```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");
```

```
60      private JRadioButton colSeven = new JRadioButton("Col 7");
61
62
63      private JLabel turn = new JLabel("Current Player's Turn: Red"); // initializes the label for turn
64      private boolean playerTurn = true; // sets the active players turn - true is red - false is blue
65      private JButton activate = new JButton("Go"); // button for a player to input their turn
66      private JButton newGame = new JButton("Reset"); // button to reset the game after it has been finished
67
68      // labels for the home screen display
69      private JLabel intro = new JLabel("Welcome to my Connect Four game");
70      private JLabel intro2 = new JLabel("please click the button to start");
71
72      public ConnectFour(){
73          super("Connect Four"); // names the program window
74          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // closes program on exit of window
75          setLayout(cardLayout); // adds the card layout for the multiple windows to be changed between
76
77          //formatting for the first card - the home screen
78          home.setBackground(Color.CYAN);
79          home.setLayout(new GridLayout(3,0,50,50));
80          intro.setFont(new Font("Arial", Font.BOLD, 26));
81          intro2.setFont(new Font("Arial", Font.BOLD, 26));
82          topHome.setLayout(new FlowLayout());
83          topHome.add(intro);
84          topHome.setBackground(Color.CYAN);
85          middleHome.setLayout(new FlowLayout());
86          middleHome.add(intro2);
87          middleHome.setBackground(Color.CYAN);
88          home.add(topHome);
89          home.add(middleHome);
90          home.add(next);
91
92          //formatting for the second card - the game screen
93          group.add(colOne);
94          group.add(colTwo);
95          group.add(colThree);
96          group.add(colFour);
97          group.add(colFive);
98          group.add(colSix);
99          group.add(colSeven);
100         gameScreen.setLayout(new BorderLayout());
101         turnGame.setLayout(new FlowLayout(FlowLayout.CENTER, 200,0));
102         topGame.setLayout(new BorderLayout());
103         turnGame.add(turn);
104         turnGame.add(activate);
105         buttonsGame.setLayout(new GridLayout(1,7,0,100));
106         bottomGame.setLayout(new GridLayout(6,7,5,5));
107         topGame.add(turnGame, BorderLayout.NORTH);
108         topGame.add(buttonsGame, BorderLayout.CENTER);
109         gameScreen.add(topGame, BorderLayout.NORTH);
110
111         // adds the buttons to the button bar
112         buttonsGame.add(colOne);
113         buttonsGame.add(colTwo);
114         buttonsGame.add(colThree);
115         buttonsGame.add(colFour);
116         buttonsGame.add(colFive);
117         buttonsGame.add(colSix);
118         buttonsGame.add(colSeven);
```

```
119         // formats the header above the game
120         gameScreen.add(topGame, BorderLayout.NORTH);
121         // adds the game to the game screen
122         gameScreen.add(bottomGame, BorderLayout.CENTER);
123 
124         // adds action listeners to all buttons that are going to be used in the program
125         activate.addActionListener(this);
126         next.addActionListener(this);
127         newGame.addActionListener(this);
128         colOne.addActionListener(this);
129         colTwo.addActionListener(this);
130         colThree.addActionListener(this);
131         colFour.addActionListener(this);
132         colFive.addActionListener(this);
133         colSix.addActionListener(this);
134         colSeven.addActionListener(this);
135 
136         // adds the different cards to the card layout
137         add("home", home);
138         add("game", gameScreen);
139 
140     }
141 
142 
143     @Override
144     public void actionPerformed(ActionEvent e) {
145         Object source = e.getSource();
146         // the button that goes from home screen to the game
147         if(source == next){
148             // initializes the arrays that are used in the board so they are all neutral
149             for(int y = 0; y < ROWS; ++y){
150                 for(int x = 0; x < COLS; ++x){
151                     board[x][y] = 'y';
152                     panel[x][y] = new JPanel();
153                     bottomGame.add(panel[x][y]);
154                     panel[x][y].setBackground(Color.YELLOW);
155 
156             }
157         }
158             // swaps to the game card
159             cardLayout.next(getContentPane());
160         }
161         else if(source == activate){
162             //checks to see if one of the columns is selected
163             if(colOne.isSelected() || colTwo.isSelected() ||
164                colThree.isSelected() || colFour.isSelected() ||
165                colFive.isSelected() || colSix.isSelected() ||
166                colSeven.isSelected()){
167                 //-------------------------------------------------------------------------
168                 //entering pieces into the program if they can fit
169                 if(colOne.isSelected() && board[0][0] == 'y'){
170                     int colf = 0;
171                     int rowf = 0;
172 
173 
174                     //Checks to see if there are other pieces in the column
175                     while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
176                             rowf < 5){
```

```
177
178                     rowf ++;
179                 }
180                 //if there is one piece in the column then it will go to the
181                 // place just before it
182                 if(rowf == 5 && (board[colf][rowf] == 'r' ||
183                         board[colf][rowf] == 'b')){
184                     rowf --;
185                     if(playerTurn){
186                         panel[colf][rowf].setBackground(Color.RED);
187                         board[colf][rowf] = 'r';
188                                 }
189                     else{
190                         panel[colf][rowf].setBackground(Color.BLUE);
191                         board[colf][rowf] = 'b';
192                     }
193
194                 }
195                 //if there are no pieces in the column then it will go to the
196                 // bottom of the column
197                 else if(rowf == 5){
198                     if(playerTurn){
199                         panel[colf][rowf].setBackground(Color.RED);
200                         board[colf][rowf] = 'r';
201                                 }
202                     else{
203                         panel[colf][rowf].setBackground(Color.BLUE);
204                         board[colf][rowf] = 'b';
205                     }
206                 }
207                 //it will fill in to any other position that isn't the bottom
208                 // or the space just below the bottom
209                 else{
210                     rowf --;
211                     if(playerTurn){
212                         panel[colf][rowf].setBackground(Color.RED);
213                         board[colf][rowf] = 'r';
214                                 }
215                     else{
