File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\Main.java

```
 1 package connect4;
 2
 3 import javafx.application.Application;
 4 import javafx.fxml.FXMLLoader;
 5 import javafx.geometry.Rectangle2D;
 6 import javafx.scene.Parent;
 7 import javafx.scene.Scene;
 8 import javafx.scene.input.KeyCombination;
 9 import javafx.stage.Screen;
10 import javafx.stage.Stage;
11
12 public class Main extends Application {
13
14     //global variables
15     public Rectangle2D bounds = Screen.getPrimary().getBounds();
16     public static Parent mainMenu;
17     public static Parent gameView;
18     public static Scene mainScene;
19     public static Scene gameScene;
20     public static Stage mainStage;
21     public static double screenWidth;
22     public static double screenHeight;
23     public static boolean computerOn;
24     public static String player1Name, player2Name;
25
26     //where the program is run from
27     @Override
28     public void start(Stage primaryStage) throws Exception{
29         screenWidth = bounds.getWidth();
30         screenHeight = bounds.getHeight();
31
32         mainStage = primaryStage;
33         mainMenu = FXMLLoader.load(getClass().getResource("mainMenu.fxml"));
34         gameView = FXMLLoader.load(getClass().getResource("gameView.fxml"));
35         primaryStage.setTitle("Wykes Connect 4");
36         primaryStage.setFullScreen(true);
37         primaryStage.setFullScreenExitKeyCombination(KeyCombination.NO_MATCH);
38         mainScene = new Scene(mainMenu, screenWidth, screenHeight);
39         gameScene = new Scene(gameView, screenWidth, screenHeight);
40         primaryStage.setScene(mainScene);
41         primaryStage.show();
42     }
43
44
45     public static void main(String[] args) {
46         launch(args);
47     }
48 }
49
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\Board.java

```
1  package connect4;
2
3  import javafx.scene.Parent;
4
5  public class Board {
6
7      private int[][] scores;
8      private int winStartX, winStartY, winEndX, winEndY = -1;
9
10
11     public Board(int[][] scores) {
12         this.scores = scores;
13     }
14
15     //setters and getters
16     public int[][] getScores() {
17         return this.scores;
18     }
19     public int getMoveAvailableForColumn(int column) {
20         int rowIndex = 5;
21         for(int i = 0; i < 6; i++) {
22             if(scores[column][i] > 0) {
23                 rowIndex = i - 1;
24                 break;
25             }
26         }
27         return rowIndex;
28     }
29     public void setScore(int x, int y, int value) {
30         this.scores[x][y] = value;
31     }
32
33     //checks to see if all of the slots are full
34     public boolean checkForDraw() {
35         boolean full = true;
36         for(int i = 0; i < 7; i++) {
37             for(int j = 0; j < 6; j++) {
38                 if(scores[i][j] == 0) full = false;
39             }
40         }
41
42         return full;
43     }
44
45     public boolean checkForWin() {
46         //there is a 4 by 3 amount of possible 4 by 4 squares I have named them
47         //TopFarLeft TopLeft TopRight TopFarRight
48         //MiddleFarLeft MiddleLeft MiddleRight MiddleFarRight
49         //BottomFarLeft BottomLeft BottomRight BottomFarRight;
50         int[][] topFarLeft, topLeft, topRight, topFarRight,
51                 middleFarLeft, middleLeft, middleRight, middleFarRight,
52                 bottomFarLeft, bottomLeft, bottomRight, bottomFarRight = new int[4][4];
53
54         topFarLeft = createFourByFourFromCoordinates(0, 0);
55         topLeft = createFourByFourFromCoordinates(1, 0);
56         topRight = createFourByFourFromCoordinates(2, 0);
57         topFarRight = createFourByFourFromCoordinates(3, 0);
58
59         middleFarLeft = createFourByFourFromCoordinates(0, 1);
60         middleLeft = createFourByFourFromCoordinates(1, 1);
61         middleRight = createFourByFourFromCoordinates(2, 1);
62         middleFarRight = createFourByFourFromCoordinates(3, 1);
63
64         bottomFarLeft = createFourByFourFromCoordinates(0, 2);
65         bottomLeft = createFourByFourFromCoordinates(1, 2);
66         bottomRight = createFourByFourFromCoordinates(2, 2);
67         bottomFarRight = createFourByFourFromCoordinates(3, 2);
```

