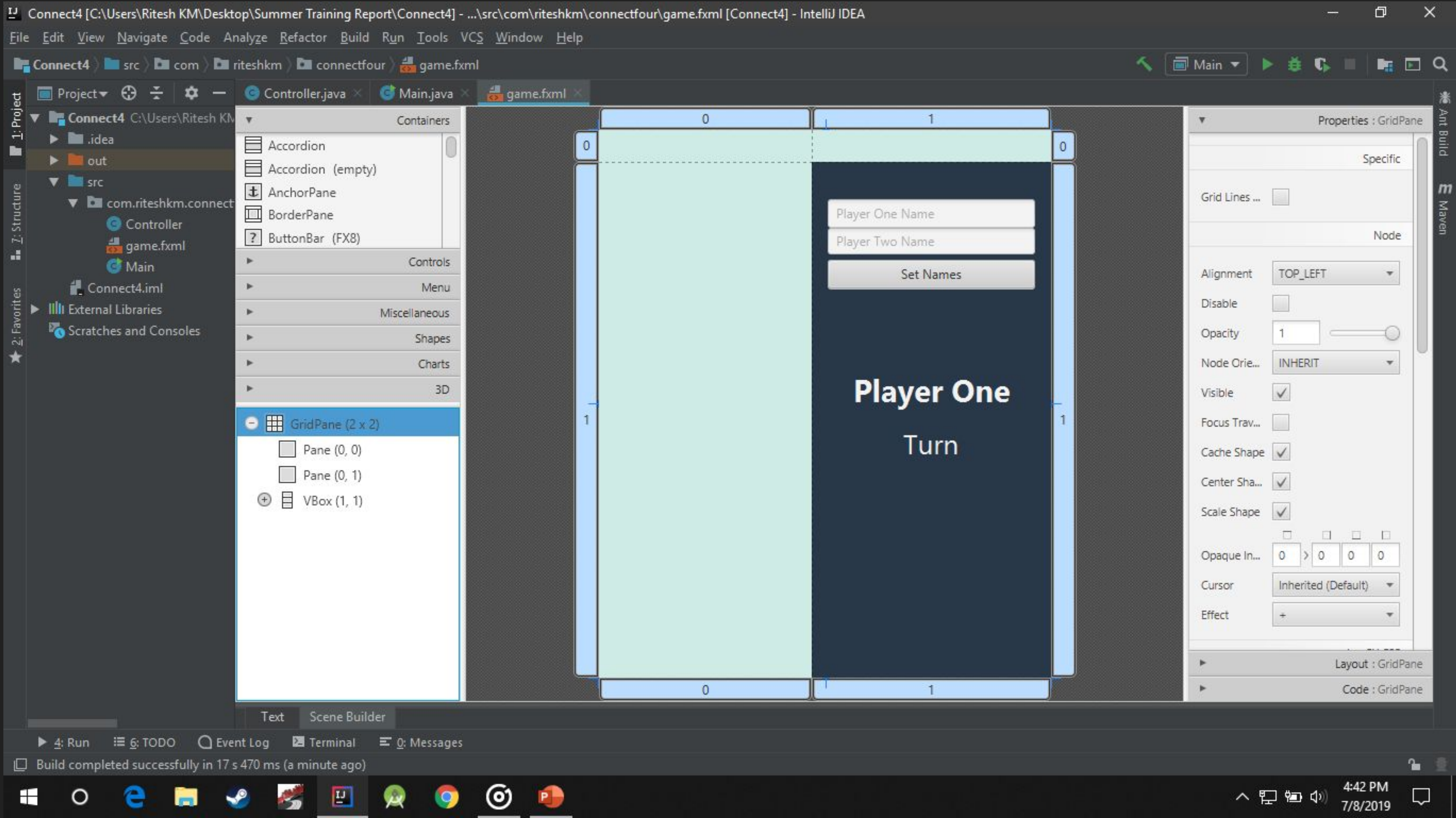
Project Work

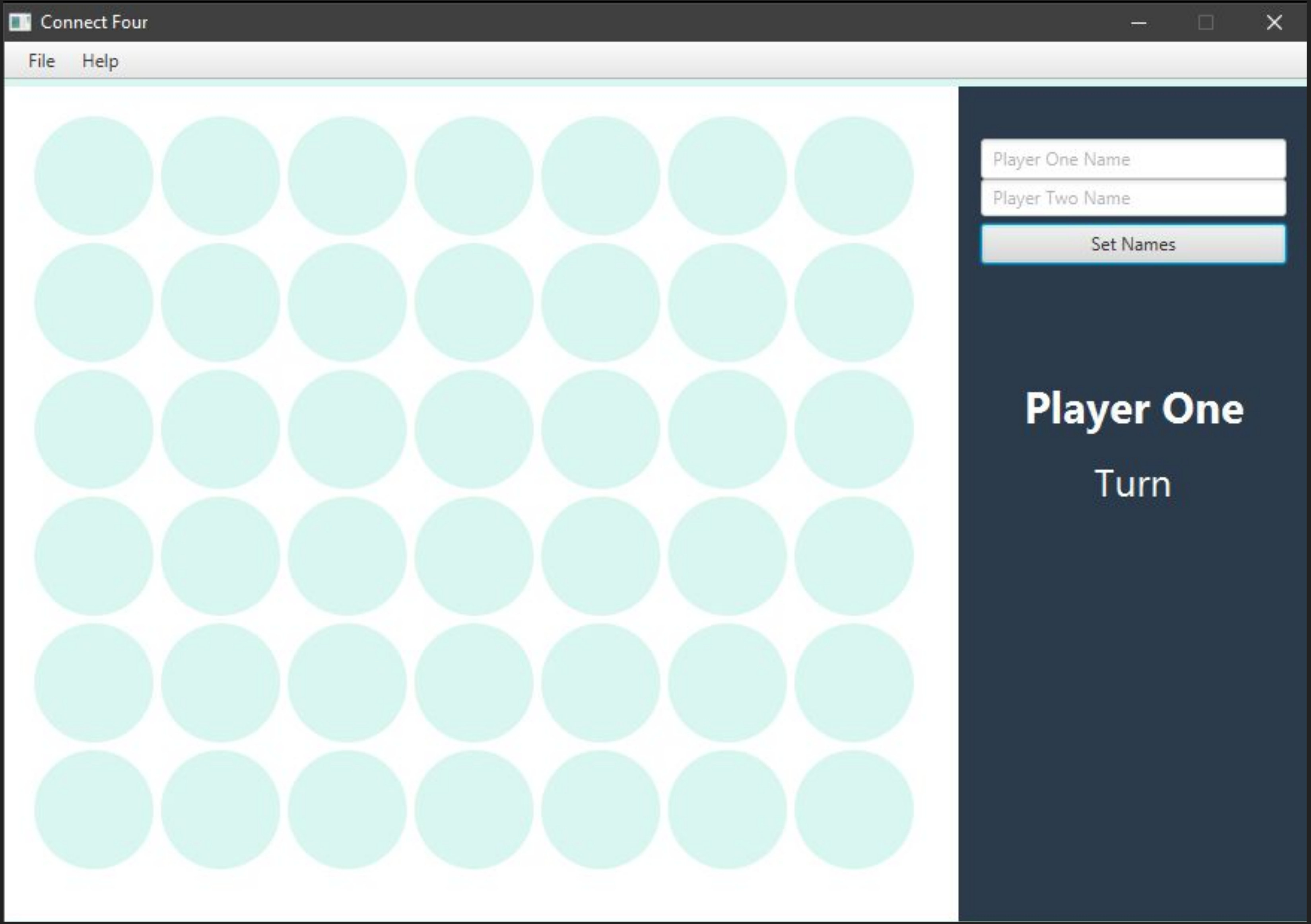
Using – Java, JavaFX, Scene Builder

Layout

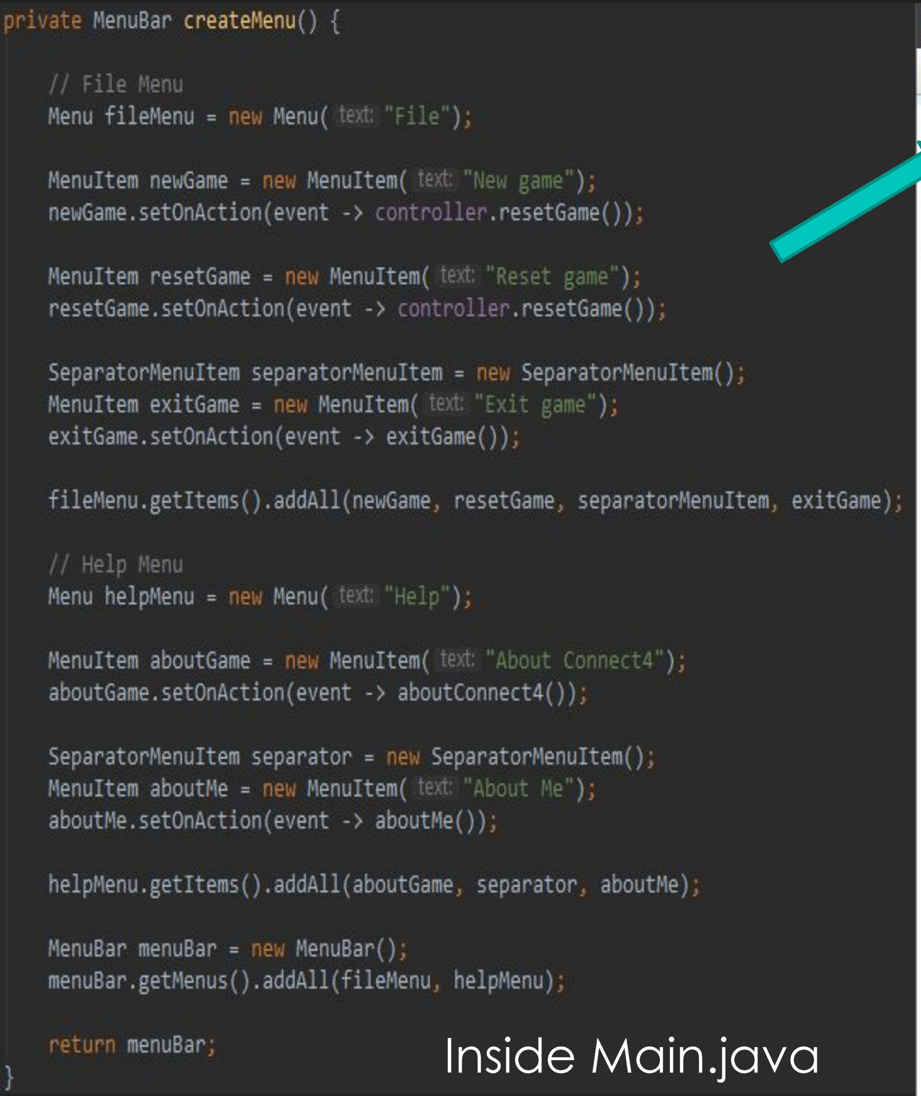
In Scene Builder



Whole Layout of the APP

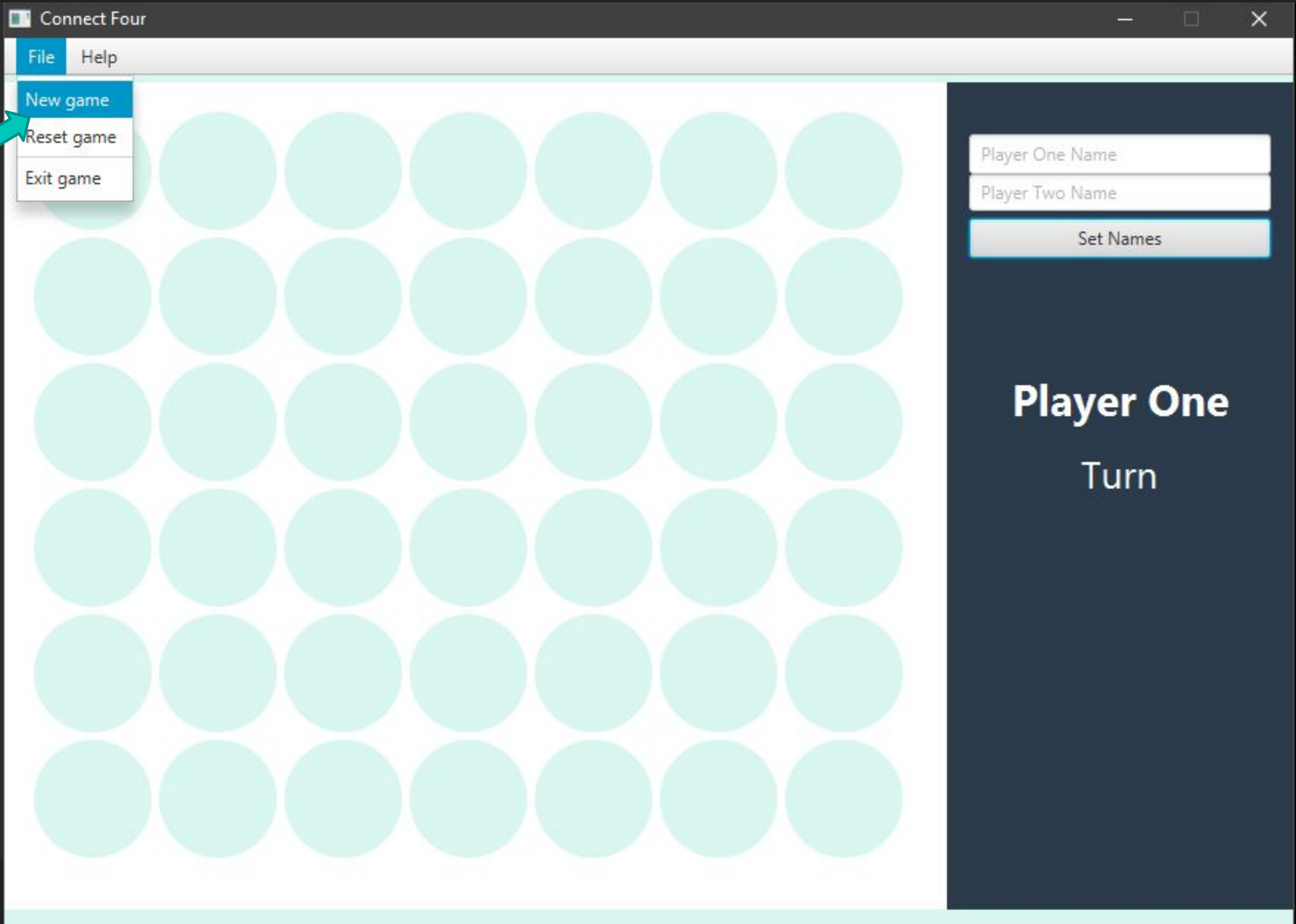


Menu

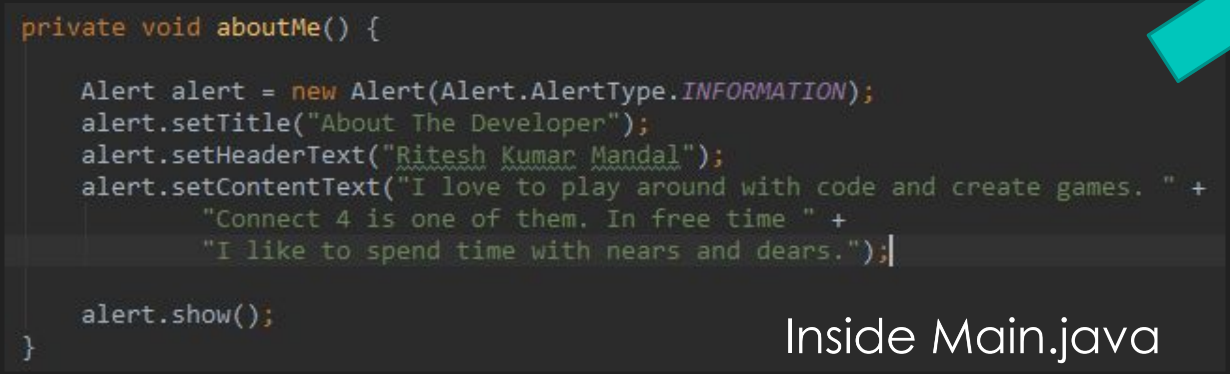


```java
private MenuBar createMenu() {

    // File Menu
    Menu fileMenu = new Menu( text: "File");

    MenuItem newGame = new MenuItem( text: "New game");
    newGame.setOnAction(event -> controller.resetGame());

    MenuItem resetGame = new MenuItem( text: "Reset game");
    resetGame.setOnAction(event -> controller.resetGame());

    SeparatorMenuItem separatorMenuItem = new SeparatorMenuItem();
    MenuItem exitGame = new MenuItem( text: "Exit game");
    exitGame.setOnAction(event -> exitGame());

    fileMenu.getItems().addAll(newGame, resetGame, separatorMenuItem, exitGame);

    // Help Menu
    Menu helpMenu = new Menu( text: "Help");

    MenuItem aboutGame = new MenuItem( text: "About Connect4");
    aboutGame.setOnAction(event -> aboutConnect4());

    SeparatorMenuItem separator = new SeparatorMenuItem();
    MenuItem aboutMe = new MenuItem( text: "About Me");
    aboutMe.setOnAction(event -> aboutMe());

    helpMenu.getItems().addAll(aboutGame, separator, aboutMe);

    MenuBar menuBar = new MenuBar();
    menuBar.getMenus().addAll(fileMenu, helpMenu);

    return menuBar;
}
```

