File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Noughts And Crosses\src\sample\Main.java

```
 1 package sample;
 2
 3 import javafx.application.Application;
 4 import javafx.fxml.FXMLLoader;
 5 import javafx.scene.Parent;
 6 import javafx.scene.Scene;
 7 import javafx.stage.Stage;
 8
 9 public class Main extends Application {
10
11     @Override
12     public void start(Stage primaryStage) throws Exception{
13         Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
14         primaryStage.setTitle("Wykes Noughts And Crosses");
15         primaryStage.setScene(new Scene(root, 600, 710));
16         primaryStage.setResizable(false);
17         primaryStage.show();
18     }
19
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }
25
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Noughts And Crosses\src\sample\Style.css

```
.root {
    -fx-background-color: #f1f3f3;
}

.gameButton {
    -fx-pref-width: 190;
    -fx-pref-height: 190;
    -fx-background-radius: 0;
    -fx-background-color: #f1f3f3;
    -fx-font-size: 75;
}

.gameButton:hover {
    -fx-background-color: #e3e8e8;
}
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Noughts And Crosses\src\sample\Board.java

```
 1 package sample;
 2
 3 import javax.sound.midi.SysexMessage;
 4 import javax.swing.plaf.synth.SynthTextAreaUI;
 5 import java.util.ArrayList;
 6
 7 public class Board {
 8
 9     private int WIN = 10;
10     private int LOSE = -10;
11     private int DRAW = 0;
12     private int BOARDSIZE = 3;
13     private int addValue;
14
15     private int[][] scores;
16     private int xChange, yChange;
17
18     public Board(int[][] scores, int xChanged, int yChanged, int addValue) {
19         this.scores = scores;
20         this.addValue = addValue;
21
22         if(xChanged != -1 || yChanged != -1) {
23             this.xChange = xChanged;
24             this.yChange = yChanged;
25
26         }
27
28     }
29
30     public ArrayList<Board> getPossibleBoards() {
31         ArrayList<Board> possibleBoards = new ArrayList<>();
32
33         for(int i = 0; i < BOARDSIZE; i++) {
34             for(int j = 0; j < BOARDSIZE; j++) {
35                 if(scores[i][j]==0) {
36                     int[][] tempScores = new int[3][3];
37                     for(int k = 0; k < 3; k++) {
38                         for(int l = 0; l < 3; l++) {
39                             tempScores[k][l] = scores[k][l];
40                         }
41                     }
42                     tempScores[i][j] = addValue;
43
44                     int tempAddValue = 0;
45                     if(addValue == 10) tempAddValue = 1;
46                     else if(addValue == 1) tempAddValue = 10;
47
48                     Board b = new Board(tempScores, i, j, tempAddValue);
49                     possibleBoards.add(b);
50
51                 }
52             }
53         }
54
55         return possibleBoards;
56     }
57
58     //method to check if this is a winning board, draw, or still in play
59     public int checkWin() {
60         int diagLeftRight = scores[0][0] + scores[1][1] + scores[2][2];
61         int diagRightLeft = scores[0][2] + scores[1][1] + scores[2][0];
62         int topRow = scores[0][0] + scores[1][0] + scores[2][0];
63         int centerRow = scores[0][1] + scores[1][1] + scores[2][1];
64         int bottomRow = scores[0][2] + scores[1][2] + scores[2][2];
65         int leftColumn = scores[0][0] + scores[0][1] + scores[0][2];
66         int centerColumn = scores[1][0] + scores[1][1] + scores[1][2];
67         int rightColumn = scores[2][0] + scores[2][1] + scores[2][2];
```

```
68
69          //Checks if any of the rows columns or diagonals is equal to 3 or 30
70          if(diagLeftRight == 3 || diagRightLeft == 3 || topRow == 3 || centerRow == 3||
    bottomRow == 3|| leftColumn == 3
71                  || centerColumn == 3 || rightColumn == 3) {
72              return LOSE;
73          }
74
75          else if(diagLeftRight == 30 || diagRightLeft == 30 || topRow == 30 || centerRow
