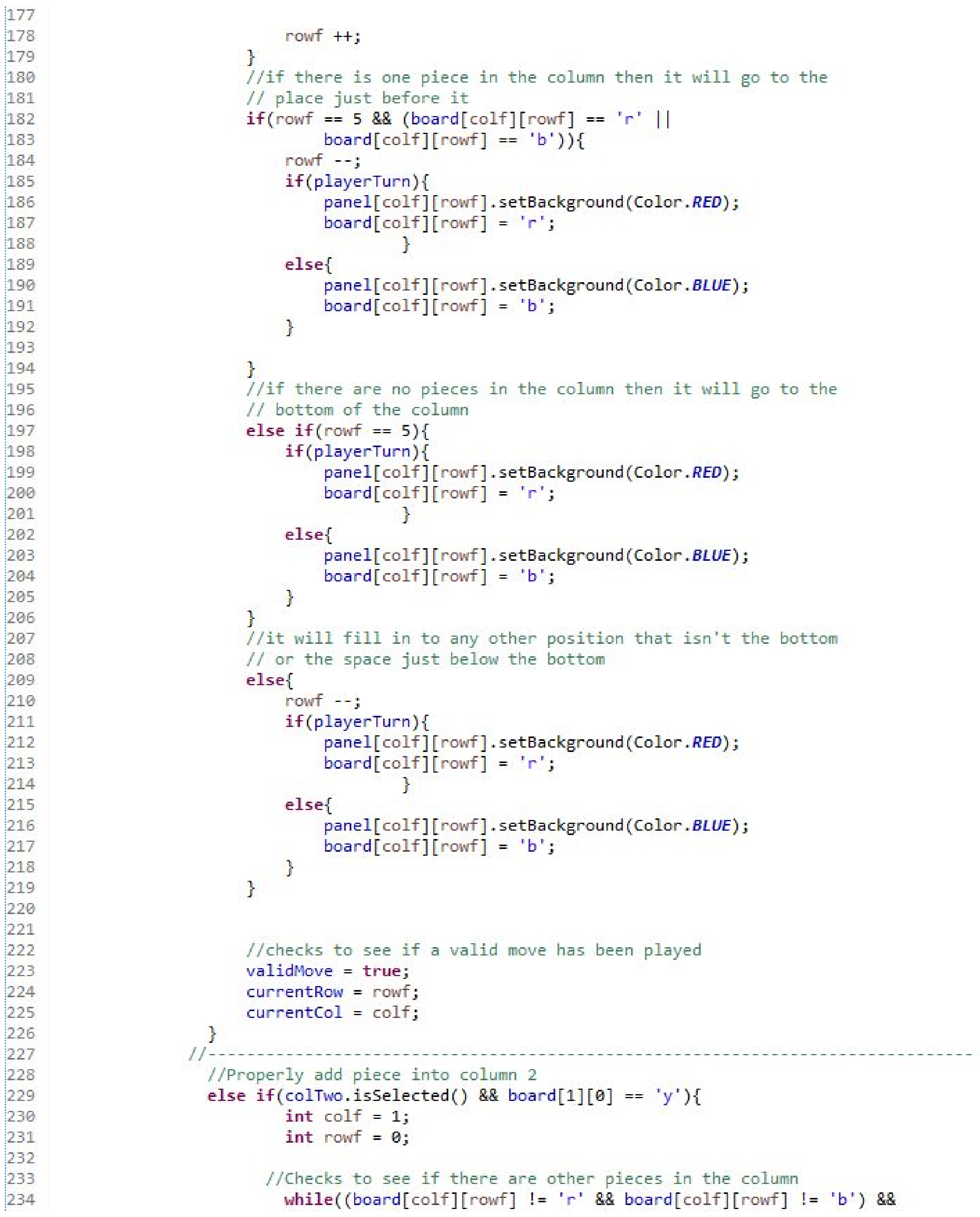
```java

import java.awt.*;
import java.awt.Color.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.HashSet;
import java.util.Set;
import javax.swing.JOptionPane;

/*  Programmed by: Ryan Martinez
 *  Class        : CPSC 223J
 *  Section      : Tu/Th 8:00AM
 *  Project      : Final Project
 *
 *  Description  : This Program emulates a 2 player connect four game.
 *  In this program there is a menu screen with a button that allows
 *  the user to start the game. Following the menu screen there is an
 *  emulated connect four board that allows each user to take their
 *  turn and choose the column they wish to insert their piece. The game
 *  covers the win case of horizontal, vertical, and both diagonals. The
 *  game also covers the case of a draw.
 *
 */
public class ConnectFour extends JFrame implements ActionListener{
    private final int ROWS = 6;     // variable for the number of rows on the board
    private final int COLS = 7;     // variable for the number of columns on the board

    private boolean validMove = false; // variable to check if a move is valid
    private boolean checkWin = false;  // variable to check if a user has won
    private boolean checkDraw = false; // variable to check to see if the game is a draw
    private int currentRow = 0;     // used to keep track of row active location
    private int currentCol = 0;     // used to keep track of column active location

    //elements for the game
    private char board[][] = new char[COLS][ROWS]; // board that stores the char of neutral, red, or blue
    private JPanel panel[][] = new JPanel[COLS][ROWS]; // array of pannels holding colors for the char array

    // various panels used to format the elements used in the program
    private JPanel topHome = new JPanel();
    private JPanel middleHome = new JPanel();
    private JPanel home = new JPanel();
    private JPanel gameScreen = new JPanel();
    private JPanel topGame = new JPanel();
    private JPanel buttonsGame = new JPanel();
    private JPanel turnGame = new JPanel();
    private JPanel bottomGame = new JPanel();

    private JButton next = new JButton("Start"); // button used to enter the game from the home screen

    private CardLayout cardLayout = new CardLayout(); // creates a new card layout for multiple screens

    // button group is created and adds all of the different buttons for the different columns
    private ButtonGroup group = new ButtonGroup();
    private JRadioButton colOne = new JRadioButton("Col 1");
    private JRadioButton colTwo = new JRadioButton("Col 2");
    private JRadioButton colThree = new JRadioButton("Col 3");
    private JRadioButton colFour = new JRadioButton("Col 4");
    private JRadioButton colFive = new JRadioButton("Col 5");
    private JRadioButton colSix = new JRadioButton("Col 6");
    private JRadioButton colSeven = new JRadioButton("Col 7");


    private JLabel turn = new JLabel("Current Player's Turn: Red"); // initializes the label for turn
    private boolean playerTurn = true; // sets the active players turn - true is red - false is blue
    private JButton activate = new JButton("Go"); // button for a player to input their turn
    private JButton newGame = new JButton("Reset"); // button to reset the game after it has been finished

    // labels for the home screen display
    private JLabel intro = new JLabel("Welcome to my Connect Four game");
    private JLabel intro2 = new JLabel("please click the button to start");

    public ConnectFour(){
        super("Connect Four"); // names the program window
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // closes program on exit of window
        setLayout(cardLayout); // adds the card layout for the multiple windows to be changed between

        //formatting for the first card - the home screen
        home.setBackground(Color.CYAN);
        home.setLayout(new GridLayout(3,0,50,50));
        intro.setFont(new Font("Arial", Font.BOLD, 26));
        intro2.setFont(new Font("Arial", Font.BOLD, 26));
        topHome.setLayout(new FlowLayout());
        topHome.add(intro);
        topHome.setBackground(Color.CYAN);
        middleHome.setLayout(new FlowLayout());
        middleHome.add(intro2);
        middleHome.setBackground(Color.CYAN);
        home.add(topHome);
        home.add(middleHome);
        home.add(next);

        //formatting for the second card - the game screen
        group.add(colOne);
        group.add(colTwo);
        group.add(colThree);
        group.add(colFour);
        group.add(colFive);
        group.add(colSix);
        group.add(colSeven);
        gameScreen.setLayout(new BorderLayout());
        turnGame.setLayout(new FlowLayout(FlowLayout.CENTER, 200,0));
        topGame.setLayout(new BorderLayout());
        turnGame.add(turn);
        turnGame.add(activate);
        buttonsGame.setLayout(new GridLayout(1,7,0,100));
        bottomGame.setLayout(new GridLayout(6,7,5,5));
        topGame.add(turnGame, BorderLayout.NORTH);
        topGame.add(buttonsGame, BorderLayout.CENTER);
        gameScreen.add(topGame, BorderLayout.NORTH);

        // adds the buttons to the button bar
        buttonsGame.add(colOne);
        buttonsGame.add(colTwo);
        buttonsGame.add(colThree);
        buttonsGame.add(colFour);
        buttonsGame.add(colFive);
        buttonsGame.add(colSix);
        buttonsGame.add(colSeven);
        // formats the header above the game
        gameScreen.add(topGame, BorderLayout.NORTH);
        // adds the game to the game screen
        gameScreen.add(bottomGame, BorderLayout.CENTER);

        // adds action listeners to all buttons that are going to be used in the program
        activate.addActionListener(this);
        next.addActionListener(this);
        newGame.addActionListener(this);
        colOne.addActionListener(this);
        colTwo.addActionListener(this);
        colThree.addActionListener(this);
        colFour.addActionListener(this);
        colFive.addActionListener(this);
        colSix.addActionListener(this);
        colSeven.addActionListener(this);

        // adds the different cards to the card layout
        add("home", home);
        add("game", gameScreen);

    }


    @Override
    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        // the button that goes from home screen to the game
        if(source == next){
            // initializes the arrays that are used in the board so they are all neutral
            for(int y = 0; y < ROWS; ++y){
                for(int x = 0; x < COLS; ++x){
                    board[x][y] = 'y';
                    panel[x][y] = new JPanel();
                    bottomGame.add(panel[x][y]);
                    panel[x][y].setBackground(Color.YELLOW);

            }
        }
            // swaps to the game card
            cardLayout.next(getContentPane());
        }
        else if(source == activate){
            //checks to see if one of the columns is selected
            if(colOne.isSelected() || colTwo.isSelected() ||
               colThree.isSelected() || colFour.isSelected() ||
               colFive.isSelected() || colSix.isSelected() ||
               colSeven.isSelected()){
                //-------------------------------------------------------------------------
                //entering pieces into the program if they can fit
                if(colOne.isSelected() && board[0][0] == 'y'){
                    int colf = 0;
                    int rowf = 0;


                    //Checks to see if there are other pieces in the column
                    while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
                            rowf < 5){
```