Inside Main.java

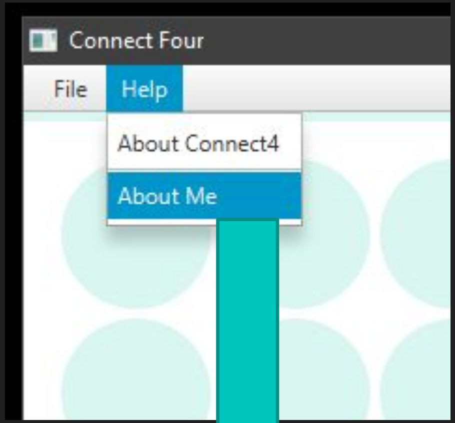


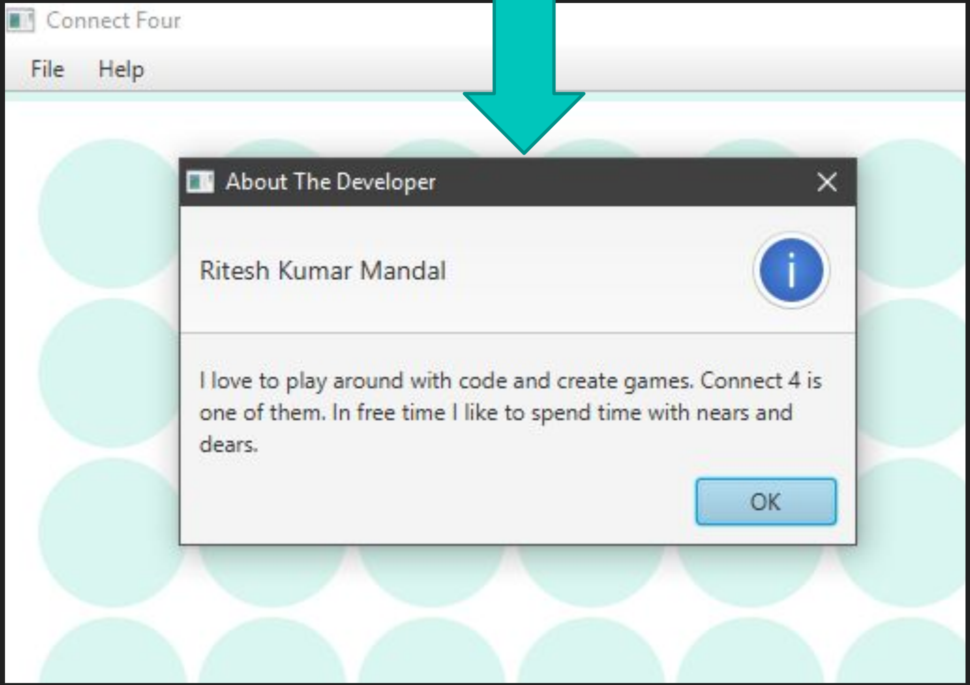
Alert Dialog Box



```java
private void aboutMe() {

    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("About The Developer");
    alert.setHeaderText("Ritesh Kumar Mandal");
    alert.setContentText("I love to play around with code and create games. " +
            "Connect 4 is one of them. In free time " +
            "I like to spend time with nears and dears.");

    alert.show();
}
```