```
68
69          if(checkForWinFromSmallArray(topFarLeft, 0, 0) ||
70                  checkForWinFromSmallArray(topLeft, 1, 0) ||
71                  checkForWinFromSmallArray(topRight, 2, 0) ||
72                  checkForWinFromSmallArray(topFarRight, 3, 0) ||
73                  checkForWinFromSmallArray(middleFarLeft, 0, 1) ||
74                  checkForWinFromSmallArray(middleLeft, 1, 1) ||
75                  checkForWinFromSmallArray(middleRight, 2, 1) ||
76                  checkForWinFromSmallArray(middleFarRight, 3, 1) ||
77                  checkForWinFromSmallArray(bottomFarLeft, 0, 2) ||
78                  checkForWinFromSmallArray(bottomLeft, 1, 2) ||
79                  checkForWinFromSmallArray(bottomRight, 2, 2) ||
80                  checkForWinFromSmallArray(bottomFarRight, 3, 2)
81                  ) {
82              return true;
83          }
84          else return false;
85      }
86
87      //Essentially, each four by four square can be made from the top left corner
    coordinates
88      private int[][] createFourByFourFromCoordinates(int startX, int startY) {
89          int[][] output = new int[4][4];
90          for(int i = 0; i < 4; i++) {
91              for(int j = 0; j < 4; j++) {
92                  output[i][j] = scores[i + startX][j + startY];
93              }
94          }
95          return output;
96      }
97
98      //when given a 4 by 4 input will calculate whether there is a win in that section of
     the board
99      private boolean checkForWinFromSmallArray(int[][] input, int startX, int startY) {
100         boolean win = false;
101         int leftRightDiag = 0;
102         int rightLeftDiag = 0;
103
104         //cycles through rows and columns and adds score if it finds one
105         for(int i = 0; i < 4; i++) {
106
107             int rowTotal = 0;
108             int columnTotal = 0;
109
110             for(int j = 0; j < 4; j++) {
111                 rowTotal += input[j][i];
112                 columnTotal += input[i][j];
113
114             }
115
116             //checks if any rows are winning and then records the start and end x and y
    coordinates
117             if(rowTotal == 40 || rowTotal == 4) {
118                 win = true;
119                 winStartY = i + startY;
120                 winEndY = i + startY;
121
122                 winStartX = startX;
123                 winEndX = startX + 3;
124
125                 break;
126             }
127
128             //does the same for columns
129             if(columnTotal == 40 || columnTotal == 4) {
130                 win = true;
131
```

```
132                 winStartX = i + startX;
133                 winEndX = i + startX;
134
135                 winStartY = startY;
136                 winEndY = startY + 3;
137
138                 break;
139
140             }
141
142         }
143
144         //diagonals
145         int topBottomDiag = 0;
146         int bottomTopDiag = 0;
147         for(int i = 0; i < 4; i++) {
148
149             topBottomDiag += input[i][i];
150             bottomTopDiag += input[i][3-i];
151
152             if(topBottomDiag == 40 || topBottomDiag == 4) {
153                 win = true;
154                 winStartX = startX;
155                 winStartY = startY;
156                 winEndX = 3 + startX;
157                 winEndY = 3 + startY;
158
159                 break;
160             }
161             if(bottomTopDiag == 40 || bottomTopDiag == 4){
162                 win = true;
163                 winStartX = startX;
164                 winStartY = 3 + startY;
165                 winEndX = 3 + startX;
166                 winEndY = startY;
167
168                 break;
169             }
170
171         }
172
173
174         return win;
175
176     }
177
178     //checks for if the board is full
179     public boolean isFull() {
180         boolean full = true;
181         for(int i = 0; i < 7; i++) {
182             for(int j = 0; j < 6; j++) {
183                 if(scores[i][j] == 0) {
184                     full = false;
185                 }
186             }
187         }
188
189         return full;
190     }
191
192     //getters
193     public int getWinStartX() {
194         return this.winStartX;
195     }
196     public int getWinStartY() {
197         return this.winStartY;
198     }
```