    == 30 || bottomRow == 30 || leftColumn == 30
76                  || centerColumn == 30 || rightColumn == 30) {
77              return WIN;
78          }
79          else {
80              boolean isFull = true;
81              for(int i = 0; i < BOARDSIZE; i++) {
82                  for(int j = 0; j < BOARDSIZE; j++) {
83                      if(scores[i][j]==0) isFull = false;
84                  }
85              }
86              if(isFull) return DRAW;
87              else return -1;
88
89          }
90      }
91
92      public boolean isValidToMove() {
93          if(this.checkWin() == -1) return true;
94          else return false;
95      }
96
97      public int getXChange() {
98          return this.xChange;
99      }
100     public int getYChange() {
101         return this.yChange;
102     }
103
104
105 }
106
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Noughts And Crosses\src\sample\sample.fxml

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuItem?>
<?import java.net.URL?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.canvas.Canvas?>
<GridPane fx:controller="sample.Controller"
          xmlns:fx="http://javafx.com/fxml" alignment="TOP_CENTER" vgap="10">

    <MenuBar GridPane.columnIndex="0" GridPane.rowIndex="0" prefWidth="600">
        <menus>
            <Menu text="Game" fx:id="gamesMenu">
                <MenuItem text="New Game" onAction="#newGamePressed"/>
            </Menu>

            <Menu text="Edit" fx:id="editMenu">

            </Menu>
        </menus>
    </MenuBar>

    <HBox GridPane.columnIndex="0" GridPane.rowIndex="1">
        <Separator prefWidth="30" prefHeight="20" opacity="0"/>
        <Label text="Player 1 Score: " prefWidth="125" prefHeight="20"/>
        <TextField fx:id="player1Field" editable="false" prefWidth="30"/>
        <Separator opacity="0" prefWidth="30"/>
        <Label text="Player 2 Score: " prefWidth="125" prefHeight="20"/>
        <TextField fx:id="player2Field" editable="false" prefWidth="30"/>
        <Separator opacity="0" prefWidth="30"/>
        <Label text="Computer" prefHeight="20"/>
        <Separator opacity="0" prefWidth="10"/>
        <CheckBox fx:id="computerCheckBox"/>
    </HBox>

    <Separator GridPane.columnIndex="0" GridPane.rowIndex="2"/>

    <GridPane GridPane.columnIndex="0" GridPane.rowIndex="3" hgap="10" vgap="10"
alignment="CENTER" fx:id="gridStyle">
        <Canvas fx:id="gridCanvas" GridPane.columnIndex="0" GridPane.rowIndex="0" width="
590" height="590"
                GridPane.columnSpan="3" GridPane.rowSpan="3"/>
        <Button styleClass="gameButton" fx:id="topLeftButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
            GridPane.columnIndex="0" GridPane.rowIndex="0" onAction="#buttonPressed"/>
        <Button styleClass="gameButton" fx:id="topCenterButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
                GridPane.columnIndex="1" GridPane.rowIndex="0" onAction="#buttonPressed"
/>
        <Button styleClass="gameButton" fx:id="topRightButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
                GridPane.columnIndex="2" GridPane.rowIndex="0" onAction="#buttonPressed"
/>

        <Button styleClass="gameButton" fx:id="leftButton" onMouseEntered="#buttonHovered
" onMouseExited="#buttonMouseLeft"
                GridPane.columnIndex="0" GridPane.rowIndex="1" onAction="#buttonPressed"
/>
        <Button styleClass="gameButton" fx:id="centerButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
```

```
58                GridPane.columnIndex="1" GridPane.rowIndex="1" onAction="#buttonPressed"
/>
59          <Button styleClass="gameButton" fx:id="rightButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
60                GridPane.columnIndex="2" GridPane.rowIndex="1" onAction="#buttonPressed"
/>
61
62          <Button styleClass="gameButton" fx:id="bottomLeftButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
63                GridPane.columnIndex="0" GridPane.rowIndex="2" onAction="#buttonPressed"