216                         panel[colf][rowf].setBackground(Color.BLUE);
217                         board[colf][rowf] = 'b';
218                     }
219                 }
220
221
222                 //checks to see if a valid move has been played
223                 validMove = true;
224                 currentRow = rowf;
225                 currentCol = colf;
226               }
227             //-----------------------------------------------------------------------------
228               //Properly add piece into column 2
229               else if(colTwo.isSelected() && board[1][0] == 'y'){
230                       int colf = 1;
231                       int rowf = 0;
232
233                     //Checks to see if there are other pieces in the column
234                       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 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){
```

```
                        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 ++;
```

```
353                         }
354
355                         //if there is one piece in the column then it will go to the
356                         // place just before it
357                           if(rowf == 5 && (board[colf][rowf] == 'r' ||
358                                   board[colf][rowf] == 'b')){
359                               rowf --;
360                               if(playerTurn){
361                                   panel[colf][rowf].setBackground(Color.RED);
362                                   board[colf][rowf] = 'r';
363                                           }
364                               else{
365                                   panel[colf][rowf].setBackground(Color.BLUE);
366                                   board[colf][rowf] = 'b';
367                               }
368
369                           }
370                         //if there are no pieces in the column then it will go to the
371                         // bottom of the column
372                           else if(rowf == 5){
373                               if(playerTurn){
374                                   panel[colf][rowf].setBackground(Color.RED);
375                                   board[colf][rowf] = 'r';
376                                           }
377                               else{
378                                   panel[colf][rowf].setBackground(Color.BLUE);
379                                   board[colf][rowf] = 'b';
380                               }
381                           }
382                         //it will fill in to any other position that isn't the bottom
383                         // or the space just below the bottom
384                           else{
385                               rowf --;
386                               if(playerTurn){
387                                   panel[colf][rowf].setBackground(Color.RED);
388                                   board[colf][rowf] = 'r';
389                                           }
390                               else{
391                                   panel[colf][rowf].setBackground(Color.BLUE);
392                                   board[colf][rowf] = 'b';
393                               }
394                           }
395
396                        //checks to see if a valid move has been played
397                        validMove = true;
398                        currentRow = rowf;
399                        currentCol = colf;
400                    }
401                //-----------------------------------------------------------------------------
402                  //Properly add piece into column 5
403                    else if(colFive.isSelected() && board[4][0] == 'y'){
404                            int colf = 4;
405                            int rowf = 0;
406
407                          //Checks to see if there are other pieces in the column
408                            while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
409                                    rowf < 5){
410                               rowf ++;
411                            }
```

```
412
413                         //if there is one piece in the column then it will go to the
414                         // place just before it
415                           if(rowf == 5 && (board[colf][rowf] == 'r' ||
416                                   board[colf][rowf] == 'b')){
417                               rowf --;
418                               if(playerTurn){
419                                   panel[colf][rowf].setBackground(Color.RED);
420                                   board[colf][rowf] = 'r';
421                                           }
422                               else{
423                                   panel[colf][rowf].setBackground(Color.BLUE);
424                                   board[colf][rowf] = 'b';
425                               }
426
427                           }
428                         //if there are no pieces in the column then it will go to the
429                         // bottom of the column
430                           else if(rowf == 5){
431                               if(playerTurn){
432                                   panel[colf][rowf].setBackground(Color.RED);
433                                   board[colf][rowf] = 'r';
434                                           }
435                               else{
436                                   panel[colf][rowf].setBackground(Color.BLUE);
437                                   board[colf][rowf] = 'b';
438                               }
439                           }
440                         //it will fill in to any other position that isn't the bottom
441                         // or the space just below the bottom
442                           else{
443                               rowf --;
444                               if(playerTurn){
445                                   panel[colf][rowf].setBackground(Color.RED);
446                                   board[colf][rowf] = 'r';
447                                           }
448                               else{
449                                   panel[colf][rowf].setBackground(Color.BLUE);
450                                   board[colf][rowf] = 'b';
451                               }
452                           }
453
454                       //checks to see if a valid move has been played
455                       validMove = true;
456                       currentRow = rowf;
457                       currentCol = colf;
458                   }
459               //-----------------------------------------------------------------------------
460                 //Properly add piece into column 6
461                   else if(colSix.isSelected() && board[5][0] == 'y'){
462                           int colf = 5;
463                           int rowf = 0;
464
465                         //Checks to see if there are other pieces in the column
466                           while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
467                                   rowf < 5){
468                               rowf ++;