Inside Main.java



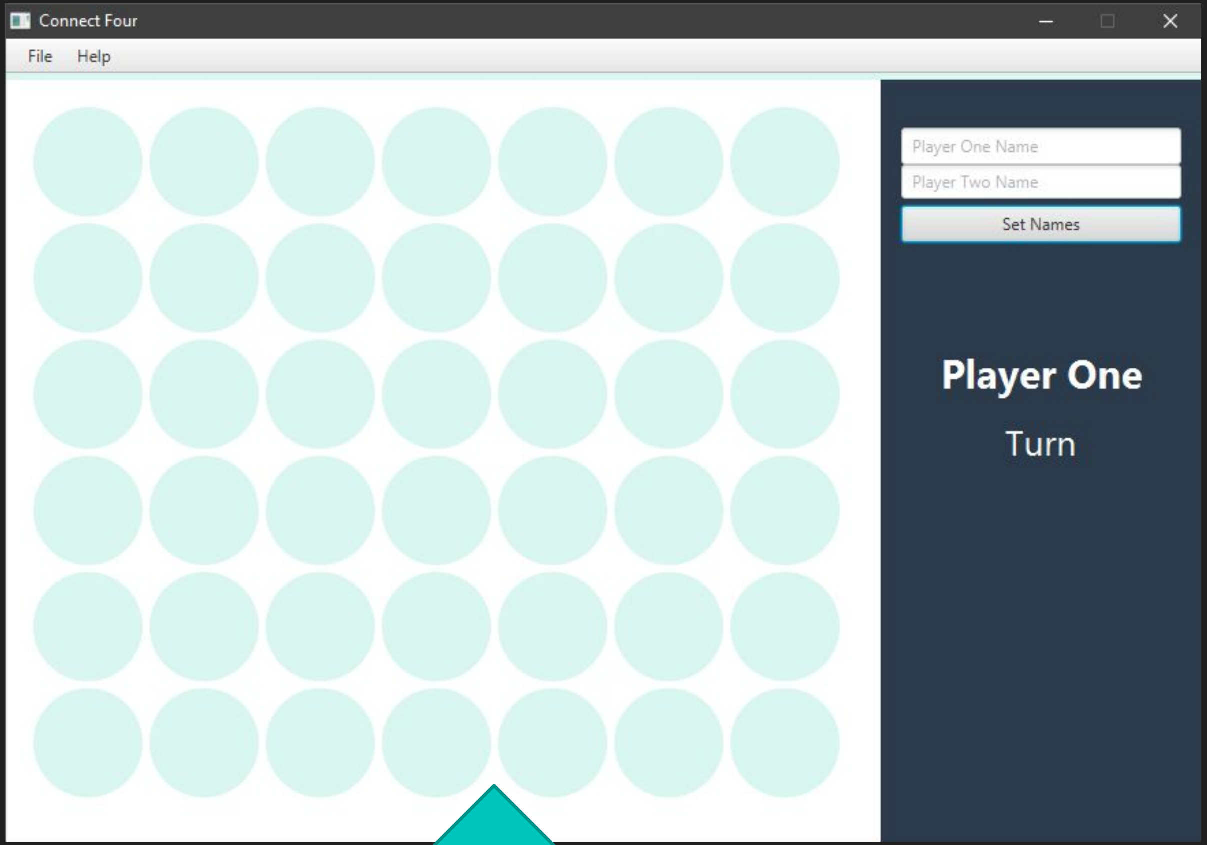


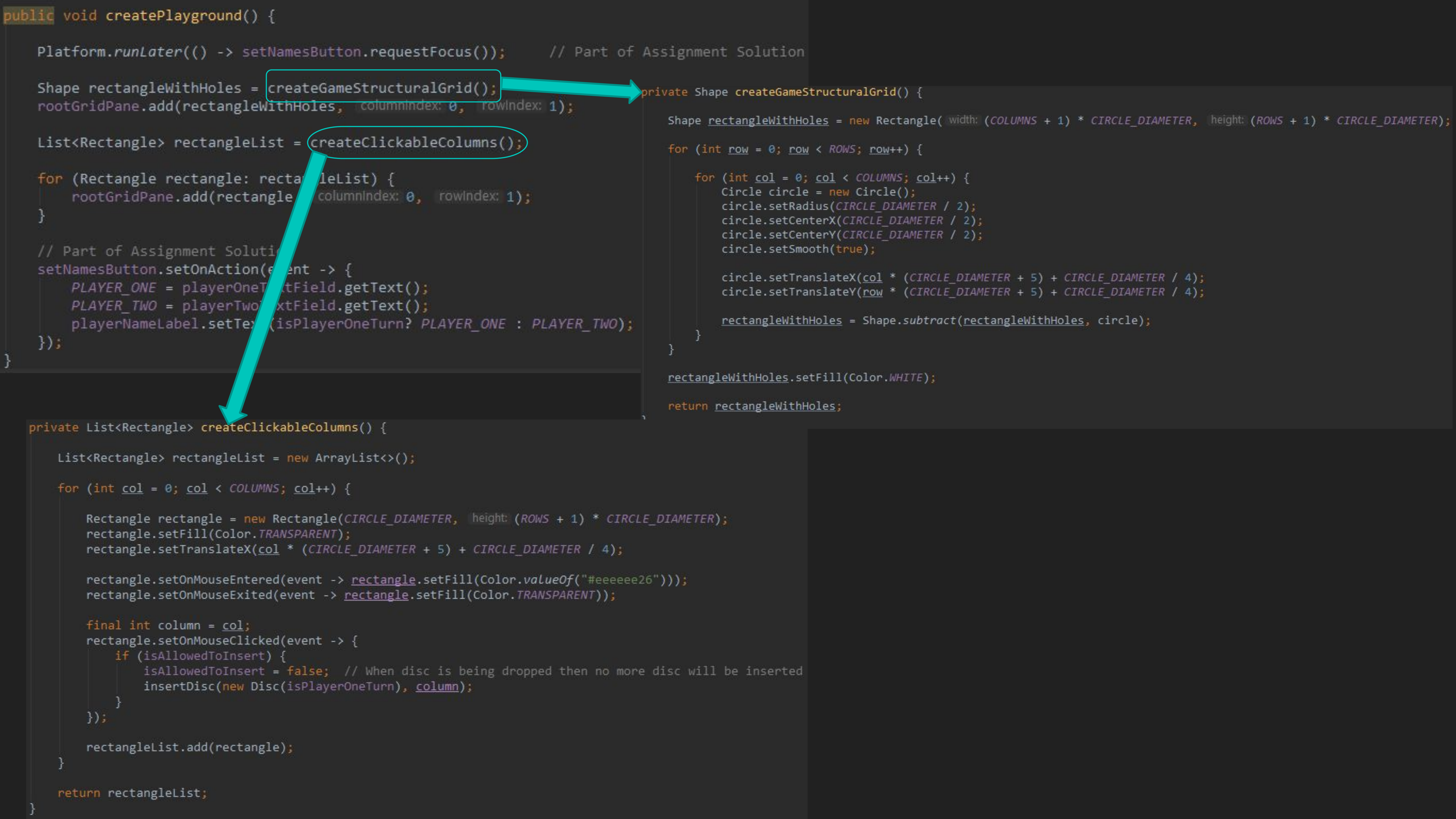
Ritesh Kumar Mandal

I love to play around with code and create games. Connect 4 is one of them. In free time I like to spend time with nears and dears.