```
199     public int getWinEndX() {
200         return this.winEndX;
201     }
202     public int getWinEndY() {
203         return this.winEndY;
204     }
205
206 }
207
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\gameView.fxml

```
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.canvas.Canvas?>
<?import javafx.scene.layout.VBox?>
<?import java.net.URL?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.MenuItem?>
<GridPane fx:controller="connect4.gameViewController" fx:id="root"
          xmlns:fx="http://javafx.com/fxml" alignment="center">

    <GridPane GridPane.columnIndex="0" GridPane.rowIndex="0" GridPane.columnSpan="3">
        <MenuBar fx:id="gameMenuBar" GridPane.columnIndex="0" GridPane.rowIndex="0"
GridPane.columnSpan="2">
            <menus>
                <Menu text="Game" fx:id="gameOption">
                    <MenuItem text="Main Menu" onAction="#mainMenuButtonPressed"/>
                    <MenuItem text="Close" onAction="#closeWindow"/>
                </Menu>

            </menus>
        </MenuBar>

        <Button text="X" fx:id="exitButton" onAction="#closeWindow"
                GridPane.columnIndex="1" GridPane.rowIndex="0" GridPane.halignment="RIGHT
"/>
    </GridPane>

    <GridPane fx:id="leftHandPane" styleClass="sidePane" vgap="20"
              GridPane.columnIndex="0" GridPane.rowIndex="1" alignment="CENTER_LEFT">
        <Label fx:id="player1Label" styleClass="scoreLabel"
               GridPane.columnIndex="0" GridPane.rowIndex="0" alignment="CENTER"/>
        <Label fx:id="player1Score" styleClass="scoreOutput"
               GridPane.columnIndex="1" GridPane.rowIndex="0" alignment="CENTER"/>
        <Label fx:id="player2Label" styleClass="scoreLabel"
               GridPane.columnIndex="0" GridPane.rowIndex="1" alignment="CENTER"/>
        <Label fx:id="player2Score" styleClass="scoreOutput"
               GridPane.columnIndex="1" GridPane.rowIndex="1" alignment="CENTER"/>

        <Button text="New Game" styleClass="gameButton" onAction="#newGameButtonPressed"
                GridPane.columnIndex="0" GridPane.rowIndex="2" GridPane.columnSpan="2"
GridPane.halignment="RIGHT"/>
        <Button text="New Set" styleClass="gameButton" onAction="#newSetButtonPressed"
                GridPane.columnIndex="0" GridPane.rowIndex="3" GridPane.columnSpan="2"
GridPane.halignment="RIGHT"/>

        <VBox prefWidth="20" GridPane.columnIndex="2" GridPane.rowIndex="0" GridPane.
rowSpan="5"/>
    </GridPane>
    <Separator opacity="0" prefWidth="50" GridPane.columnIndex="1" GridPane.rowIndex="1"
/>
    <GridPane GridPane.columnIndex="2" GridPane.rowIndex="1">
        <Canvas fx:id="gameCanvas" GridPane.columnIndex="0" GridPane.rowIndex="0"
GridPane.columnSpan="7"/>
        <Button fx:id="columnOneButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
                GridPane.columnIndex="0" GridPane.rowIndex="0"/>
        <Button fx:id="columnTwoButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
                GridPane.columnIndex="1" GridPane.rowIndex="0"/>
        <Button fx:id="columnThreeButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
                GridPane.columnIndex="2" GridPane.rowIndex="0"/>
        <Button fx:id="columnFourButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
```

```
                GridPane.columnIndex="3" GridPane.rowIndex="0"/>
        <Button fx:id="columnFiveButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
                GridPane.columnIndex="4" GridPane.rowIndex="0"/>
        <Button fx:id="columnSixButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
                GridPane.columnIndex="5" GridPane.rowIndex="0"/>
        <Button fx:id="columnSevenButton" opacity="0" onAction="#columnButtonPressed"
onMouseEntered="#columnHovered" onMouseExited="#columnUnhovered"
                GridPane.columnIndex="6" GridPane.rowIndex="0"/>
    </GridPane>

    <stylesheets>
        <URL value="@gameViewStyle.css"/>
    </stylesheets>
</GridPane>
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\mainMenu.fxml