469                           }
470
```

```
471                     //if there is one piece in the column then it will go to the
472                     // place just before it
473                       if(rowf == 5 && (board[colf][rowf] == 'r' ||
474                               board[colf][rowf] == 'b')){
475                           rowf --;
476                           if(playerTurn){
477                               panel[colf][rowf].setBackground(Color.RED);
478                               board[colf][rowf] = 'r';
479                                       }
480                           else{
481                               panel[colf][rowf].setBackground(Color.BLUE);
482                               board[colf][rowf] = 'b';
483                           }
484
485                       }
486                     //if there are no pieces in the column then it will go to the
487                     // bottom of the column
488                       else if(rowf == 5){
489                           if(playerTurn){
490                               panel[colf][rowf].setBackground(Color.RED);
491                               board[colf][rowf] = 'r';
492                                       }
493                           else{
494                               panel[colf][rowf].setBackground(Color.BLUE);
495                               board[colf][rowf] = 'b';
496                           }
497                       }
498                     //it will fill in to any other position that isn't the bottom
499                     // or the space just below the bottom
500                       else{
501                           rowf --;
502                           if(playerTurn){
503                               panel[colf][rowf].setBackground(Color.RED);
504                               board[colf][rowf] = 'r';
505                                       }
506                           else{
507                               panel[colf][rowf].setBackground(Color.BLUE);
508                               board[colf][rowf] = 'b';
509                           }
510                       }
511                   //checks to see if a valid move has been played
512                   validMove = true;
513                   currentRow = rowf;
514                   currentCol = colf;
515               }
516             //-------------------------------------------------------------------------------
517               //Properly add piece into column 7
518                 else if(colSeven.isSelected() && board[6][0] == 'y'){
519                         int colf = 6;
520                         int rowf = 0;
521
522                     //Checks to see if there are other pieces in the column
523                       while((board[colf][rowf] != 'r' && board[colf][rowf] != 'b') &&
524                               rowf < 5){
525                           rowf ++;
526                       }
527
528                     //if there is one piece in the column then it will go to the
529                     // 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);
```

```
589             }
590
591
592             //this checks to see if the piece has activated a winning sequence
593             if(validMove) {
594                 group.clearSelection();
595                 //currentRow is the current row position for the valid move
596                 //currentCol is the current column position for the valid move
597
598                 //check horizontal win
599                 char tempTurn;
600                 int tempCount = 0;
601                 int tempCurrentRow = currentRow;
602                 int tempCurrentCol = currentCol - 3;
603
604                 if(playerTurn) {
605                     tempTurn = 'r';
606                 }
607                 else {
608                     tempTurn = 'b';
609                 }
610
611                 for(int i = 0; i <7; ++i) {
612
613                     if(tempCurrentRow >= 0 && tempCurrentRow < 6 && tempCurrentCol >= 0
614                             && tempCurrentCol < 7) {
615                         if(board[tempCurrentCol][tempCurrentRow] == tempTurn) {
616                             ++tempCount;
617                         }
618                         else {
619                             tempCount = 0;
620                         }
621                     }
622                     else {
623                         tempCount = 0;
624                     }
625
626                     ++ tempCurrentCol;
627
628                     if(tempCount == 4) {
629                         checkWin = true;
630                     }
631                 }
632
633                 //check vertical win
634                 tempCount = 0;
635                 tempCurrentRow = currentRow - 3;
636                 tempCurrentCol = currentCol;
637                 for(int i = 0; i <7; ++i) {
638
639                     if(tempCurrentRow >= 0 && tempCurrentRow < 6 && tempCurrentCol >= 0
640                             && tempCurrentCol < 7) {
641                         if(board[tempCurrentCol][tempCurrentRow] == tempTurn) {
642                             ++tempCount;
643                         }
644                         else {
645                             tempCount = 0;
646                         }
647                     }
```

```
                    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;
```

```
                    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",
```

```
765                         JOptionPane.INFORMATION_MESSAGE);
766                 turn.setText("The game ends in a Draw");
767                 // removes the button that triggers a move and replaces it with
768                 //      a button that resets the game
769                 turnGame.remove(activate);
770                 turnGame.add(newGame);
771                 // fixes bug where program wouldn't load the restart button after first game resolution
772                 turnGame.revalidate();
773             }
774         }
775         else if(source == newGame) {
776             //goes back to the home screen
777             cardLayout.previous(getContentPane());
778             board = new char[COLS][ROWS];// resets the char array
779             panel = new JPanel[COLS][ROWS];// resets the pannel array
780             bottomGame.removeAll();// removes all of the old pannels
781             topGame.add(buttonsGame);// re adds the radio buttons
782             turnGame.add(activate);// re adds the button to activate a player's turn
783             turnGame.remove(newGame); // removes the reset button
784             checkDraw = false; // resets the check for draw
785             checkWin = false; // resets the check for win
786 
787             // resets the starting player to red player
788             playerTurn = true;
789             turn.setText("Current Player's Turn: Red");
790         }
791 }
792 
793 
794     public static void main(String[] args) {
795         ConnectFour game = new ConnectFour();
796         game.setSize(800,600);
797         game.setVisible(true);
798     }
799 
800 }
```