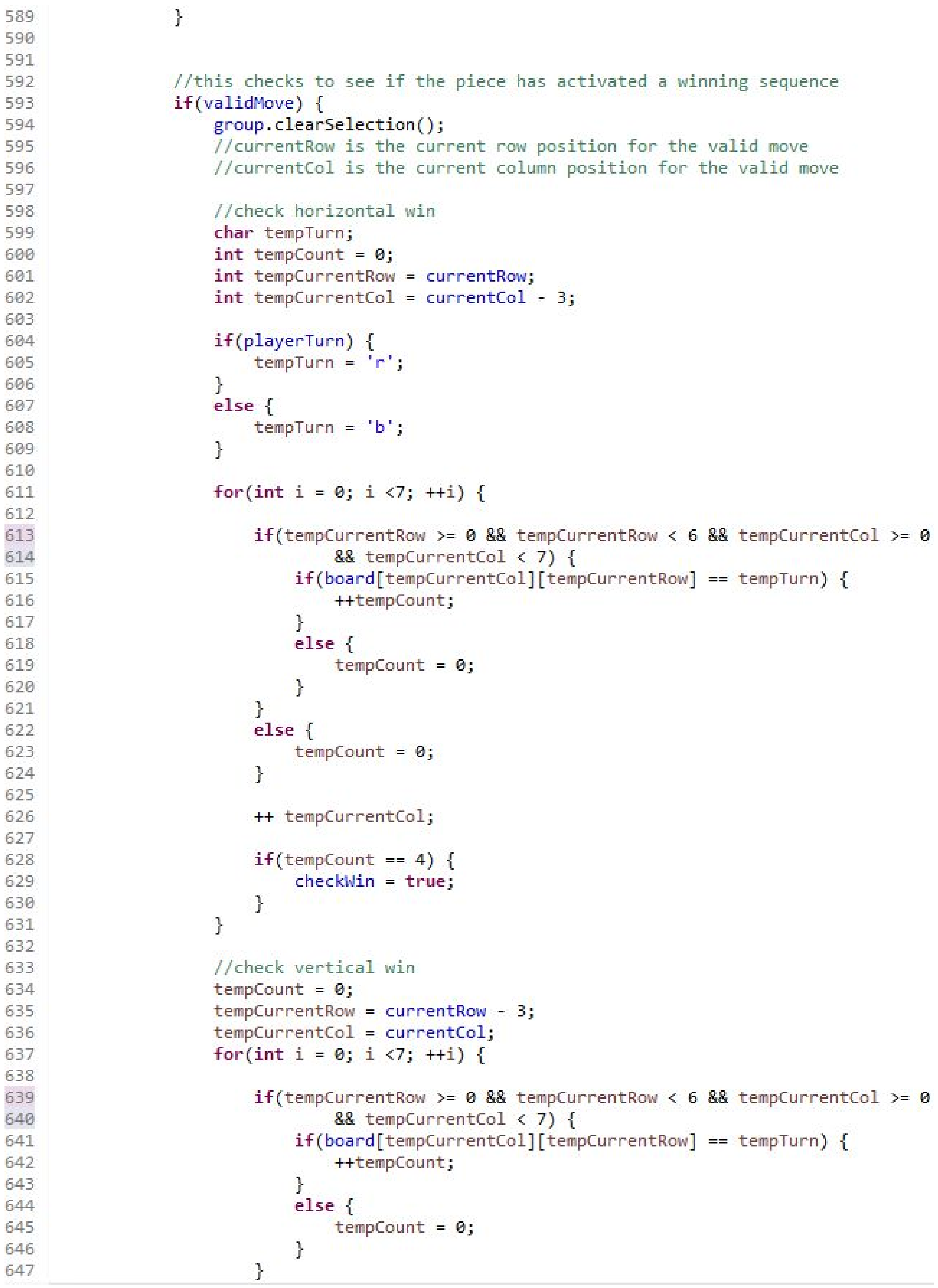
```
          rowf ++;
      }
      //if there is one piece in the column then it will go to the
      // place just before it
      if(rowf == 5 && (board[colf][rowf] == 'r' ||
              board[colf][rowf] == 'b')){
          rowf --;
          if(playerTurn){
              panel[colf][rowf].setBackground(Color.RED);
              board[colf][rowf] = 'r';
                      }
          else{
              panel[colf][rowf].setBackground(Color.BLUE);
              board[colf][rowf] = 'b';
          }

      }
      //if there are no pieces in the column then it will go to the
      // bottom of the column
      else if(rowf == 5){
          if(playerTurn){
              panel[colf][rowf].setBackground(Color.RED);
              board[colf][rowf] = 'r';
                      }
          else{
              panel[colf][rowf].setBackground(Color.BLUE);
              board[colf][rowf] = 'b';
          }
      }
      //it will fill in to any other position that isn't the bottom
      // or the space just below the bottom
      else{
          rowf --;
          if(playerTurn){
              panel[colf][rowf].setBackground(Color.RED);
              board[colf][rowf] = 'r';
                      }
          else{
              panel[colf][rowf].setBackground(Color.BLUE);
              board[colf][rowf] = 'b';
          }
      }


      //checks to see if a valid move has been played
      validMove = true;
      currentRow = rowf;
      currentCol = colf;
  }
//-------------------------------------------------------------------------------
  //Properly add piece into column 2
  else if(colTwo.isSelected() && board[1][0] == 'y'){
          int colf = 1;
          int rowf = 0;

        //Checks to see if there are other pieces in the column
          while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
```

```java
                                rowf < 5){
                            rowf ++;
                        }

                      //if there is one piece in the column then it will go to the
                      // place just before it
                        if(rowf == 5 && (board[colf][rowf] == 'r' ||
                                board[colf][rowf] == 'b')){
                            rowf --;
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }

                        }
                      //if there are no pieces in the column then it will go to the
                      // bottom of the column
                        else if(rowf == 5){
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }
                        }
                      //it will fill in to any other position that isn't the bottom
                      // or the space just below the bottom
                        else{
                            rowf --;
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }
                        }

                    //checks to see if a valid move has been played
                    validMove = true;
                    currentRow = rowf;
                    currentCol = colf;
                }
    //-------------------------------------------------------------------------------
              //Properly add piece into column 3
                else if(colThree.isSelected() && board[2][0] == 'y'){
                        int colf = 2;
                        int rowf = 0;

                      //Checks to see if there are other pieces in the column
                        while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
                                rowf < 5){
```

```java
                            rowf ++;
                        }

                      //if there is one piece in the column then it will go to the
                      // place just before it
                        if(rowf == 5 && (board[colf][rowf] == 'r' ||
                                board[colf][rowf] == 'b')){
                            rowf --;
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }

                        }
                      //if there are no pieces in the column then it will go to the
                      // bottom of the column
                        else if(rowf == 5){
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }
                        }
                      //it will fill in to any other position that isn't the bottom
                      // or the space just below the bottom
                        else{
                            rowf --;
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }
                        }

                    //checks to see if a valid move has been played
                    validMove = true;
                    currentRow = rowf;
                    currentCol = colf;
                }
              //--------------------------------------------------------------------------------
                //Properly add piece into column 4
                  else if(colFour.isSelected() && board[3][0] == 'y'){
                          int colf = 3;
                          int rowf = 0;

                        //Checks to see if there are other pieces in the column
                          while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
                                  rowf < 5){
                              rowf ++;
```

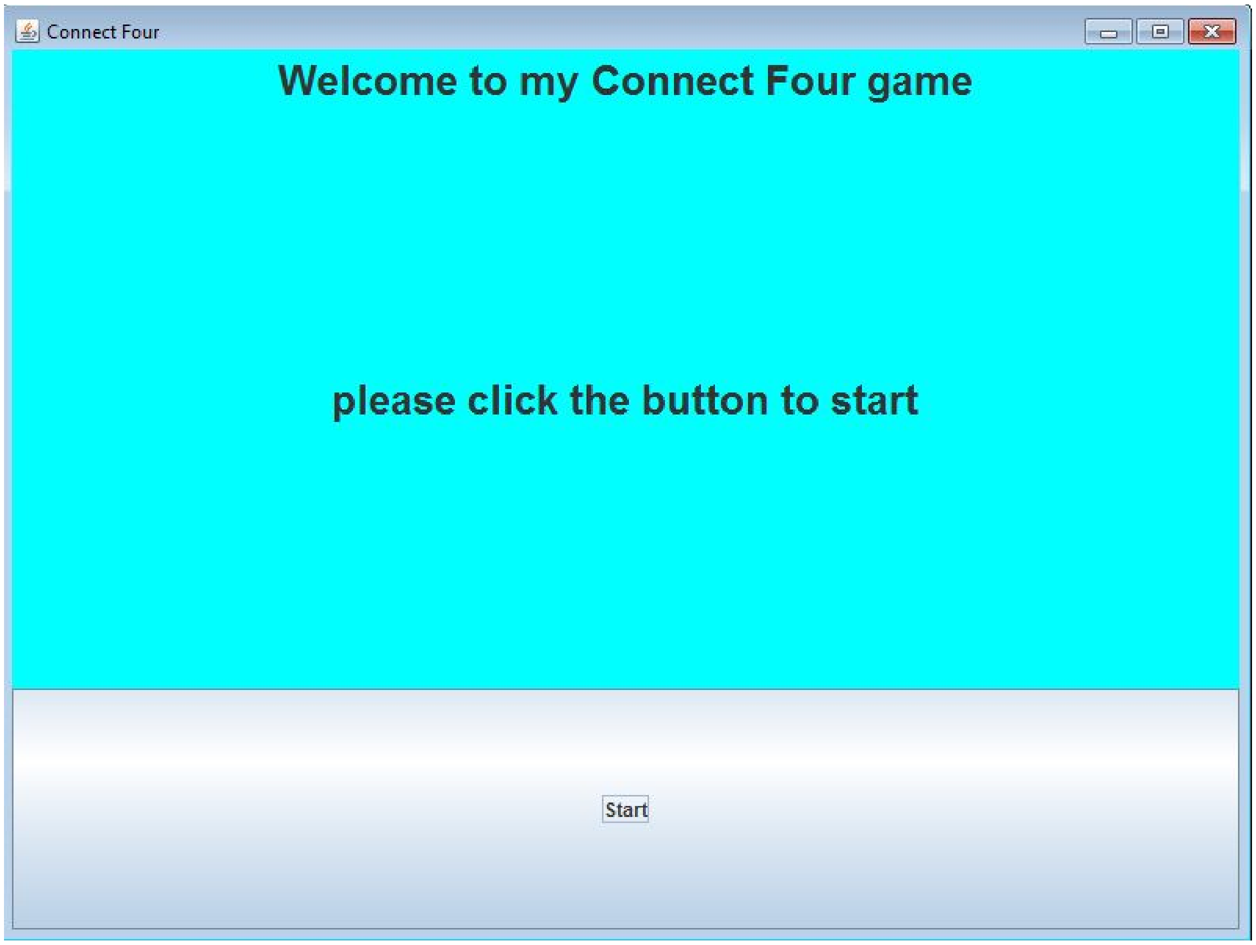
```java
            }

          //if there is one piece in the column then it will go to the
          // place just before it
            if(rowf == 5 && (board[colf][rowf] == 'r' ||
                    board[colf][rowf] == 'b')){
                rowf --;
                if(playerTurn){
                    panel[colf][rowf].setBackground(Color.RED);
                    board[colf][rowf] = 'r';
                            }
                else{
                    panel[colf][rowf].setBackground(Color.BLUE);
                    board[colf][rowf] = 'b';
                }

            }
          //if there are no pieces in the column then it will go to the
          // bottom of the column
            else if(rowf == 5){
                if(playerTurn){
                    panel[colf][rowf].setBackground(Color.RED);
                    board[colf][rowf] = 'r';
                            }
                else{
                    panel[colf][rowf].setBackground(Color.BLUE);
                    board[colf][rowf] = 'b';
                }
            }
          //it will fill in to any other position that isn't the bottom
          // or the space just below the bottom
            else{
                rowf --;
                if(playerTurn){
                    panel[colf][rowf].setBackground(Color.RED);
                    board[colf][rowf] = 'r';
                            }
                else{
                    panel[colf][rowf].setBackground(Color.BLUE);
                    board[colf][rowf] = 'b';
                }
            }

        //checks to see if a valid move has been played
        validMove = true;
        currentRow = rowf;
        currentCol = colf;
    }
//------------------------------------------------------------------------------
  //Properly add piece into column 5
    else if(colFive.isSelected() && board[4][0] == 'y'){
            int colf = 4;
            int rowf = 0;

          //Checks to see if there are other pieces in the column
            while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
                    rowf < 5){
                rowf ++;
            }
```

```java

                        //if there is one piece in the column then it will go to the
                        // place just before it
                          if(rowf == 5 && (board[colf][rowf] == 'r' ||
                                  board[colf][rowf] == 'b')){
                              rowf --;
                              if(playerTurn){
                                  panel[colf][rowf].setBackground(Color.RED);
                                  board[colf][rowf] = 'r';
                                          }
                              else{
                                  panel[colf][rowf].setBackground(Color.BLUE);
                                  board[colf][rowf] = 'b';
                              }

                          }
                        //if there are no pieces in the column then it will go to the
                        // bottom of the column
                          else if(rowf == 5){
                              if(playerTurn){
                                  panel[colf][rowf].setBackground(Color.RED);
                                  board[colf][rowf] = 'r';
                                          }
                              else{
                                  panel[colf][rowf].setBackground(Color.BLUE);
                                  board[colf][rowf] = 'b';
                              }
                          }
                        //it will fill in to any other position that isn't the bottom
                        // or the space just below the bottom
                          else{
                              rowf --;
                              if(playerTurn){
                                  panel[colf][rowf].setBackground(Color.RED);
                                  board[colf][rowf] = 'r';
                                          }
                              else{
                                  panel[colf][rowf].setBackground(Color.BLUE);
                                  board[colf][rowf] = 'b';
                              }
                          }

                      //checks to see if a valid move has been played
                      validMove = true;
                      currentRow = rowf;
                      currentCol = colf;
                  }
              //-------------------------------------------------------------------------------
                //Properly add piece into column 6
                  else if(colSix.isSelected() && board[5][0] == 'y'){
                          int colf = 5;
                          int rowf = 0;

                        //Checks to see if there are other pieces in the column
                          while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
                                  rowf < 5){
                              rowf ++;
                          }

                        //if there is one piece in the column then it will go to the
                        // place just before it
                          if(rowf == 5 && (board[colf][rowf] == 'r' ||
                                  board[colf][rowf] == 'b')){
                              rowf --;
                              if(playerTurn){
                                  panel[colf][rowf].setBackground(Color.RED);
                                  board[colf][rowf] = 'r';
                                          }
                              else{
                                  panel[colf][rowf].setBackground(Color.BLUE);
                                  board[colf][rowf] = 'b';
                              }

                          }
                        //if there are no pieces in the column then it will go to the
                        // bottom of the column
                          else if(rowf == 5){
                              if(playerTurn){
                                  panel[colf][rowf].setBackground(Color.RED);
                                  board[colf][rowf] = 'r';
                                          }
                              else{
                                  panel[colf][rowf].setBackground(Color.BLUE);
                                  board[colf][rowf] = 'b';
                              }
                          }
                        //it will fill in to any other position that isn't the bottom
                        // or the space just below the bottom
                          else{
                              rowf --;
                              if(playerTurn){
                                  panel[colf][rowf].setBackground(Color.RED);
                                  board[colf][rowf] = 'r';
                                          }
                              else{
                                  panel[colf][rowf].setBackground(Color.BLUE);
                                  board[colf][rowf] = 'b';
                              }
                          }
                      //checks to see if a valid move has been played
                      validMove = true;
                      currentRow = rowf;
                      currentCol = colf;
                  }
              //-------------------------------------------------------------------------------
                //Properly add piece into column 7
                  else if(colSeven.isSelected() && board[6][0] == 'y'){
                          int colf = 6;
                          int rowf = 0;

                        //Checks to see if there are other pieces in the column
                          while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
                                  rowf < 5){
                              rowf ++;
                          }

                        //if there is one piece in the column then it will go to the
                        // place just before it
                        if(rowf == 5 && (board[colf][rowf] == 'r' ||
                                board[colf][rowf] == 'b')){
                            rowf --;
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }

                        }
                      //if there are no pieces in the column then it will go to the
                      // bottom of the column
                        else if(rowf == 5){
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }
                        }
                      //it will fill in to any other position that isn't the bottom
                      // or the space just below the bottom
                        else{
                            rowf --;
                            if(playerTurn){
                                panel[colf][rowf].setBackground(Color.RED);
                                board[colf][rowf] = 'r';
                                        }
                            else{
                                panel[colf][rowf].setBackground(Color.BLUE);
                                board[colf][rowf] = 'b';
                            }
                        }

                    //checks to see if a valid move has been played
                    validMove = true;
                    currentRow = rowf;
                    currentCol = colf;
                }
    //-----------------------------------------------------------------------------
              // outputs an error if the column is full
              else{
                  JOptionPane.showMessageDialog(null, "Please select a column\n"
                          + "that isn't full",
                      "Error",
                      JOptionPane.ERROR_MESSAGE);
              }

          //outputs an error if no option has been selected for a column
          }
          else{
              JOptionPane.showMessageDialog(null, "Please select a column",
                      "Error",
                      JOptionPane.ERROR_MESSAGE);
```