```
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import java.net.URL?>
<?import javafx.scene.control.Separator?>
<GridPane fx:controller="connect4.mainMenuController"
          xmlns:fx="http://javafx.com/fxml" alignment="TOP_CENTER" hgap="10">

    <Separator opacity="0" fx:id="verticalBuffer" GridPane.columnIndex="0" GridPane.
rowIndex="0"/>

    <Label text="Connect 4" fx:id="mainTitle" GridPane.columnIndex="0" GridPane.rowIndex=
"1" GridPane.halignment="CENTER"/>

    <Button text="New Game" fx:id="newGameButton" styleClass="mainMenuButton" onAction="#
newGameButtonPressed"
            GridPane.columnIndex="0" GridPane.rowIndex="2" GridPane.halignment="CENTER"/>

    <GridPane fx:id="hiddenBox" GridPane.columnIndex="0" GridPane.rowIndex="3" GridPane.
halignment="CENTER"/>

    <Separator opacity="0" prefHeight="10" GridPane.columnIndex="0" GridPane.rowIndex="4"
/>

    <Button text="Exit Game" styleClass="mainMenuButton" onAction="#exitGameButtonPressed
"
            GridPane.columnIndex="0" GridPane.rowIndex="5" GridPane.halignment="CENTER"/>

    <stylesheets>
        <URL value="@mainMenuStyle.css"/>
    </stylesheets>

</GridPane>
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\gameViewStyle.css

```
.root {
    default-button-color: #e6e6e6;
    hover-button-color: #d9d9d9;
}
#gameOption {
    -fx-pref-height: 50;
}
#exitButton {
    -fx-background-color: #ff4d4d;
    -fx-text-fill: white;
    -fx-background-radius: 0;
    -fx-pref-width: 70;
    -fx-pref-height: 50;
}
#exitButton:hover {
    -fx-background-color: #ff1a1a;
}

.sidePane {
    -fx-background-color: hover-button-color;
}

.scoreLabel {
    -fx-pref-width: 250;
    -fx-pref-height: 50;
}
.scoreOutput {
    -fx-pref-width: 250;
    -fx-pref-height: 50;
    -fx-background-color: default-button-color;
}
.gameButton {
    -fx-pref-height: 50;
    -fx-pref-width: 450;
    -fx-background-radius: 0;

}


```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\mainMenuStyle.css

```
.root {
    default-button-color: #e6e6e6;
    hover-button-color: #d9d9d9;
}
#mainTitle {
    -fx-font-size: 100;
}

#hiddenBox {
    -fx-background-color: default-button-color;
    -fx-max-width: 400;
}

.mainMenuButton {
    -fx-background-radius: 0;
    -fx-background-color: default-button-color;
    -fx-pref-width: 400;
    -fx-pref-height: 70;
}

.mainMenuButton:hover {
    -fx-background-color: hover-button-color;
}
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\gameViewController.java