Connect Four
Welcome to my Connect Four game
please click the button to start
Start


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

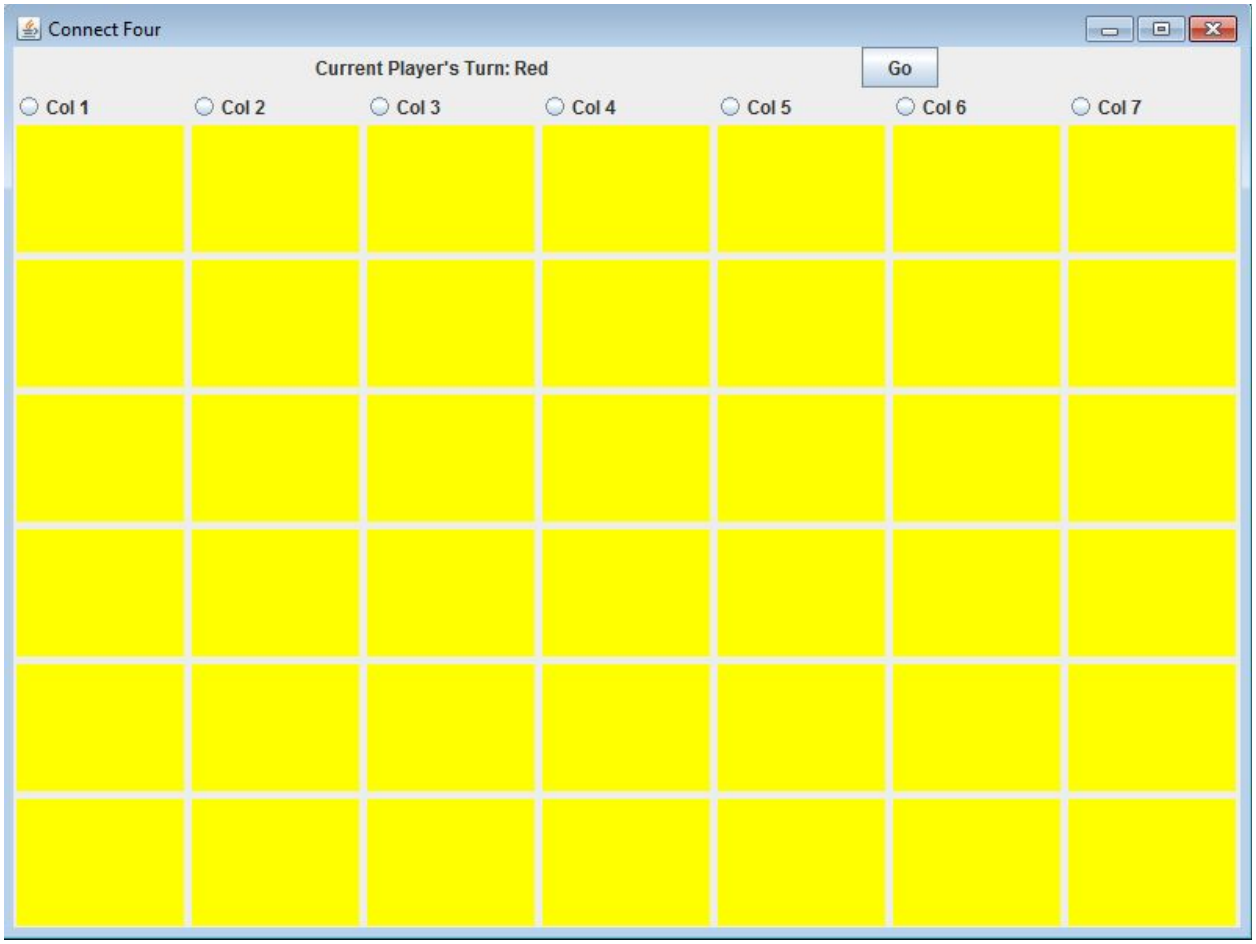

Connect Four
Current Player's Turn: Blue
Go
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7