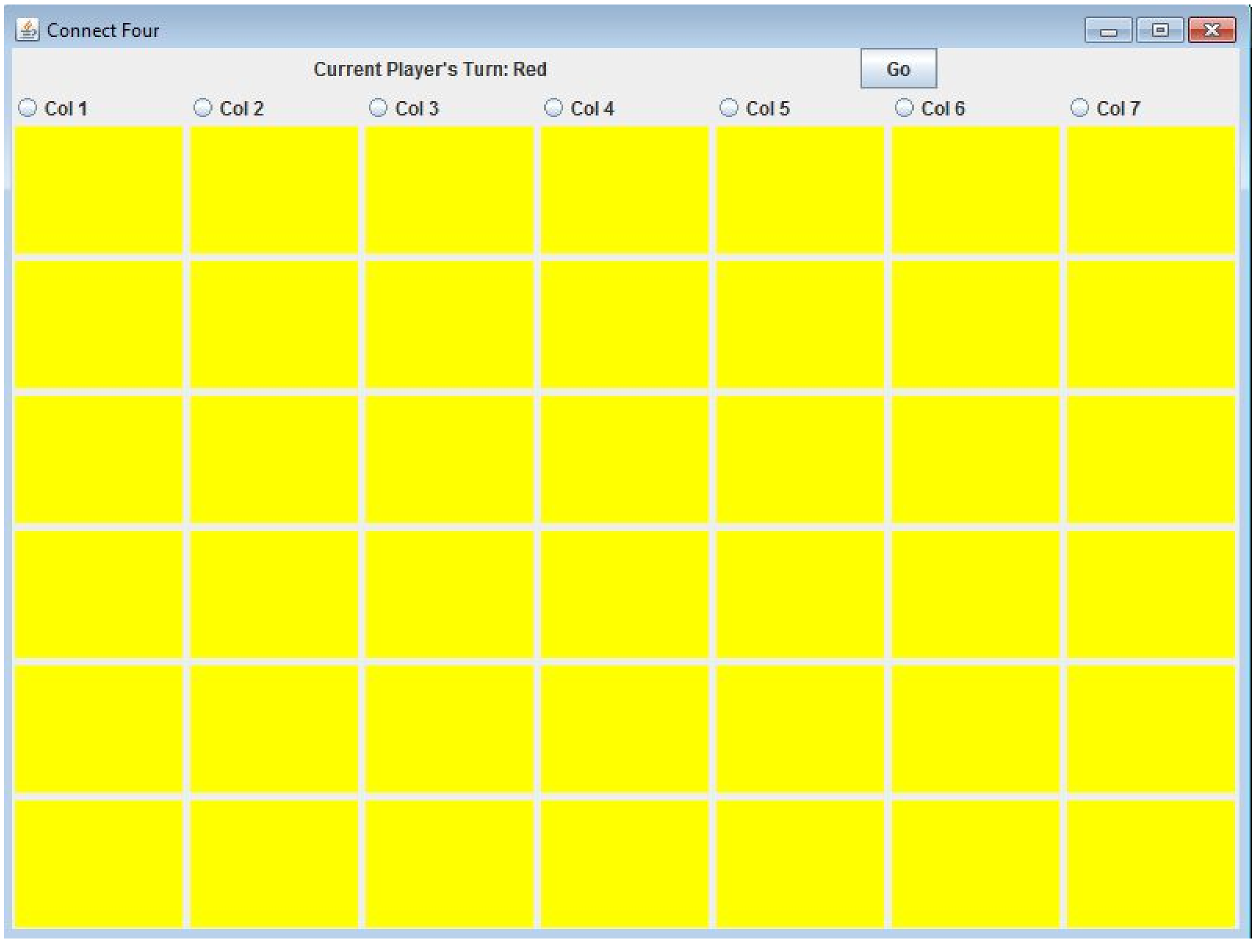
```
            }


            //this checks to see if the piece has activated a winning sequence
            if(validMove) {
                group.clearSelection();
                //currentRow is the current row position for the valid move
                //currentCol is the current column position for the valid move

                //check horizontal win
                char tempTurn;
                int tempCount = 0;
                int tempCurrentRow = currentRow;
                int tempCurrentCol = currentCol - 3;

                if(playerTurn) {
                    tempTurn = 'r';
                }
                else {
                    tempTurn = 'b';
                }

                for(int i = 0; i <7; ++i) {

                    if(tempCurrentRow >= 0 && tempCurrentRow < 6 && tempCurrentCol >= 0
                            && tempCurrentCol < 7) {
                        if(board[tempCurrentCol][tempCurrentRow] == tempTurn) {
                            ++tempCount;
                        }
                        else {
                            tempCount = 0;
                        }
                    }
                    else {
                        tempCount = 0;
                    }

                    ++ tempCurrentCol;

                    if(tempCount == 4) {
                        checkWin = true;
                    }
                }

                //check vertical win
                tempCount = 0;
                tempCurrentRow = currentRow - 3;
                tempCurrentCol = currentCol;
                for(int i = 0; i <7; ++i) {

                    if(tempCurrentRow >= 0 && tempCurrentRow < 6 && tempCurrentCol >= 0
                            && tempCurrentCol < 7) {
                        if(board[tempCurrentCol][tempCurrentRow] == tempTurn) {
                            ++tempCount;
                        }
                        else {
                            tempCount = 0;
                        }
                    }
                    else {
                        tempCount = 0;
                    }

                    ++ tempCurrentRow;

                    if(tempCount == 4) {
                        checkWin = true;
                    }
                }

                //check backward diagonal win [ \ ]
                tempCount = 0;
                tempCurrentRow = currentRow - 3;
                tempCurrentCol = currentCol - 3;
                for(int i = 0; i <7; ++i) {

                    if(tempCurrentRow >= 0 && tempCurrentRow < 6 && tempCurrentCol >= 0
                            && tempCurrentCol < 7) {
                        if(board[tempCurrentCol][tempCurrentRow] == tempTurn) {
                            ++tempCount;
                        }
                        else {
                            tempCount = 0;
                        }
                    }
                    else {
                        tempCount = 0;
                    }

                    ++ tempCurrentRow;
                    ++ tempCurrentCol;

                    if(tempCount == 4) {
                        checkWin = true;
                    }
                }

                //check forward diagonal win [ / ]
                tempCount = 0;
                tempCurrentRow = currentRow - 3;
                tempCurrentCol = currentCol + 3;
                for(int i = 0; i <7; ++i) {

                    if(tempCurrentRow >= 0 && tempCurrentRow < 6 && tempCurrentCol >= 0 && tempCurrentCol < 7) {
                        if(board[tempCurrentCol][tempCurrentRow] == tempTurn) {
                            ++tempCount;
                        }
                        else {
                            tempCount = 0;
                        }
                    }
                    else {
                        tempCount = 0;
                    }

                    ++ tempCurrentRow;
                    -- tempCurrentCol;