```
 1 package connect4;
 2
 3 import javafx.animation.Animation;
 4 import javafx.animation.AnimationTimer;
 5 import javafx.animation.PathTransition;
 6 import javafx.event.ActionEvent;
 7 import javafx.fxml.FXML;
 8 import javafx.geometry.Rectangle2D;
 9 import javafx.scene.Scene;
10 import javafx.scene.canvas.Canvas;
11 import javafx.scene.canvas.GraphicsContext;
12 import javafx.scene.control.*;
13 import javafx.scene.input.MouseEvent;
14 import javafx.scene.layout.GridPane;
15 import javafx.scene.layout.VBox;
16 import javafx.scene.paint.Color;
17 import javafx.scene.paint.Paint;
18 import javafx.scene.shape.*;
19 import javafx.scene.text.Font;
20 import javafx.util.Duration;
21
22 import javax.net.ssl.SSLContext;
23 import java.util.Optional;
24
25 import static connect4.Main.*;
26 import static javafx.animation.Animation.INDEFINITE;
27
28 public class gameViewController {
29
30     @FXML private GridPane root;
31
32     @FXML private GridPane leftHandPane;
33     @FXML private Label player1Label;
34     @FXML private Label player2Label;
35     @FXML private Label player1Score;
36     @FXML private Label player2Score;
37
38     @FXML private Canvas gameCanvas;
39     @FXML private MenuBar gameMenuBar;
40     @FXML private Button columnOneButton;
41     @FXML private Button columnTwoButton;
42     @FXML private Button columnThreeButton;
43     @FXML private Button columnFourButton;
44     @FXML private Button columnFiveButton;
45     @FXML private Button columnSixButton;
46     @FXML private Button columnSevenButton;
47     private Button[] columnButtons;
48
49     private Color BOLDYELLOW = Color.rgb(255, 255, 0);
50     private Color FADEDYELLOW = Color.rgb(255, 255, 153);
51     private Color BOLDRED = Color.rgb(255, 51, 0);
52     private Color FADEDRED = Color.rgb(255, 153, 153);
53
54     private GraphicsContext gc;
55     private int MENUBARHEIGHT = 50;
56     private double ROWHEIGHT = (screenHeight - MENUBARHEIGHT) / 6;
57     private Board currentBoard;
58     private boolean turn = false;
59     private boolean gamePlaying = true;
60     private int[][] newGame;
61     private int player1ScoreValue = 0;
62     private int player2ScoreValue = 0;
63
64     //intialize runs when the scene is loaded up
65     @FXML private void initialize() {
66
67         newGame = new int[7][6];
```

```
68          for(int i = 0; i < 7; i++) {
69              for (int j = 0; j < 6; j++) {
70                  newGame[i][j] = 0;
71              }
72          }
73          currentBoard = new Board(newGame);
74
75          //sets up the actual game canvas
76          columnButtons = new Button[]{columnOneButton, columnTwoButton, columnThreeButton
    , columnFourButton,
77                  columnFiveButton, columnSixButton, columnSevenButton};
78
79          //set the sizes of all the components in the window dynamically to fit any
    screen
80          gameCanvas.setWidth(ROWHEIGHT*7);
81          gameCanvas.setHeight(ROWHEIGHT*6);
82
83          gameMenuBar.setPrefHeight(MENUBARHEIGHT);
84          gameMenuBar.setPrefWidth(screenWidth);;
85
86          for(int i = 0; i < columnButtons.length; i++) {
87              columnButtons[i].setPrefWidth(ROWHEIGHT);
88              columnButtons[i].setPrefHeight(ROWHEIGHT * 6);
89          }
90
91
92          //set up graphics
93          try{
94              gc = gameCanvas.getGraphicsContext2D();
95          }
96          catch (NullPointerException e) {
97              System.out.println("Problem Setting Up Canvas");
98          }
99
100         drawBoard();
101
102
103         //Animation timer attempts to "tick" 60 times per second
104         new AnimationTimer() {
105             public void handle(long currentNanoTime) {
106                 tick();
107             }
108         }.start();
109
110     }
111
112     @FXML private void closeWindow() {
113         mainStage.close();
114     }
115     @FXML private void mainMenuButtonPressed() {
116         mainStage.setScene(mainScene);
117         mainStage.setFullScreen(true);
118     }
119
120     @FXML private void columnButtonPressed(ActionEvent e) {
121        if(gamePlaying) {
122            Button b = (Button) e.getSource();
123
124            int x = 0;
125            for(int i = 0; i < columnButtons.length; i++) {
126                if(b.equals(columnButtons[i])){
127                    x = i;
128                }
129            }
130
131            int y = currentBoard.getMoveAvailableForColumn(x);
132
```