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

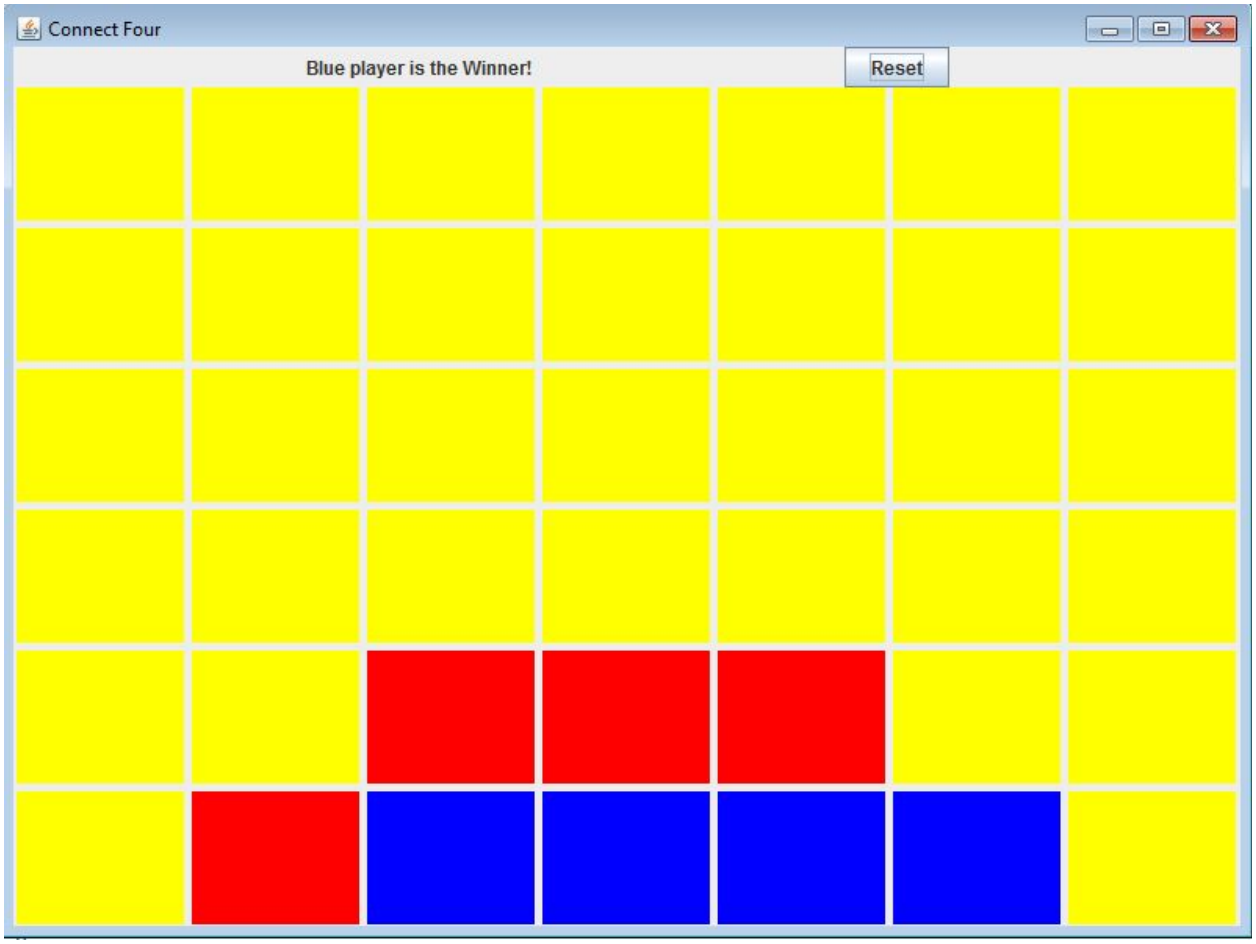

Connect Four
Current Player's Turn: Blue
Go
Winner!!
Red player Wins!
OK


Connect Four
Red player is the Winner!
Reset


Connect Four
Welcome to my Connect Four game
please click the button to start
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