```

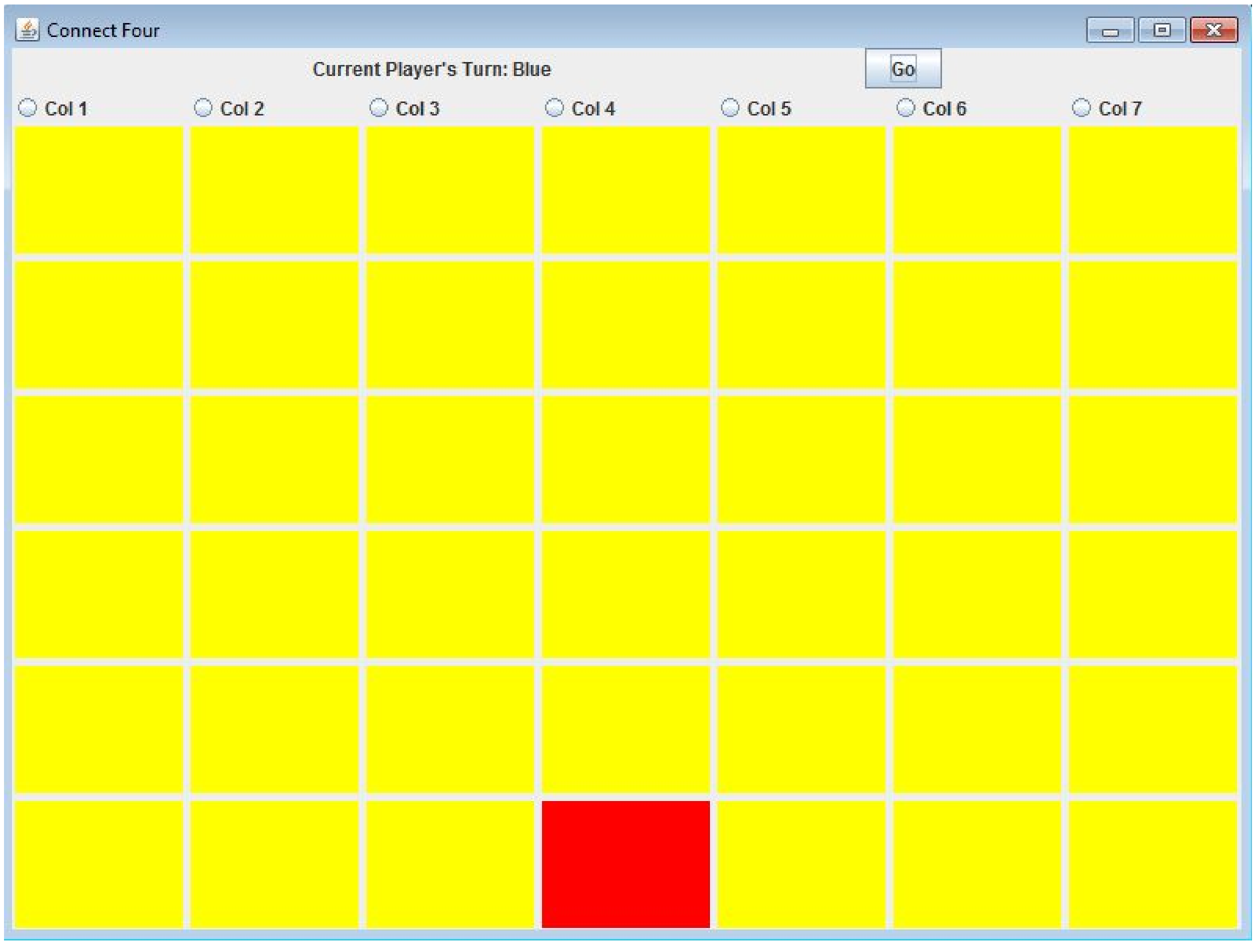
```java
                    if(tempCount == 4) {
                        checkWin = true;
                    }
                }


                //the current player will be swapped and the label will update
                if(playerTurn) {
                    playerTurn = false;
                    turn.setText("Current Player's Turn: Blue");
                }
                else {
                    playerTurn = true;
                    turn.setText("Current Player's Turn: Red");
                }
                //resets the valid move for the next move
                validMove = false;
            }
            //checks to see if a draw case has happened only if a win case has not already been triggered
            if(!(board[0][0] == 'y' || board[1][0] == 'y' || board[2][0] == 'y' || board[3][0] == 'y' ||
                    board[4][0] == 'y' || board[5][0] == 'y' ||
                    board[6][0] == 'y') && !checkWin) {
                checkDraw = true;
            }

            //handles the win case
            if(checkWin) {
                //removes the buttons after a win case shows up
                topGame.remove(buttonsGame);
                //displays that the blue player has won
                if(playerTurn) {
                    JOptionPane.showMessageDialog(null, "Blue player Wins!",
                    "Winner!!",
                    JOptionPane.INFORMATION_MESSAGE);
                    turn.setText("Blue player is the Winner!");
                }
                //displays that the red player has won
                else{
                    JOptionPane.showMessageDialog(null, "Red player Wins!",
                    "Winner!!",
                    JOptionPane.INFORMATION_MESSAGE);
                    turn.setText("Red player is the Winner!");
                }

                // removes the button that triggers a move and replaces it with
                //      a button that resets the game
                turnGame.remove(activate);
                turnGame.add(newGame);

                // fixes bug where program wouldn't load the restart button after first game resolution
                turnGame.revalidate();
            }
            else if(checkDraw) {
                //removes the buttons after a draw case
                topGame.remove(buttonsGame);
                //displays that a draw has occurred
                JOptionPane.showMessageDialog(null, "It's a Draw!",
                        "Tie Game",
                        JOptionPane.INFORMATION_MESSAGE);
                turn.setText("The game ends in a Draw");
                // removes the button that triggers a move and replaces it with
                //      a button that resets the game
                turnGame.remove(activate);
                turnGame.add(newGame);
                // fixes bug where program wouldn't load the restart button after first game resolution
                turnGame.revalidate();
            }
        }
        else if(source == newGame) {
            //goes back to the home screen
            cardLayout.previous(getContentPane());
            board = new char[COLS][ROWS];// resets the char array
            panel = new JPanel[COLS][ROWS];// resets the pannel array
            bottomGame.removeAll();// removes all of the old pannels
            topGame.add(buttonsGame);// re adds the radio buttons
            turnGame.add(activate);// re adds the button to activate a player's turn
            turnGame.remove(newGame); // removes the reset button
            checkDraw = false; // resets the check for draw
            checkWin = false; // resets the check for win

            // resets the starting player to red player
            playerTurn = true;
            turn.setText("Current Player's Turn: Red");
        }
}


    public static void main(String[] args) {
        ConnectFour game = new ConnectFour();
        game.setSize(800,600);
        game.setVisible(true);
    }

}
```