/>
64          <Button styleClass="gameButton" fx:id="bottomCenterButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
65                GridPane.columnIndex="1" GridPane.rowIndex="2" onAction="#buttonPressed"
/>
66          <Button styleClass="gameButton" fx:id="bottomRightButton" onMouseEntered="#
buttonHovered" onMouseExited="#buttonMouseLeft"
67                GridPane.columnIndex="2" GridPane.rowIndex="2" onAction="#buttonPressed"
/>
68
69      </GridPane>
70
71      <Label fx:id="turnLabel" GridPane.columnIndex="0" GridPane.rowIndex="4"/>
72
73      <stylesheets>
74          <URL value="@Style.css"/>
75      </stylesheets>
76
77  </GridPane>
```

File - D:\Libraries\Documents\Homework\Computing\Idea Projects\Noughts And Crosses\src\sample\Controller.java

```
 1 package sample;
 2
 3 import javafx.application.Application;
 4 import javafx.event.ActionEvent;
 5 import javafx.event.EventHandler;
 6 import javafx.fxml.FXML;
 7 import javafx.scene.Parent;
 8 import javafx.scene.canvas.Canvas;
 9 import javafx.scene.canvas.GraphicsContext;
10 import javafx.scene.control.*;
11 import javafx.scene.control.Button;
12 import javafx.scene.control.Label;
13 import javafx.scene.control.TextField;
14 import javafx.scene.input.MouseEvent;
15 import javafx.scene.layout.GridPane;
16 import javafx.scene.paint.Color;
17 import javafx.stage.Stage;
18 import javafx.stage.WindowEvent;
19
20 import javax.swing.*;
21 import java.lang.reflect.Array;
22 import java.util.ArrayList;
23 import java.util.Optional;
24 import java.util.Random;
25
26 public class Controller {
27
28     private boolean turn = true;
29     private int[][] scores = new int[3][3];
30     private Button[][] buttons = new Button[3][3];
31     private int player1Score = 0;
32     private int player2Score = 0;
33     private boolean computerState;
34     private int DRAW = 20;
35     private int WIN = 10;
36     private int LOSE = -10;
37     private int NOTHING = 0;
38
39     @FXML private TextField player1Field;
40     @FXML private TextField player2Field;
41
42     @FXML private CheckBox computerCheckBox;
43
44     @FXML private Button topLeftButton;
45     @FXML private Button topCenterButton;
46     @FXML private Button topRightButton;
47     @FXML private Button leftButton;
48     @FXML private Button centerButton;
49     @FXML private Button rightButton;
50     @FXML private Button bottomLeftButton;
51     @FXML private Button bottomCenterButton;
52     @FXML private Button bottomRightButton;
53
54     @FXML private Label turnLabel;
55
56     @FXML private Canvas gridCanvas;
57     private GraphicsContext gc;
58
59     private static boolean computerOn;
60     private static boolean boxChanged = false;
61
62     @FXML private void initialize(){
63         //makes a buttons array with all the buttons stored in it for use in loops
64         buttons[0][0] = topLeftButton;
65         buttons[0][1] = topCenterButton;
66         buttons[0][2] = topRightButton;
67
```

```
68          buttons[1][0] = leftButton;
69          buttons[1][1] = centerButton;
70          buttons[1][2] = rightButton;
71
72          buttons[2][0] = bottomLeftButton;
73          buttons[2][1] = bottomCenterButton;
74          buttons[2][2] = bottomRightButton;
75
76          //sets the scoreboard to 0 at the start
77          player1Field.setText(String.valueOf(player1Score));
78          player2Field.setText(String.valueOf(player2Score));
79
80          turnLabel.setText("Player 1's Turn");
81          setUpCanvas();
82      }
83
84      @FXML private void buttonPressed(ActionEvent e) {
85          //get source of mouse click
86          Button b = (Button) e.getSource();
87
88          if(b.getText().isEmpty() || b.getText().equals("") || boxChanged) {
89              b.setTextFill(Color.BLACK);
90              boxChanged = false;
91              //if it is player 1's turn set source button's text to X
92              if(turn == true) {
93                  b.setText("X");