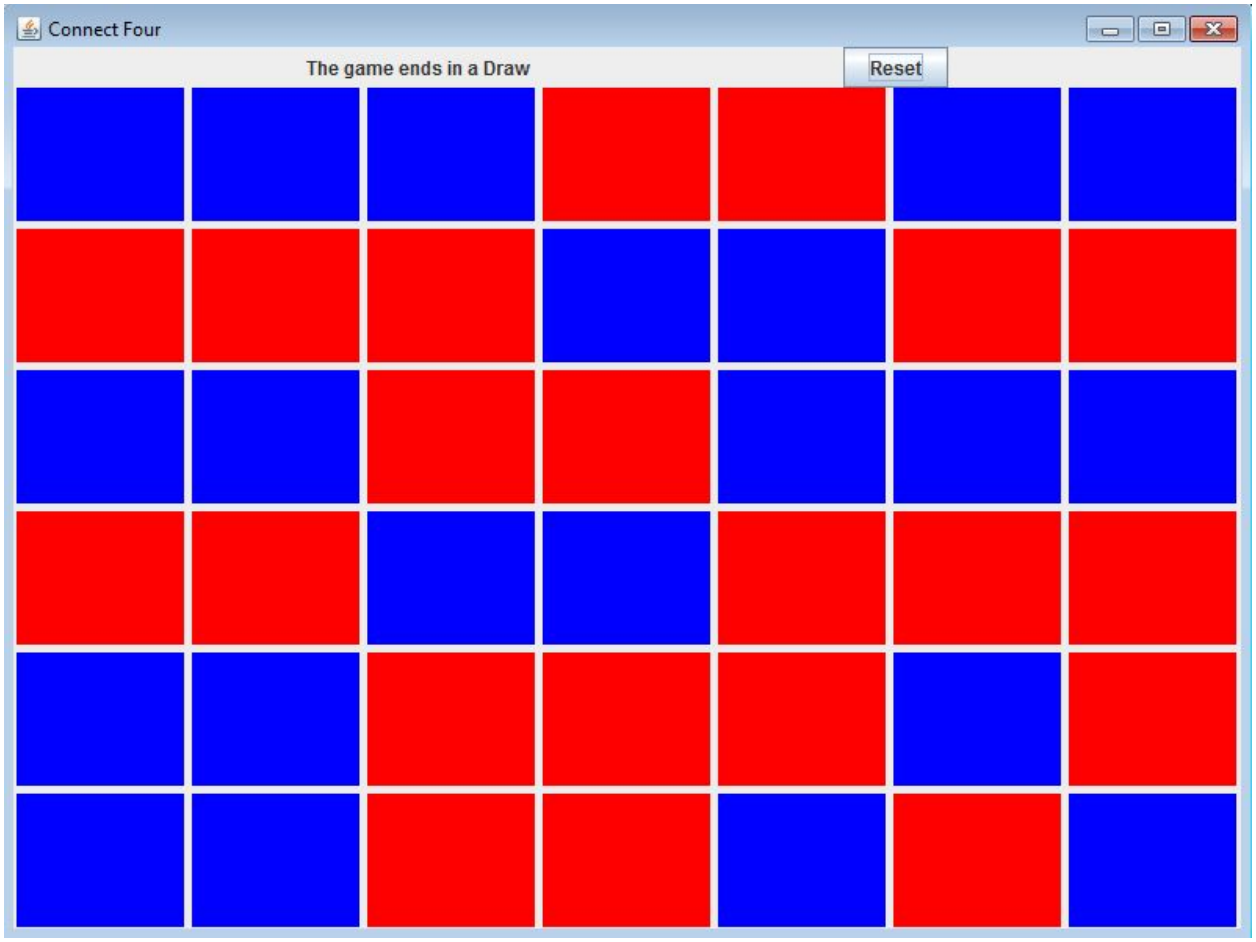

Connect Four
Blue player is the Winner!
Reset

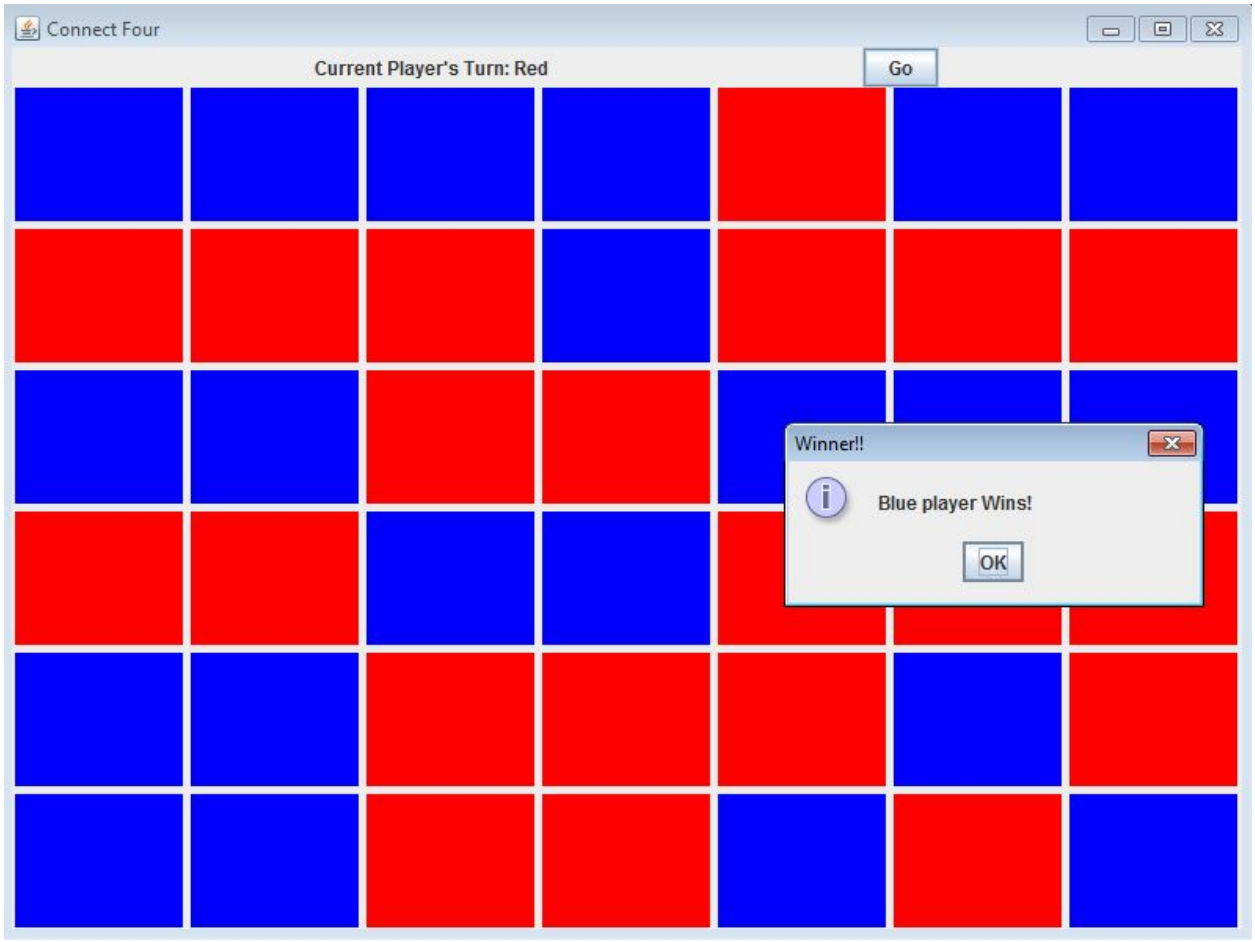

Connect Four
Current Player's Turn: Blue
Go
Winner!!
Red player Wins!