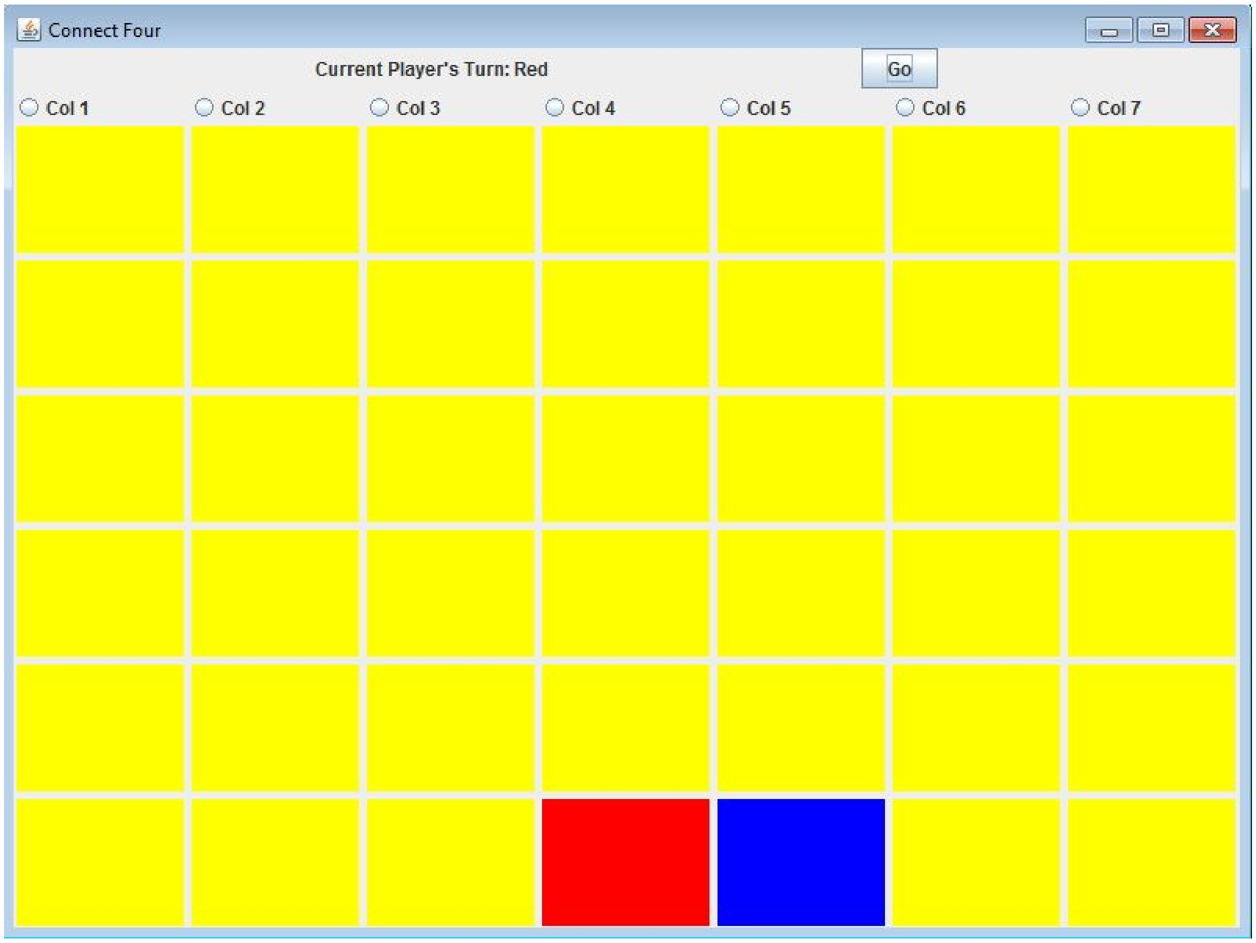


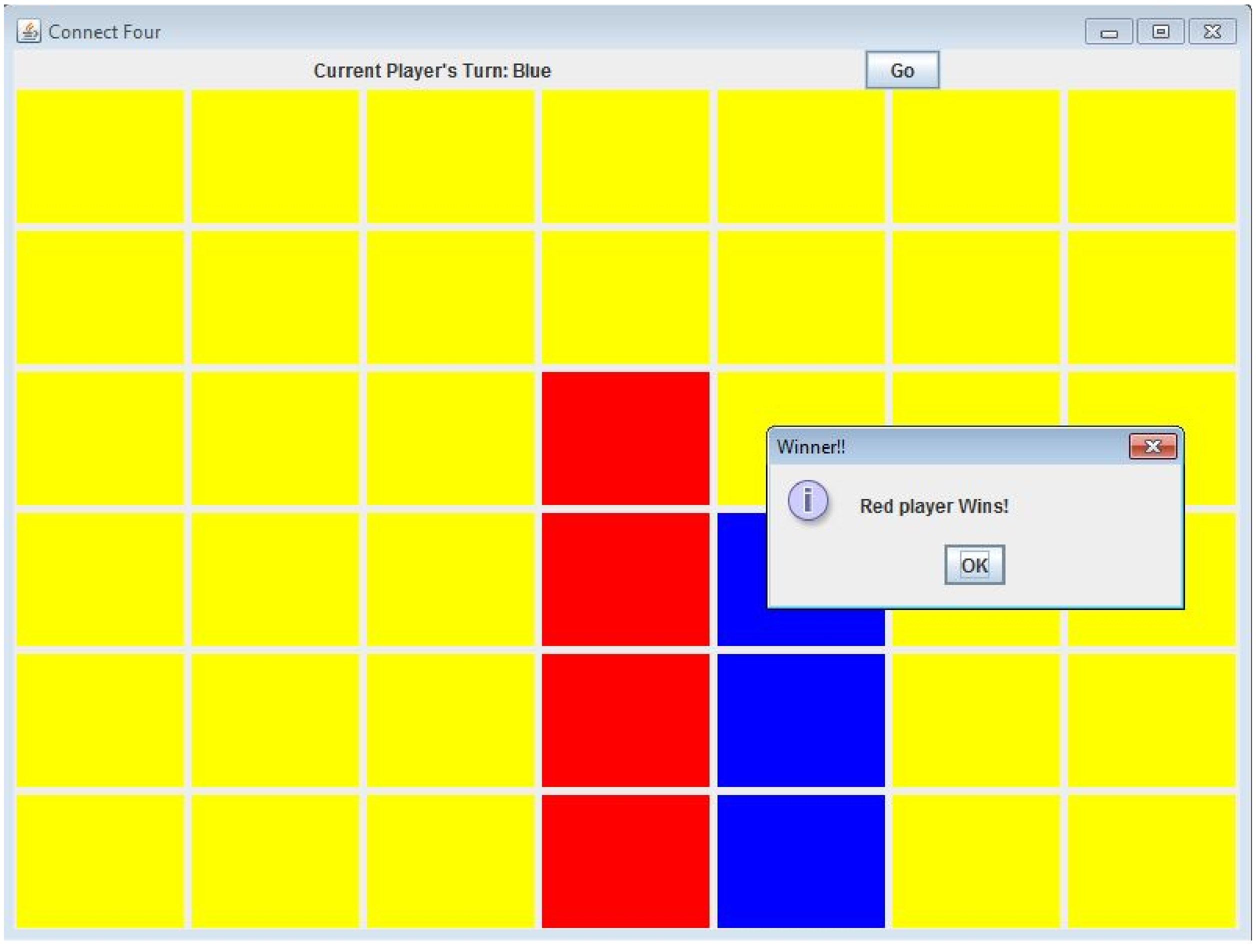
Welcome to my Connect Four game

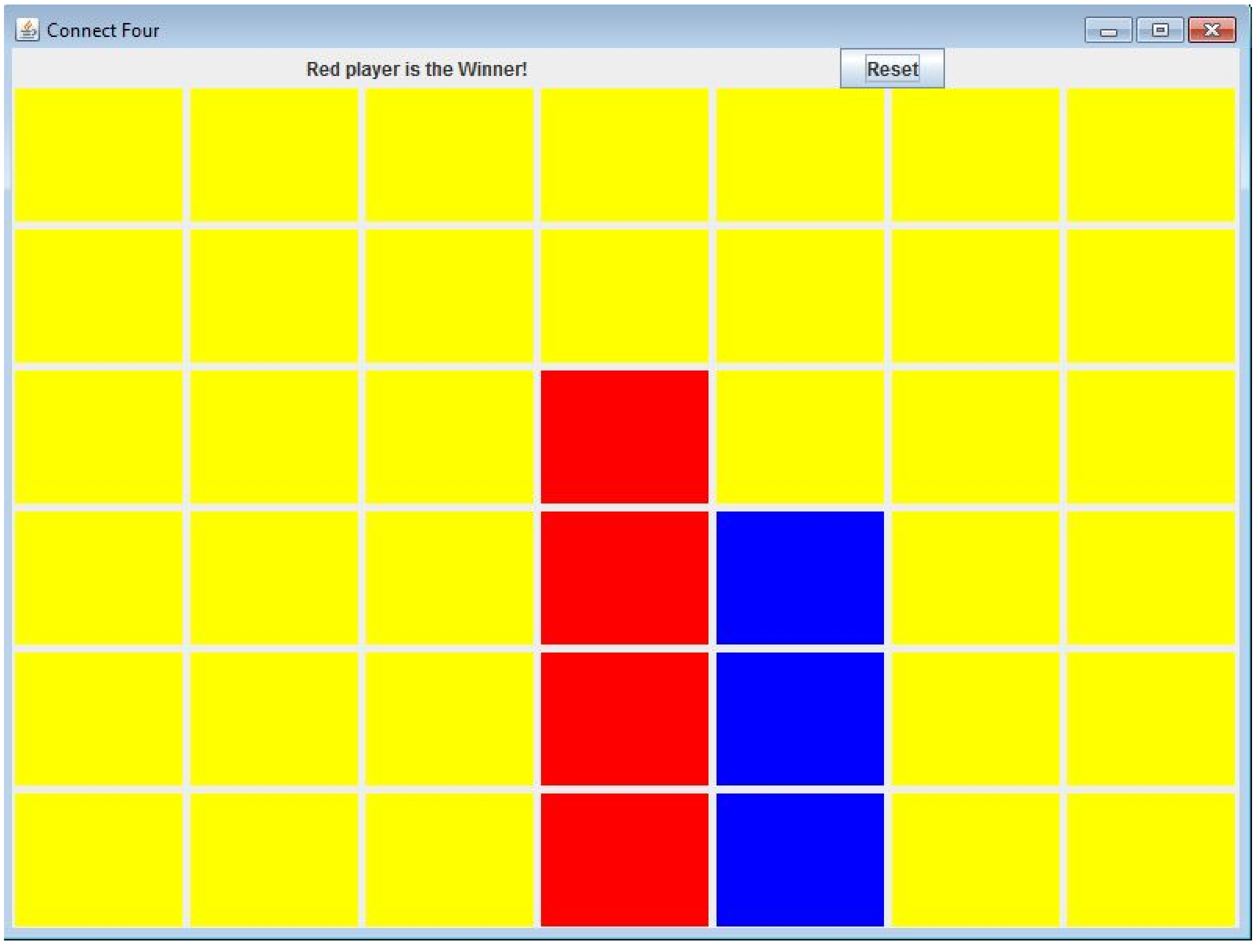
please click the button to start


Connect Four
Current Player's Turn: Red
Go
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7

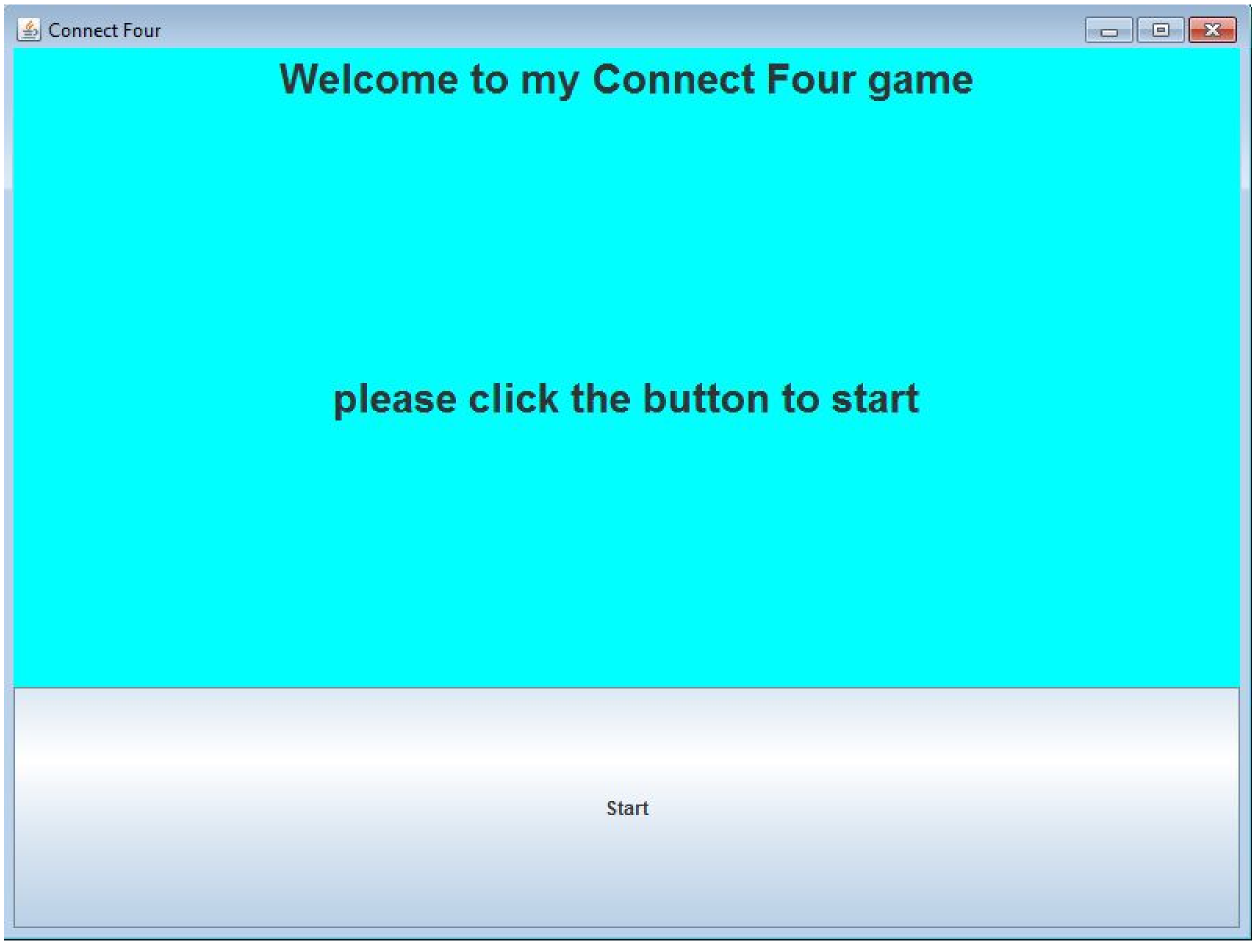

Connect Four
Current Player's Turn: Blue
Go
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7


Connect Four
Current Player's Turn: Red
Go
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7