94                  turn = false;
95              }
96
97              //if it is player 2's turn set the source button's text to O
98              else if(turn == false) {
99                  b.setText("O");
100                 turn = true;
101             }
102             //after setting the button text the game switches to the other player's turn
103
104             updateScores();
105
106             //check for win
107             checkWin();
108
109             if(turn) turnLabel.setText("Player 1's Turn");
110             else if(!turn) turnLabel.setText("Player 2's Turn");
111
112             if(computerState) {
113                 computerTurn();
114                 turn = true;
115                 turnLabel.setText("Player 1's Turn");
116             }
117
118             player1Field.setText(String.valueOf(player1Score));
119             player2Field.setText(String.valueOf(player2Score));
120         }
121
122     }
123
124     //when the box is hovered set the value temporarily to
125     @FXML private void buttonHovered(MouseEvent e) {
126         Button b = (Button) e.getSource();
127
128         if(b.getText().isEmpty() || b.getText().equals("")) {
129             b.setTextFill(Color.LIGHTGRAY);
130             boxChanged = true;
131             if(turn) b.setText("X");
132             else if(!turn) b.setText("O");
133         }
134         else { boxChanged = false; }
```

```
135
136     }
137
138     @FXML private void buttonMouseLeft(MouseEvent e) {
139         Button b = (Button) e.getSource();
140         b.setTextFill(Color.BLACK);
141
142         if(boxChanged) b.setText("");
143
144     }
145
146     //resets the board and sets the scores to 0
147     @FXML private void newGamePressed() {
148         resetBoard();
149         player1Score = 0;
150         player2Score = 0;
151         player1Field.setText("0");
152         player2Field.setText("0");
153
154         computerState = computerCheckBox.isSelected();
155
156     }
157
158     //method to make the score array be equal to the visual state of the board
159     private void updateScores() {
160         boolean isFull = true;
161         for(int i = 0; i < 3; i++) {
162             for(int j = 0; j < 3; j++) {
163                 int temp;
164                 if(!buttons[i][j].getText().isEmpty()) {
165                     if (buttons[i][j].getText().equals("X")) {
166                         scores[i][j] = 1;
167                     }
168                     else if (buttons[i][j].getText().equals("O")) {
169                         scores[i][j] = 10;
170                     }
171                 }
172                 else {
173                     scores[i][j] = 0;
174                     isFull = false;
175                 }
176             }
177         }
178         if(isFull) {
179             popUpWindow("Draw");
180         }
181     }
182
183     //rests the score whenever someone wins or draws
184     private void resetBoard() {
185         for(int i = 0; i < 3; i++) {
186             for(int j = 0; j < 3; j++) {
187                 buttons[i][j].setText("");
188                 scores[i][j] = 0;
189
190                 turn = true;
191                 turnLabel.setText("Player 1's turn");
192             }
193         }
194     }
195
196     //win checking method
197     private void checkWin() {
198         int diagLeftRight = scores[0][0] + scores[1][1] + scores[2][2];
199         int diagRightLeft = scores[0][2] + scores[1][1] + scores[2][0];
200         int topRow = scores[0][0] + scores[1][0] + scores[2][0];
201         int centerRow = scores[0][1] + scores[1][1] + scores[2][1];
```

```
202         int bottomRow = scores[0][2] + scores[1][2] + scores[2][2];
203         int leftColumn = scores[0][0] + scores[0][1] + scores[0][2];
204         int centerColumn = scores[1][0] + scores[1][1] + scores[1][2];
205         int rightColumn = scores[2][0] + scores[2][1] + scores[2][2];
206
207         //Checks if any of the rows columns or diagonals is equal to 3 or 30