OK

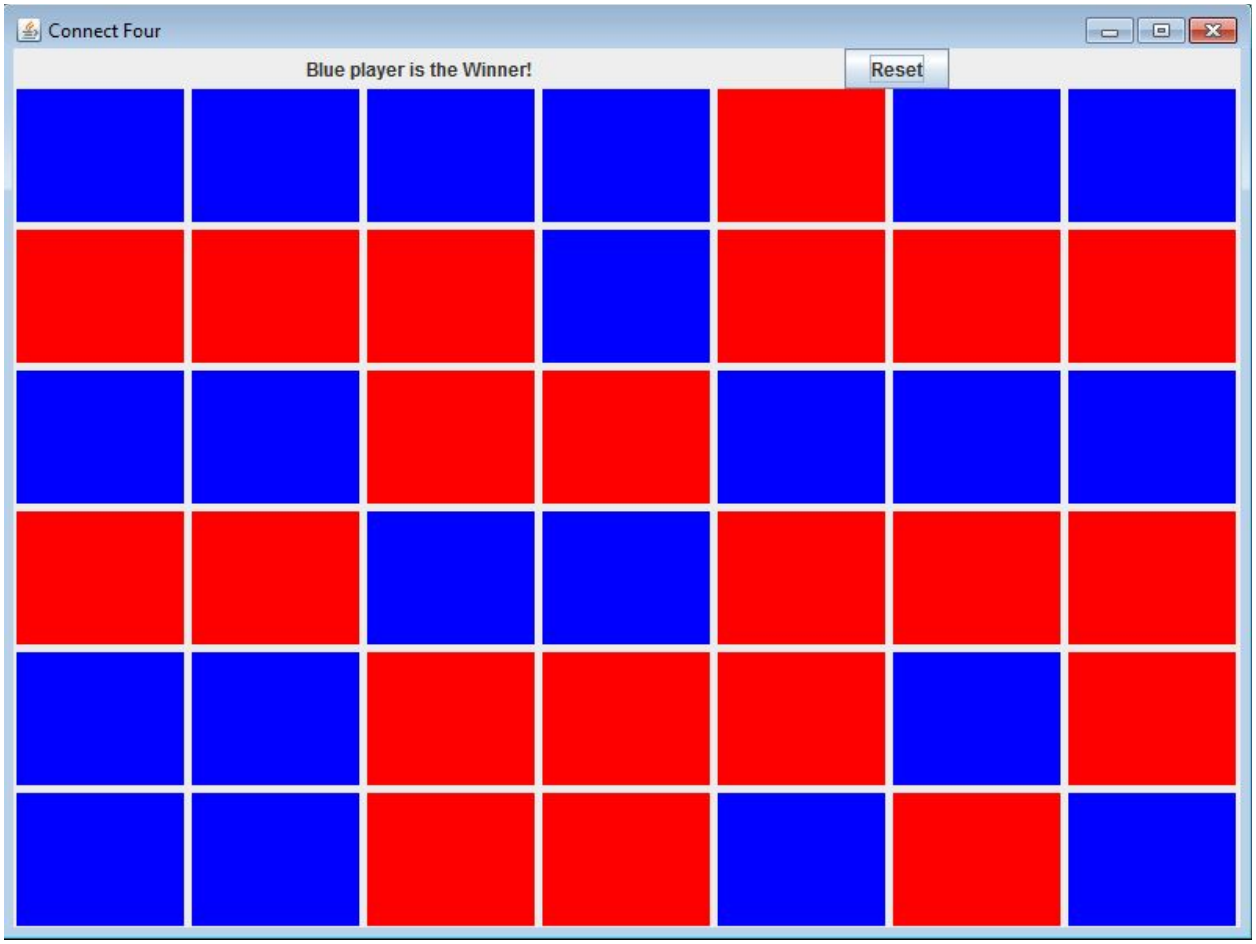

Connect Four
Current Player's Turn: Red
Go
Winner!!
Blue player Wins!
OK


Connect Four
Blue player is the Winner!
Reset


Connect Four
Current Player's Turn: Red
Go
Tie Game
It's a Draw!
OK


Connect Four
The game ends in a Draw
Reset


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


Connect Four
Blue player is the Winner!
Reset