Connect Four
Current Player's Turn: Blue
Go
Winner!!
Red player Wins!
OK

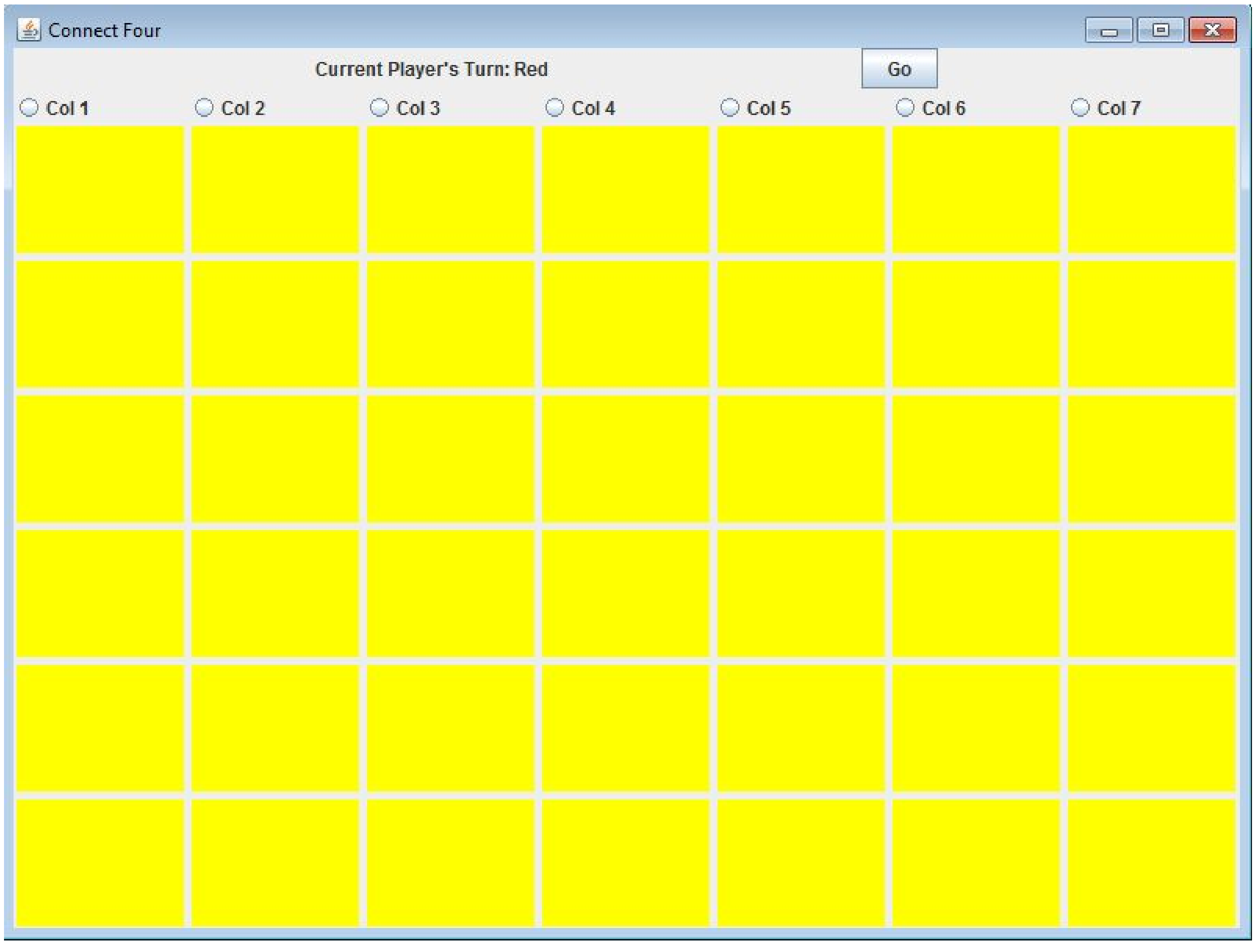


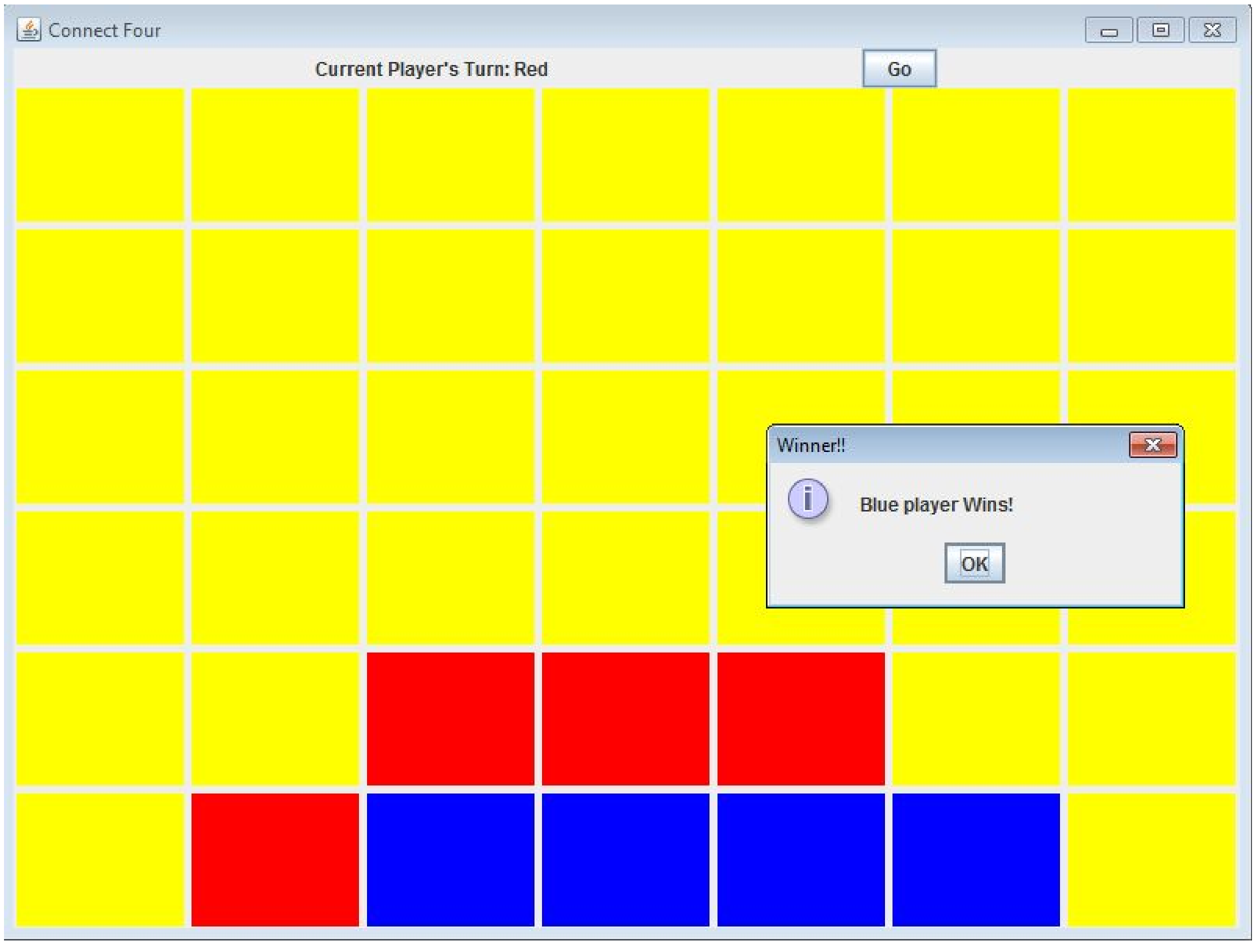
Red player is the Winner!

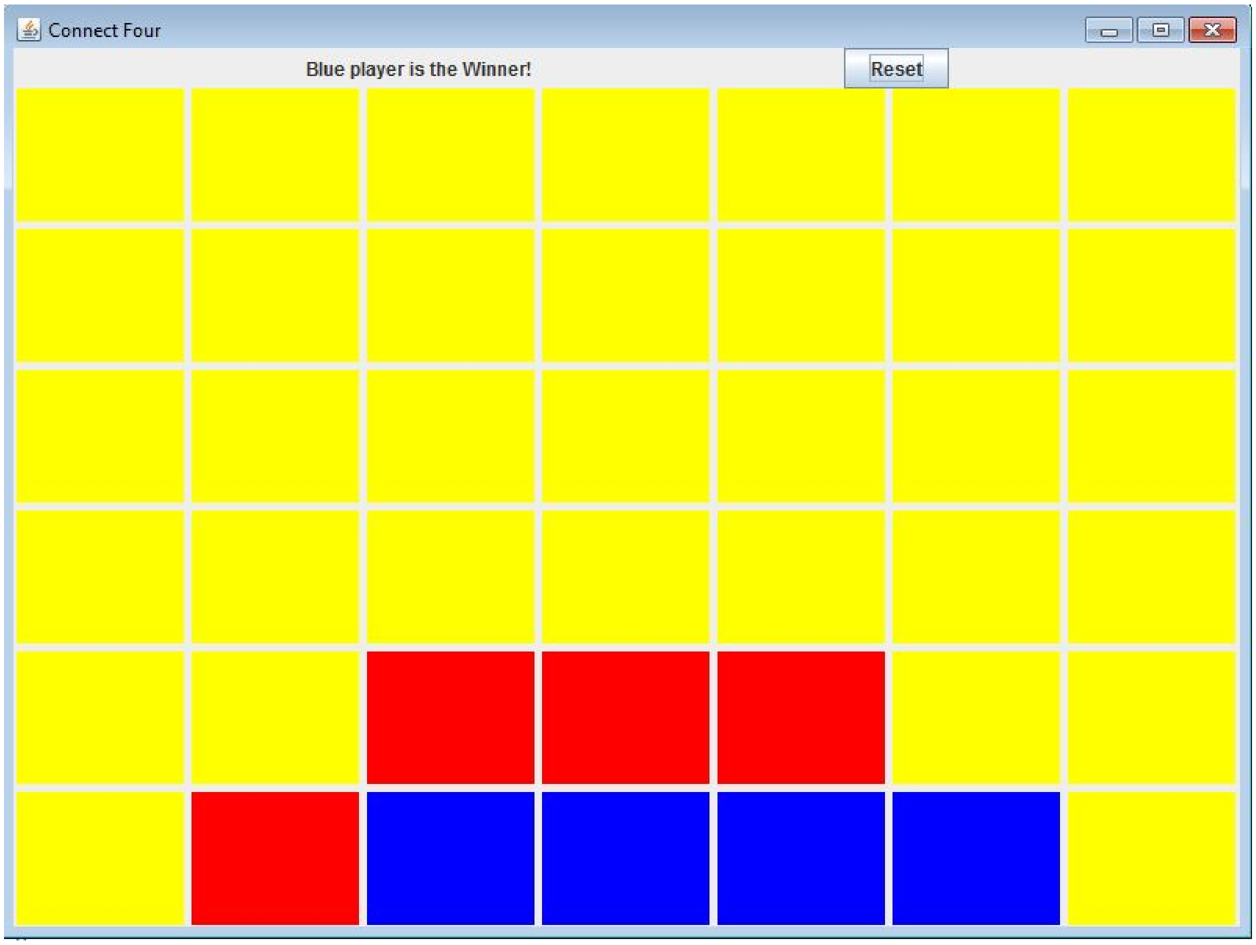


Welcome to my Connect Four game

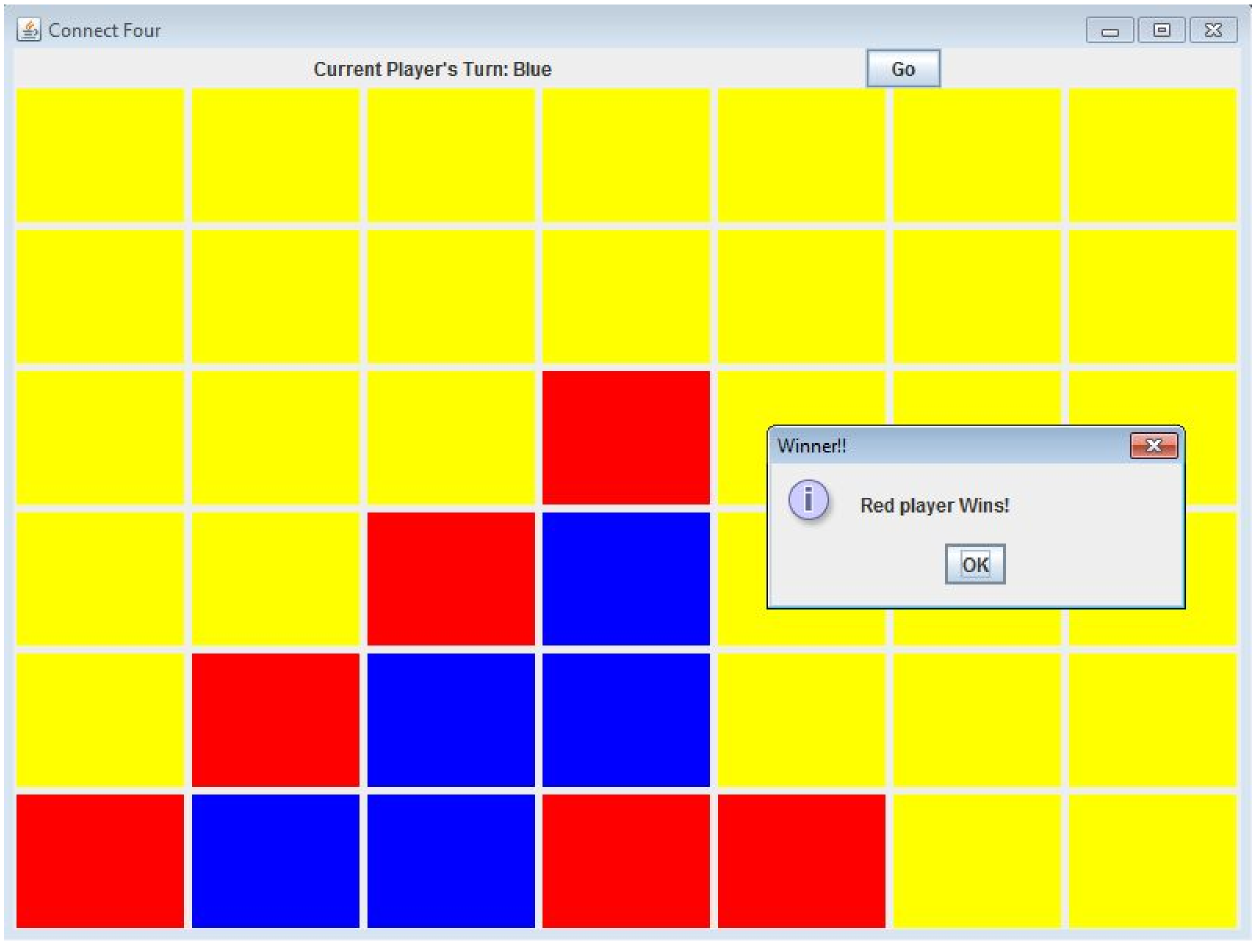
please click the button to start


Connect Four
Current Player's Turn: Red
Go
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7