Creating PlayGround

Inside Controller.java



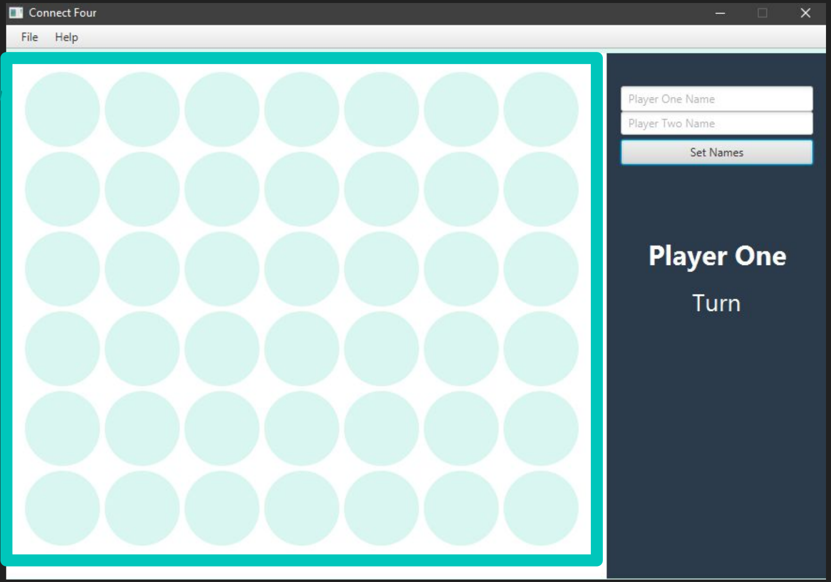


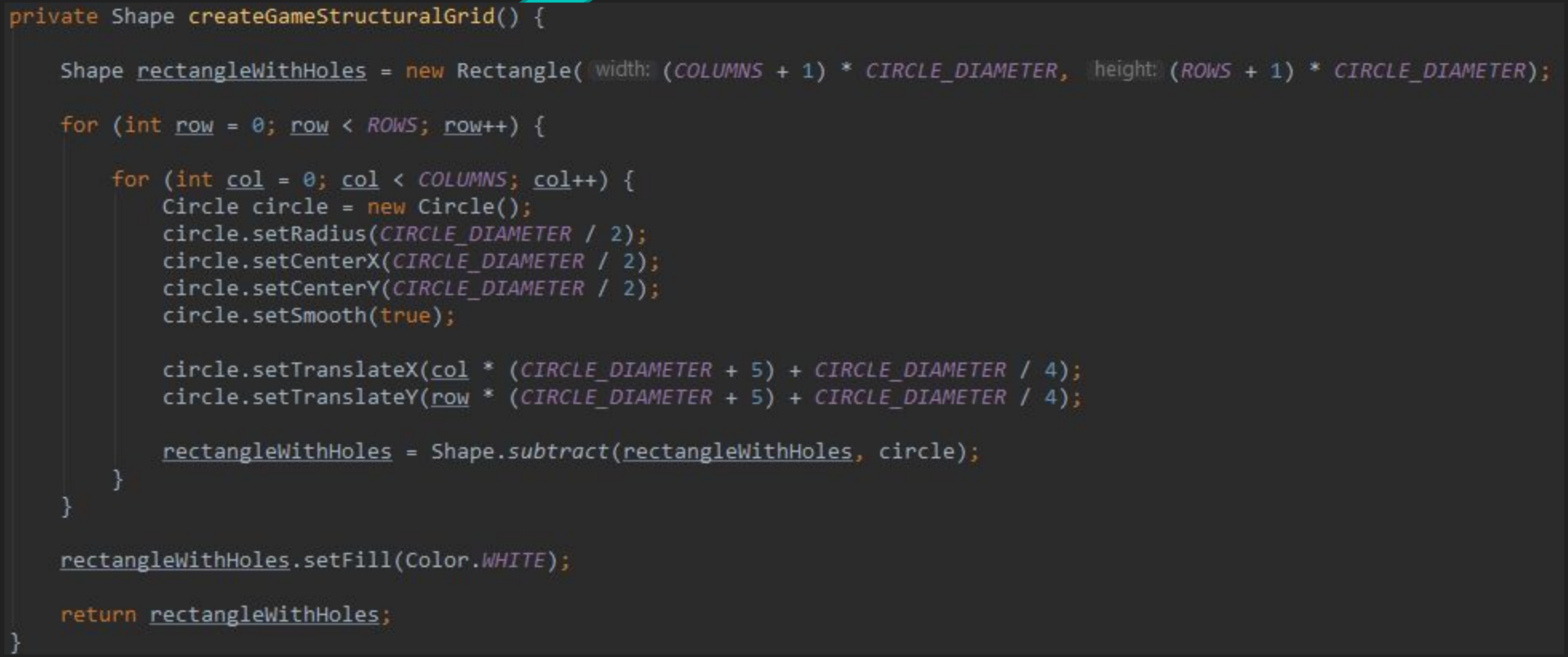
```
public void createPlayground() {

    Platform.runLater(() -> setNamesButton.requestFocus());     // Part of Assignment Solution

    Shape rectangleWithHoles = createGameStructuralGrid();
    rootGridPane.add(rectangleWithHoles,  columnIndex: 0,  rowIndex: 1);

    List<Rectangle> rectangleList = createClickableColumns();

    for (Rectangle rectangle: rectangleList) {
        rootGridPane.add(rectangle,  columnIndex: 0,  rowIndex: 1);
    }

    // Part of Assignment Solution
    setNamesButton.setOnAction(event -> {
        PLAYER_ONE = playerOneTextField.getText();
        PLAYER_TWO = playerTwoTextField.getText();
        playerNameLabel.setText(isPlayerOneTurn? PLAYER_ONE : PLAYER_TWO);
    });
}
```

```
private Shape createGameStructuralGrid() {

    Shape rectangleWithHoles = new Rectangle( width: (COLUMNS + 1) * CIRCLE_DIAMETER,  height: (ROWS + 1) * CIRCLE_DIAMETER);

    for (int row = 0; row < ROWS; row++) {

        for (int col = 0; col < COLUMNS; col++) {
            Circle circle = new Circle();
            circle.setRadius(CIRCLE_DIAMETER / 2);
            circle.setCenterX(CIRCLE_DIAMETER / 2);
            circle.setCenterY(CIRCLE_DIAMETER / 2);
            circle.setSmooth(true);

            circle.setTranslateX(col * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);
            circle.setTranslateY(row * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);

            rectangleWithHoles = Shape.subtract(rectangleWithHoles, circle);
        }
    }

    rectangleWithHoles.setFill(Color.WHITE);

    return rectangleWithHoles;
```

```
private List<Rectangle> createClickableColumns() {

    List<Rectangle> rectangleList = new ArrayList<>();

    for (int col = 0; col < COLUMNS; col++) {

        Rectangle rectangle = new Rectangle(CIRCLE_DIAMETER,  height: (ROWS + 1) * CIRCLE_DIAMETER);
        rectangle.setFill(Color.TRANSPARENT);
        rectangle.setTranslateX(col * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);

        rectangle.setOnMouseEntered(event -> rectangle.setFill(Color.valueOf("#eeeeee26")));
        rectangle.setOnMouseExited(event -> rectangle.setFill(Color.TRANSPARENT));

        final int column = col;
        rectangle.setOnMouseClicked(event -> {
            if (isAllowedToInsert) {
                isAllowedToInsert = false;  // When disc is being dropped then no more disc will be inserted
                insertDisc(new Disc(isPlayerOneTurn), column);
            }
        });

        rectangleList.add(rectangle);
    }

    return rectangleList;
}
```