208         if(diagLeftRight == 3 || diagRightLeft == 3 || topRow == 3 || centerRow == 3||
    bottomRow == 3|| leftColumn == 3
209                 || centerColumn == 3 || rightColumn == 3) {
210             popUpWindow("Player One Won");
211             player1Score++;
212         }
213
214         if(diagLeftRight == 30 || diagRightLeft == 30 || topRow == 30 || centerRow == 30
     || bottomRow == 30 || leftColumn == 30
215                 || centerColumn == 30 || rightColumn == 30) {
216             player2Score++;
217             popUpWindow("Player Two Won");
218         }
219     }
220
221     //method to display a popup whenever someone wins, loses or draws
222     private void popUpWindow(String playerWonString) {
223         Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
224         alert.setTitle("Game Over:");
225         alert.setHeaderText(null);
226         alert.setContentText(playerWonString);
227         Optional<ButtonType> result = alert.showAndWait();
228         if (result.get() == ButtonType.OK){
229             resetBoard();
230         } else {
231             resetBoard();
232         }
233
234     }
235
236     //processes the computer's turn
237     private void computerTurn() {
238
239         Board currentBoard = new Board(scores, -1, -1, 10);
240         ArrayList<Board> possibleBoards = currentBoard.getPossibleBoards();
241         int[] computerScores = new int[possibleBoards.size()];
242
243         for(int i = 0; i < possibleBoards.size(); i++) {
244             computerScores[i]=0;
245         }
246
247         for(int i = 0; i < possibleBoards.size(); i++) {
248             if(possibleBoards.get(i).checkWin() == 10) {
249                 computerScores[i] += 100;
250             }
251
252             else if(possibleBoards.get(i).checkWin()==-1) {
253                 ArrayList<Board> firstIteration = possibleBoards.get(i).
    getPossibleBoards();
254                 for(int j = 0; j < firstIteration.size(); j++) {
255                     if(firstIteration.get(j).checkWin() == -10) {
256                         computerScores[i] -= 100;
257                     }
258
259                 }
260
261             }
262         }
263
264
265         //find out what the maximum possible value for a turn is
```

```
266         int maximumValue = computerScores[0];
267         for(int i = 0; i < computerScores.length; i++) {
268             if(computerScores[i] > maximumValue) maximumValue = computerScores[i];
269         }
270 
271         //add all possible moves with this value to a list
272         ArrayList<Board> maximumMoves = new ArrayList<>();
273         for(int i = 0; i < computerScores.length; i++) {
274             if(computerScores[i] == maximumValue) {
275                 maximumMoves.add(possibleBoards.get(i));
276             }
277         }
278 
279 //         randomly play a move in any of the maximum moves
280         Random r = new Random();
281         int maximumIndex = r.nextInt(maximumMoves.size());
282         int xChange = maximumMoves.get(maximumIndex).getXChange();
283         int yChange = maximumMoves.get(maximumIndex).getYChange();
284 
285         buttons[xChange][yChange].setText("O");
286         updateScores();
287         checkWin();
288 
289     }
290 
291     //tries to set up canvas
292     private void setUpCanvas() {
293         try {
294             gc = gridCanvas.getGraphicsContext2D();
295             drawGrid(gc);
296 
297         }
298         catch (NullPointerException exception) {
299             System.out.println("Could Not Draw On Canvas");
300         }
301     }
302 
303     //draws grid
304     private void drawGrid(GraphicsContext gc) {
305         gc.setStroke(Color.BLACK);
306         gc.setLineWidth(10);
307 
308         gc.strokeLine(195, 0, 195, 590);
309         gc.strokeLine(395, 0, 395, 590);
310         gc.strokeLine(0, 195, 590, 195);
311         gc.strokeLine(0, 395, 590, 395);
312 
313     }
314 }
315 
```