Connect Four
Current Player's Turn: Red
Go
Winner!!
Blue player Wins!
OK



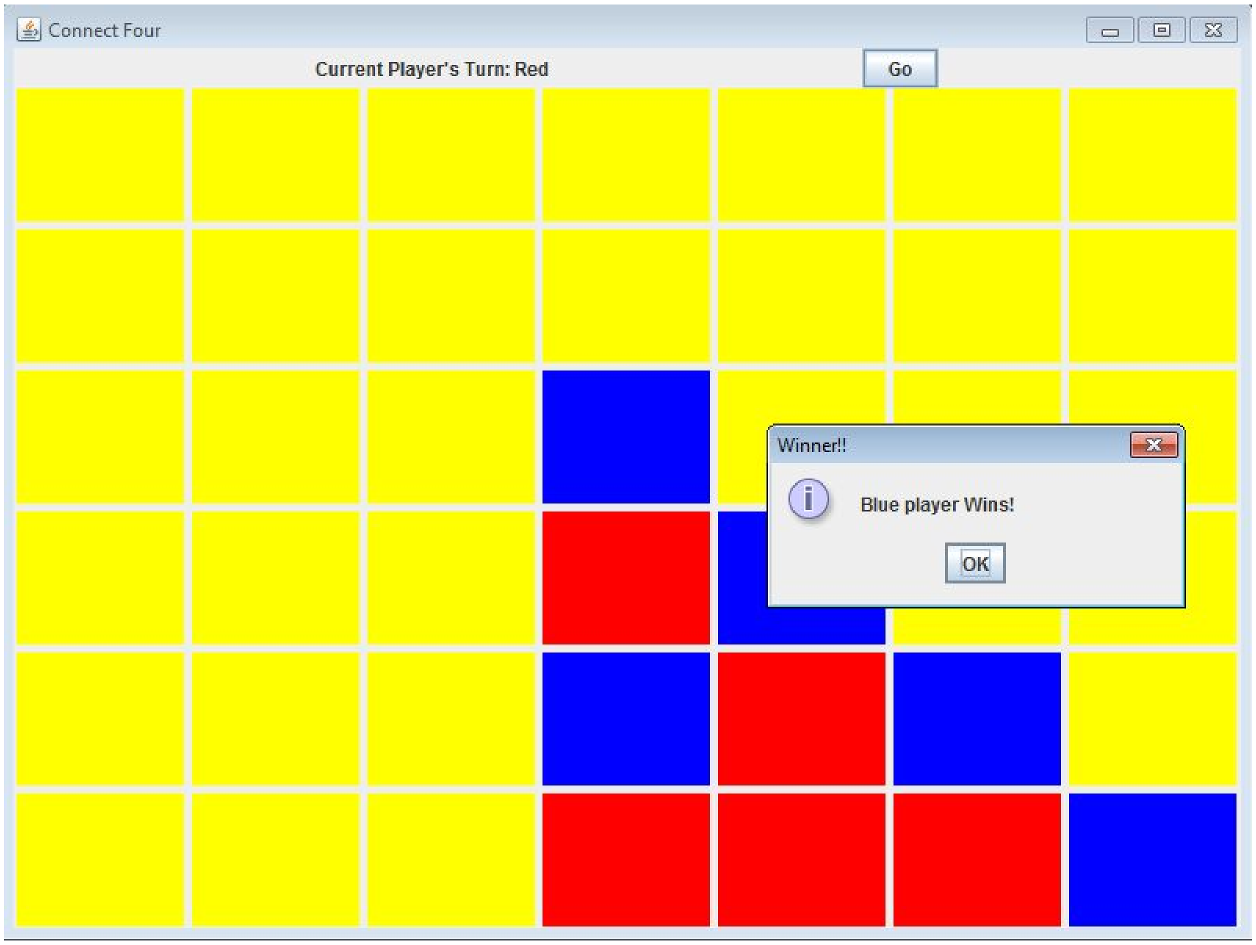
Blue player is the Winner!



Current Player's Turn: Blue

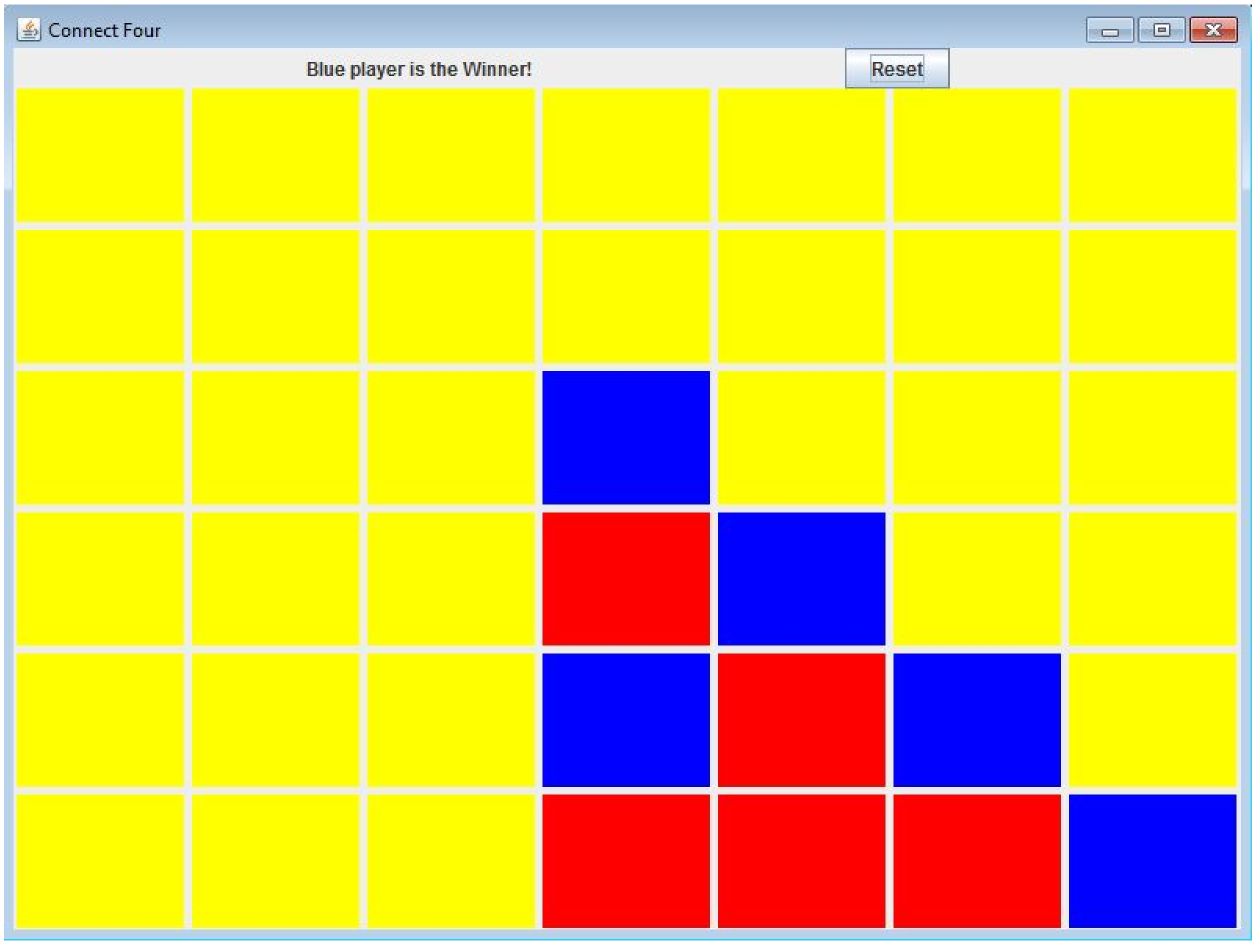
Winner!!

Red player Wins!

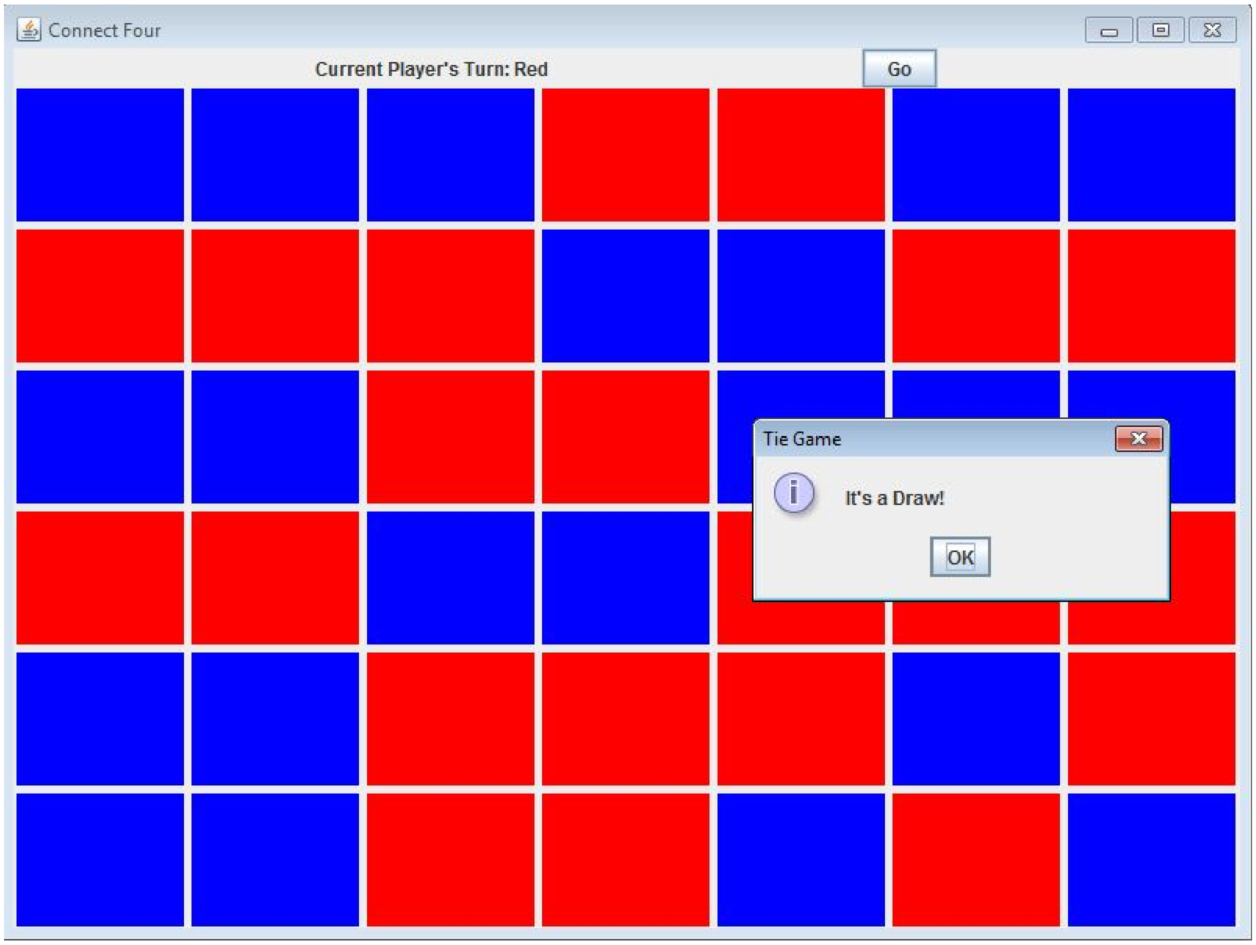


Current Player's Turn: Red

Blue player Wins!