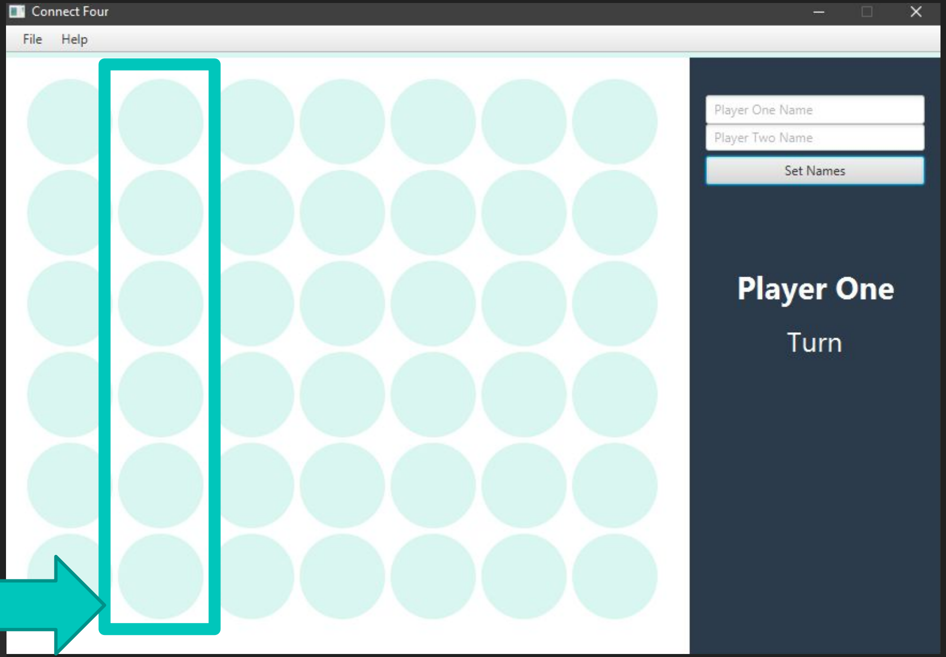
Creating Holes For Discs





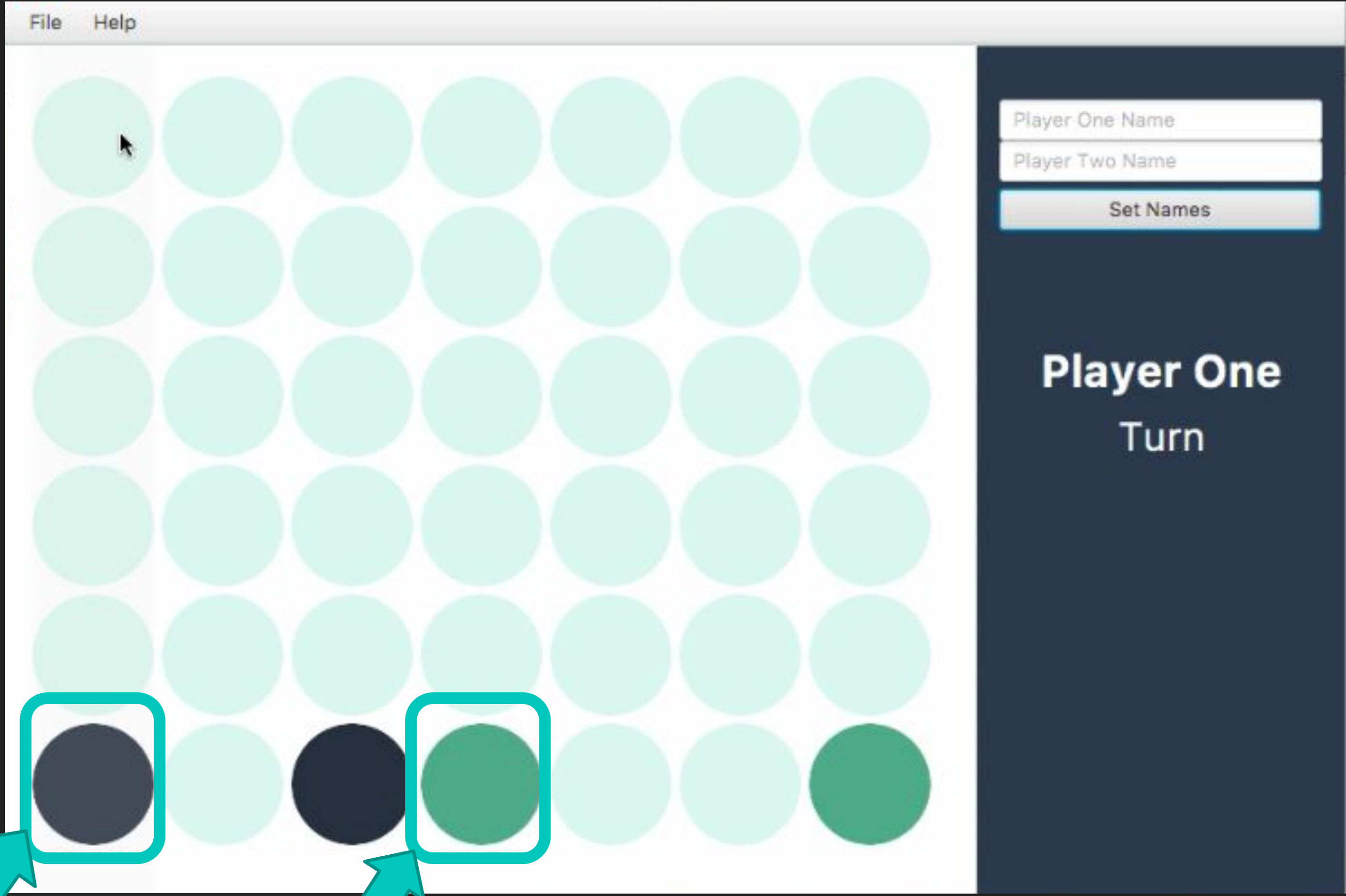
```
private Shape createGameStructuralGrid() {

    Shape rectangleWithHoles = new Rectangle( width: (COLUMNS + 1) * CIRCLE_DIAMETER,  height: (ROWS + 1) * CIRCLE_DIAMETER);

    for (int row = 0; row < ROWS; row++) {

        for (int col = 0; col < COLUMNS; col++) {
            Circle circle = new Circle();
            circle.setRadius(CIRCLE_DIAMETER / 2);
            circle.setCenterX(CIRCLE_DIAMETER / 2);
            circle.setCenterY(CIRCLE_DIAMETER / 2);
            circle.setSmooth(true);

            circle.setTranslateX(col * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);
            circle.setTranslateY(row * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);

            rectangleWithHoles = Shape.subtract(rectangleWithHoles, circle);
        }
    }

    rectangleWithHoles.setFill(Color.WHITE);

    return rectangleWithHoles;
}
```