```
133             if(y > -1) {
134                 if(turn) {
135                     currentBoard.getScores()[x][y] = 10;
136                     updateBoard();
137                     turn = false;
138                 }
139                 else if (!turn) {
140                     currentBoard.getScores()[x][y] = 1;
141                     updateBoard();
142                     turn = true;
143                 }
144             }
145 
146             checkForEndState();
147 
148             if(!currentBoard.checkForWin()) {
149                 drawFadedRectangle(b);
150             }
151         }
152 
153 
154     }
155     @FXML private void columnHovered(MouseEvent e) {
156         if(gamePlaying) {
157             Button b = (Button) e.getSource();
158 
159             drawFadedRectangle(b);
160         }
161     }
162     @FXML private void columnUnhovered(MouseEvent e) {
163         if(gamePlaying) {
164             Button b = (Button) e.getSource();
165 
166             int x = 0;
167             for(int i = 0; i < columnButtons.length; i++) {
168                 if(b.equals(columnButtons[i])){
169                     x = i;
170                 }
171             }
172 
173             int y = currentBoard.getMoveAvailableForColumn(x);
174 
175             double posX = x * ROWHEIGHT;
176             double posY = y * ROWHEIGHT;
177             double radius = (ROWHEIGHT - 30) / 2;
178 
179             Color color  = Color.rgb(217, 217, 217);
180             fillHoveredSlot(x, y, radius, color);
181         }
182 
183     }
184 
185     @FXML private void newGameButtonPressed() {
186         resetBoard();
187     }
188     @FXML private void newSetButtonPressed() {
189         resetBoard();
190         player1ScoreValue = 0;
191         player2ScoreValue = 0;
192     }
193 
194     //the tick command
195     private void tick() {
196         checkScores();
197 }
198 
199     //method to draw board
```

```
200     private void drawBoard() {
201         double circleDiameter = ROWHEIGHT - 30;
202 
203         double startX = ROWHEIGHT / 2;
204         double startY = ROWHEIGHT / 2;
205 
206         for(int i = 0; i < 7; i++)
207             for (int j = 0; j < 6; j++) {
208                 double x = startX + i * ROWHEIGHT;
209                 double y = startY + j * ROWHEIGHT;
210 
211                 Circle slot = new Circle(x, y, circleDiameter / 2);
212                 slot.setFill(Color.rgb(217, 217, 217));
213 
214                 drawCircle(slot);
215             }
216     }
217 
218     //method to draw circle from given Circle class
219     private void drawCircle(Circle circle) {
220         double x = circle.getCenterX() - circle.getRadius();
221         double y = circle.getCenterY() - circle.getRadius();
222         double diameter = circle.getRadius() * 2;
223         gc.setFill(circle.getFill());
224         gc.fillOval(x, y, diameter, diameter);
225     }
226 
227     //method to set labels to be equal to the main menu name input
228     private void checkScores() {
229         if(player1Name!=null) {
230             if(player1Name.isEmpty()) {
231                 player1Label.setText("Player 1's Score: ");
232             }
233             else {
234                 player1Label.setText(player1Name + "'s Score: ");
235             }
236         }
237         if(player2Name!=null) {
238             if(player2Name.isEmpty()) {
239                 player2Label.setText("Player 2's Score: ");
240             }
241             else {
242                 player2Label.setText(player2Name + "'s Score: ");
243             }
244         }
245         player1Score.setText(String.valueOf(player1ScoreValue));
246         player2Score.setText(String.valueOf(player2ScoreValue));
247     }
248 
249     //method to fill a slot on the board when given the position, radius and colour
250     private void fillSlot(int x, int y, double radius, Color color) {
251         double posX = (ROWHEIGHT * x) + (ROWHEIGHT / 2);
252         double posY = (ROWHEIGHT * y) + (ROWHEIGHT / 2);
253 
254         Circle circle = new Circle(posX,posY,radius, color);
255 
256         drawCircle(circle);
257     }
258 
259     //method to fill a slot on the board when given the position, radius and colour
260     private void fillHoveredSlot(int x, int y, double radius, Color color) {
261         double posX = (ROWHEIGHT * x) + (ROWHEIGHT / 2);
262         double posY = (ROWHEIGHT * y) + (ROWHEIGHT / 2);
263         Circle circle = new Circle(posX, posY, radius, color);
264         drawCircle(circle);
265 
266     }
```

