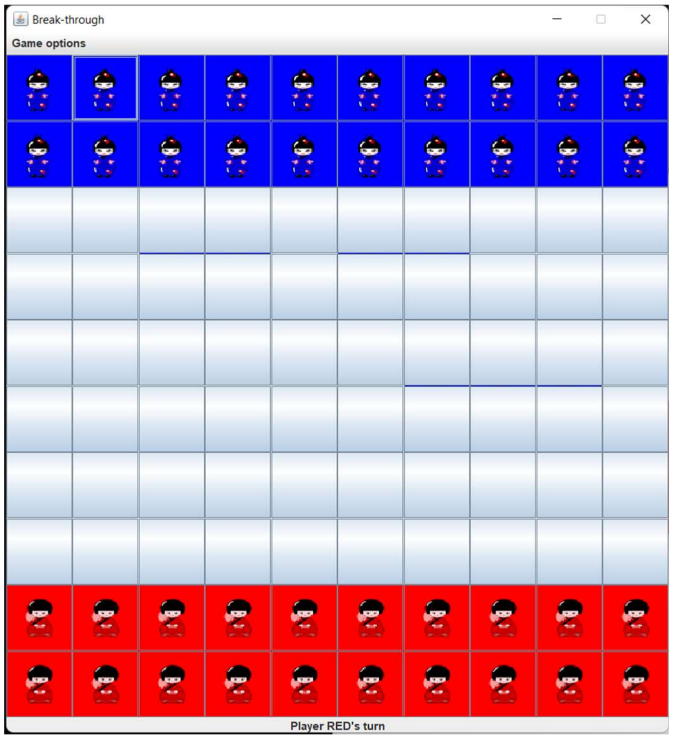


12. 8X8 board size



Player RED's turn

13. 10x10 board size



Player RED's turn