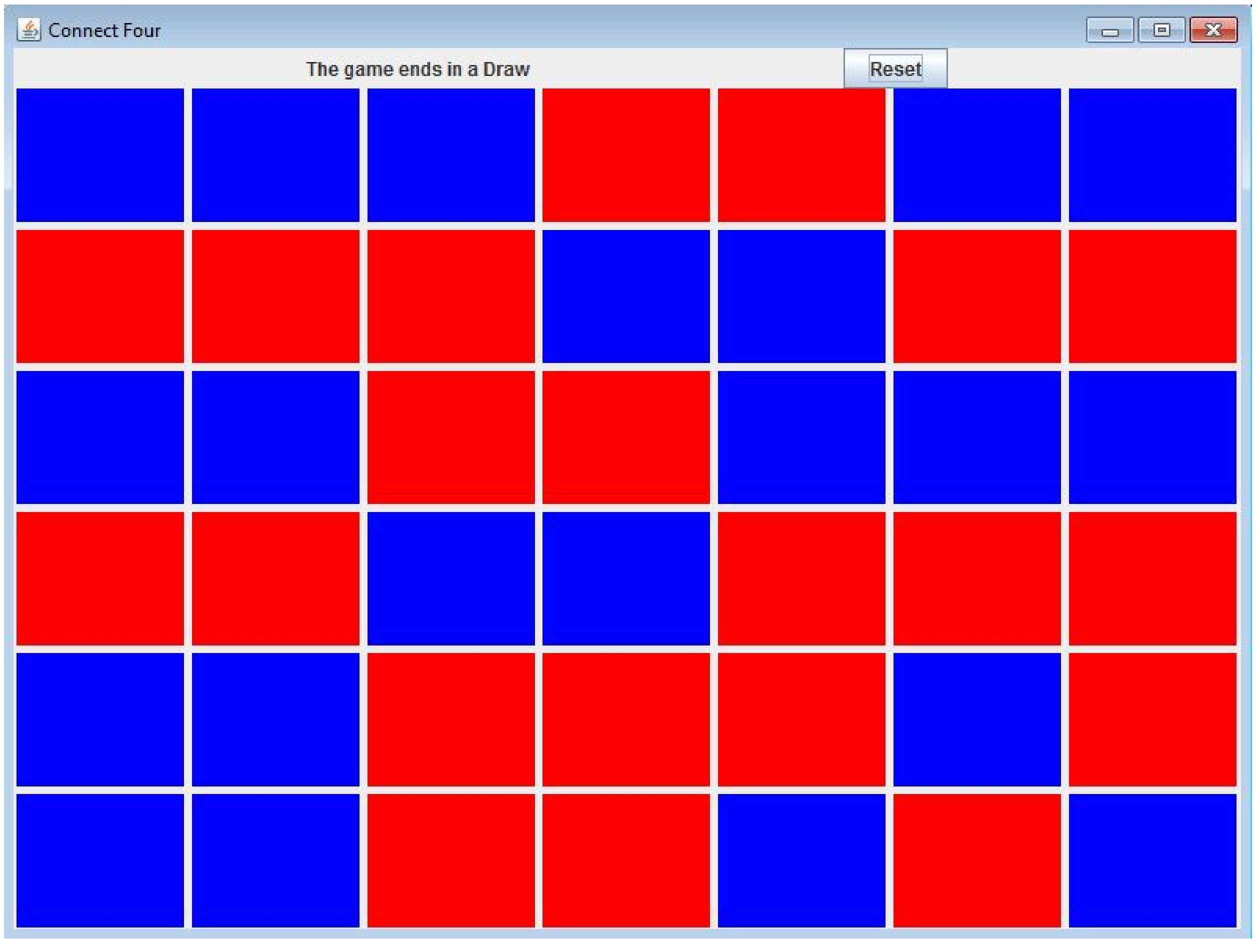


Blue player is the Winner!

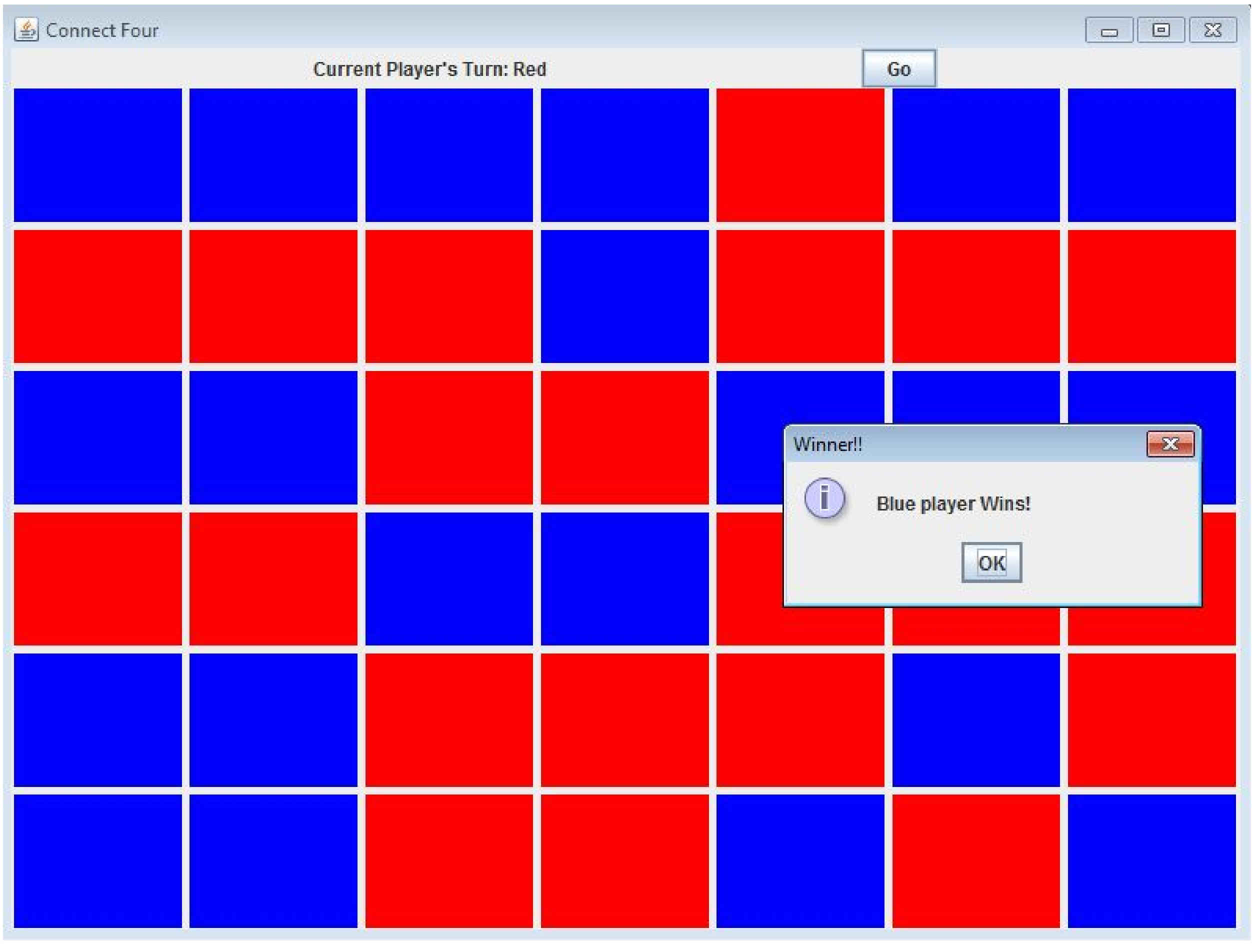


Current Player's Turn: Red

It's a Draw!

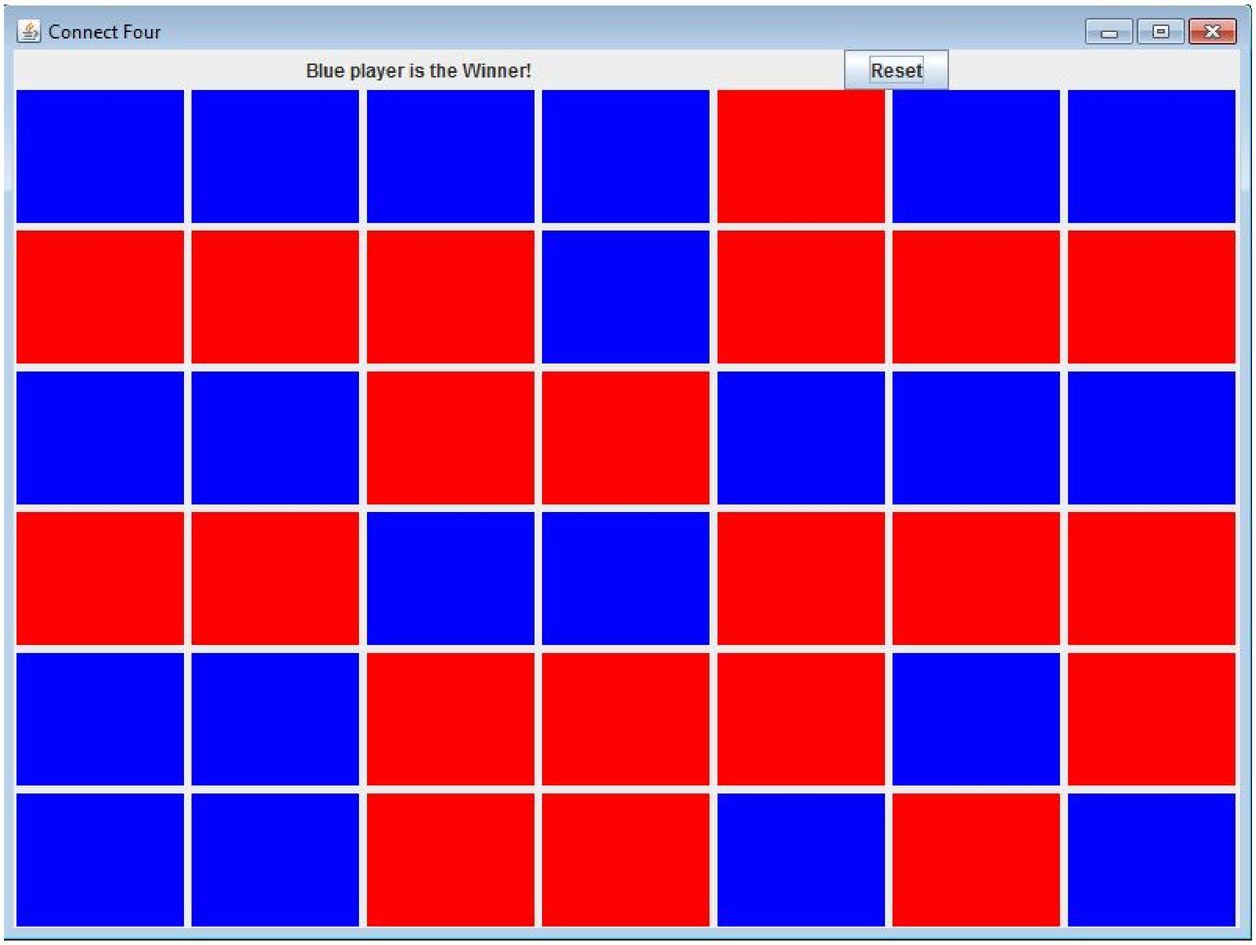


The game ends in a Draw



Current Player's Turn: Red

Blue player Wins!



Blue player is the Winner!