```
267
268     //method to update the graphics on the board to be equal to the score
269     private void updateBoard() {
270         gc.clearRect(0, 0, screenWidth, screenHeight);
271         drawBoard();
272         int[][] scores = currentBoard.getScores();
273         for(int i = 0; i < 7; i++) {
274             for(int j = 0; j < 6; j++) {
275                 int tempScore = scores[i][j];
276                 double radius = (ROWHEIGHT - 40) / 2;
277                 if(tempScore == 1) {
278                     fillSlot(i, j, radius, BOLDRED);
279                 }
280                 else if(tempScore == 10) {
281                     fillSlot(i, j, radius, BOLDYELLOW);
282                 }
283             }
284         }
285
286         for(int i = 0; i < 6; i++)  {
287             for(int j = 0; j < 7; j++) {
288                 System.out.print(scores[j][i] + " ");
289             }
290             System.out.print("\n");
291         }
292         System.out.println("=================");
293     }
294
295     //method to check if the game has ended
296     private void checkForEndState() {
297         if(currentBoard.checkForWin()) {
298             double startX, endX, startY, endY;
299             startX = (ROWHEIGHT * currentBoard.getWinStartX()) + (ROWHEIGHT / 2);
300             startY = (ROWHEIGHT * currentBoard.getWinStartY()) + (ROWHEIGHT / 2);
301             endX = (ROWHEIGHT * currentBoard.getWinEndX()) + (ROWHEIGHT / 2);
302             endY = (ROWHEIGHT * currentBoard.getWinEndY()) + (ROWHEIGHT / 2);
303
304             gc.setLineWidth(10);
305             gc.setStroke(Color.LIGHTGRAY);
306             gc.setLineCap(StrokeLineCap.ROUND);
307             gc.strokeLine(startX, startY, endX, endY);
308             gamePlaying = false;
309
310             if(turn) player1ScoreValue += 1;
311             else player2ScoreValue += 1;
312         }
313         else if(currentBoard.checkForDraw()) {
314             gc.setLineWidth(1);
315             Paint tempColor = gc.getFill();
316             gc.setFill(Color.BLACK);
317             gc.setFont(Font.font(100));
318             gc.fillText("Draw!",ROWHEIGHT * 3, (screenHeight - MENUBARHEIGHT) / 2, 100);
319             gc.setFill(tempColor);
320             gc.setLineWidth(10);
321             gc.setStroke(Color.LIGHTGRAY);
322         }
323
324     }
325
326     //method to create alert window when game over
327     private void createAlertWindow(String outText) {
328         Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
329         alert.setTitle("Game Over:");
330         alert.setHeaderText(null);
331         alert.setContentText(outText);
332         Optional<ButtonType> result = alert.showAndWait();
333         if (result.get() == ButtonType.OK){
```

```
334             resetBoard();
335         } else {
336             resetBoard();
337         }
338     }
339 
340     //method to draw a faded out version of where your mouse is hovering
341     private void drawFadedRectangle(Button b) {
342         int x = 0;
343         for(int i = 0; i < columnButtons.length; i++) {
344             if(b.equals(columnButtons[i])){
345                 x = i;
346             }
347         }
348 
349         int y = currentBoard.getMoveAvailableForColumn(x);
350 
351         double radius = (ROWHEIGHT - 40) / 2;
352 
353         if(y > -1) {
354             if(!turn) {
355                 fillHoveredSlot(x, y, radius, FADEDRED);
356             }
357             else if(turn) {
358                 fillHoveredSlot(x, y, radius, FADEDYELLOW);
359             }
360         }
361     }
362 
363     //method to reset board
364     private void resetBoard() {
365         for(int i = 0; i < 7; i++) {
366             for(int j = 0; j < 6; j++) {
367                 newGame[i][j] = 0;
368             }
369         }
370         currentBoard = new Board(newGame);
371         updateBoard();
372         gamePlaying = true;
373 }
374 
375 }
376 
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Connect 4\src\connect4\mainMenuController.java

```
 1 package connect4;
 2
 3 import javafx.animation.KeyFrame;
 4 import javafx.animation.KeyValue;
 5 import javafx.animation.Timeline;
 6 import javafx.event.ActionEvent;
 7 import javafx.event.EventHandler;
 8 import javafx.fxml.FXML;
 9 import javafx.geometry.Pos;
10 import javafx.geometry.Rectangle2D;
11 import javafx.scene.Scene;
12 import javafx.scene.control.*;
13 import javafx.scene.layout.GridPane;
14 import javafx.scene.layout.StackPane;
15 import javafx.scene.shape.Rectangle;
16 import javafx.stage.Screen;
17 import javafx.util.Duration;
18
19 import static connect4.Main.*;
20
21 public class mainMenuController {
22
23     private Rectangle2D bounds = Screen.getPrimary().getBounds();
24
25     @FXML private Button newGameButton;
26     @FXML private GridPane hiddenBox;
27     @FXML private Separator verticalBuffer;
28
29     private boolean hiddenBoxShown = false;
30
31     @FXML private void initialize() {
32         verticalBuffer.setPrefHeight(screenHeight / 2 - 180);
33     }
34
35     @FXML private void newGameButtonPressed() {
36         if(!hiddenBoxShown) {
37             hiddenBoxShown = true;
38             newGameButton.setStyle("default-button-color: #d9d9d9;");
39
40             Label player1Label = new Label("Player 1 Name: ");
41             player1Label.setPrefWidth(200);
42             player1Label.setPrefHeight(50);
43             player1Label.setAlignment(Pos.CENTER);
44             GridPane.setColumnIndex(player1Label, 0);
45             GridPane.setRowIndex(player1Label, 0);
46
47             Label player2Label = new Label("Player 2 Name: ");
48             player2Label.setPrefWidth(200);
49             player2Label.setPrefHeight(50);
50             player2Label.setAlignment(Pos.CENTER);
51             GridPane.setColumnIndex(player2Label, 0);
52             GridPane.setRowIndex(player2Label, 1);
53
54             TextField player1Input = new TextField();
55             player1Input.setPrefWidth(200);
56             player1Input.setPrefHeight(30);
57             player1Input.setAlignment(Pos.CENTER);
58             GridPane.setColumnIndex(player1Input, 1);
59             GridPane.setRowIndex(player1Input, 0);
60
61             TextField player2Input = new TextField();
62             player2Input.setPrefWidth(200);
63             player2Input.setPrefHeight(30);
64             player2Input.setAlignment(Pos.CENTER);
65             GridPane.setColumnIndex(player2Input, 1);
66             GridPane.setRowIndex(player2Input, 1);
67
```

```
68              //adds the go button and event listener to detect when "Enter" is clicked to
    switch scene
69              Button goButton = new Button("Enter");
70              goButton.setAlignment(Pos.CENTER);
71              goButton.setStyle("-fx-background-radius: 0;\n" +
72                      "    -fx-background-color: hover-button-color;\n" +
73                      "    -fx-pref-width: 400;\n" +
74                      "    -fx-pref-height: 70;");
75              goButton.setOnAction(new EventHandler<ActionEvent>() {
76                  @Override
77                  public void handle(ActionEvent event) {
78                      mainStage.setScene(gameScene);
79                      mainStage.setFullScreen(true);
80
81                      player1Name = player1Input.getText();
82                      System.out.println(player1Name + player1Input.getText());
83                      player2Name = player2Input.getText();
84                  }
85              });
86              GridPane.setColumnIndex(goButton, 0);
87              GridPane.setRowIndex(goButton, 3);
88              GridPane.setColumnSpan(goButton, 2);
89
90              hiddenBox.getChildren().addAll(player1Label, player2Label, player1Input,
    player2Input, goButton);
91
92          }
93          else if(hiddenBoxShown) {
94              hiddenBoxShown = false;
95              newGameButton.setStyle("default-button-color: #e6e6e6;");
96
97              hiddenBox.getChildren().clear();
98          }
99
100     }
101
102     @FXML private void exitGameButtonPressed() {
103         mainStage.close();
104     }
105
106
107 }
108
```