Creating Clickable Columns to insert Circular Discs



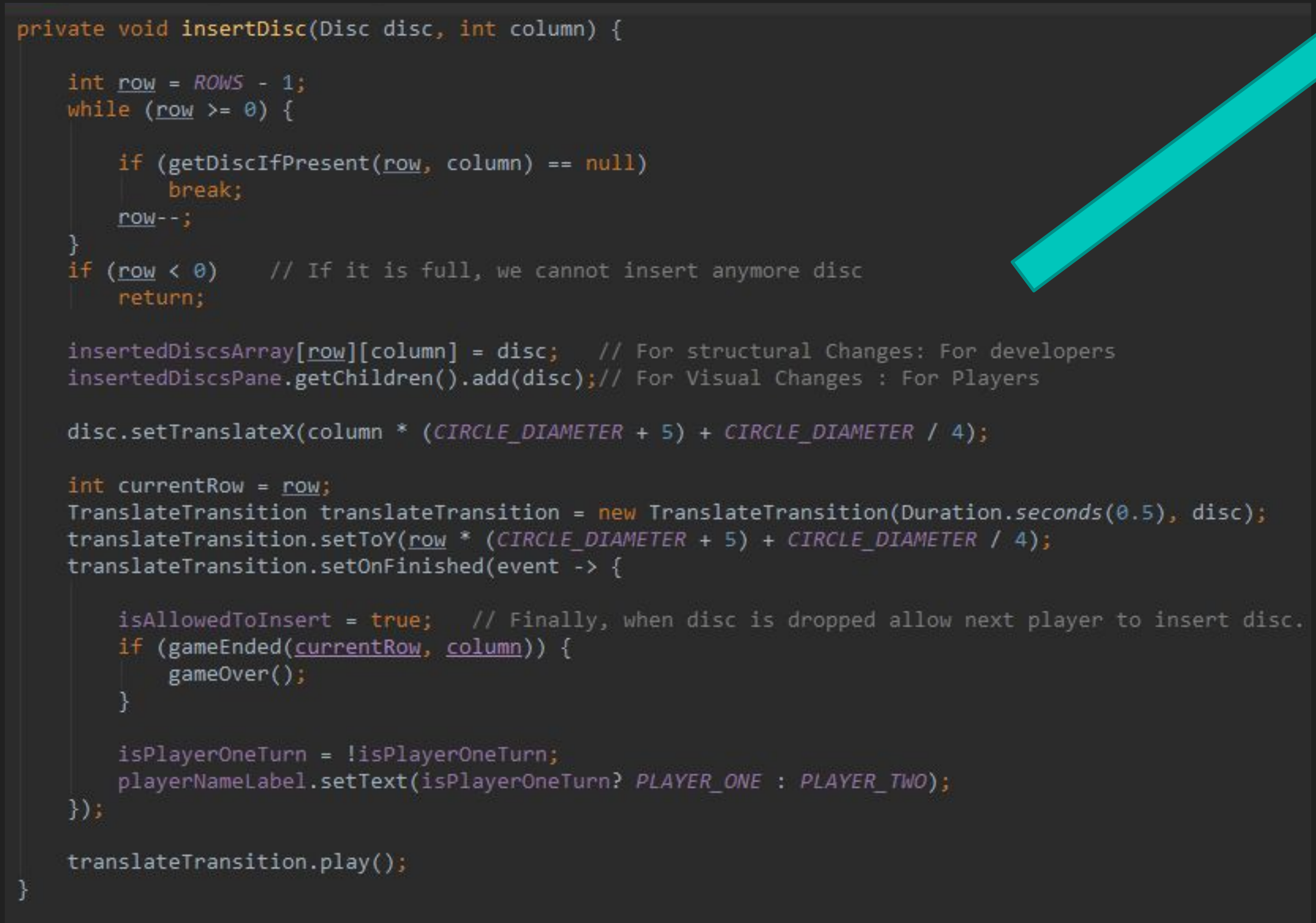
```java
private List<Rectangle> createClickableColumns() {

    List<Rectangle> rectangleList = new ArrayList<>();

    for (int col = 0; col < COLUMNS; col++) {

        Rectangle rectangle = new Rectangle(CIRCLE_DIAMETER,  height: (ROWS + 1) * CIRCLE_DIAMETER);
        rectangle.setFill(Color.TRANSPARENT);
        rectangle.setTranslateX(col * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);

        rectangle.setOnMouseEntered(event -> rectangle.setFill(Color.valueOf("#eeeeee26")));
        rectangle.setOnMouseExited(event -> rectangle.setFill(Color.TRANSPARENT));

        final int column = col;
        rectangle.setOnMouseClicked(event -> {
            if (isAllowedToInsert) {
                isAllowedToInsert = false;  // When disc is being dropped then no more disc will be inserted
                insertDisc(new Disc(isPlayerOneTurn), column);
            }
        });

        rectangleList.add(rectangle);
    }

    return rectangleList;
}
```

Disc Class

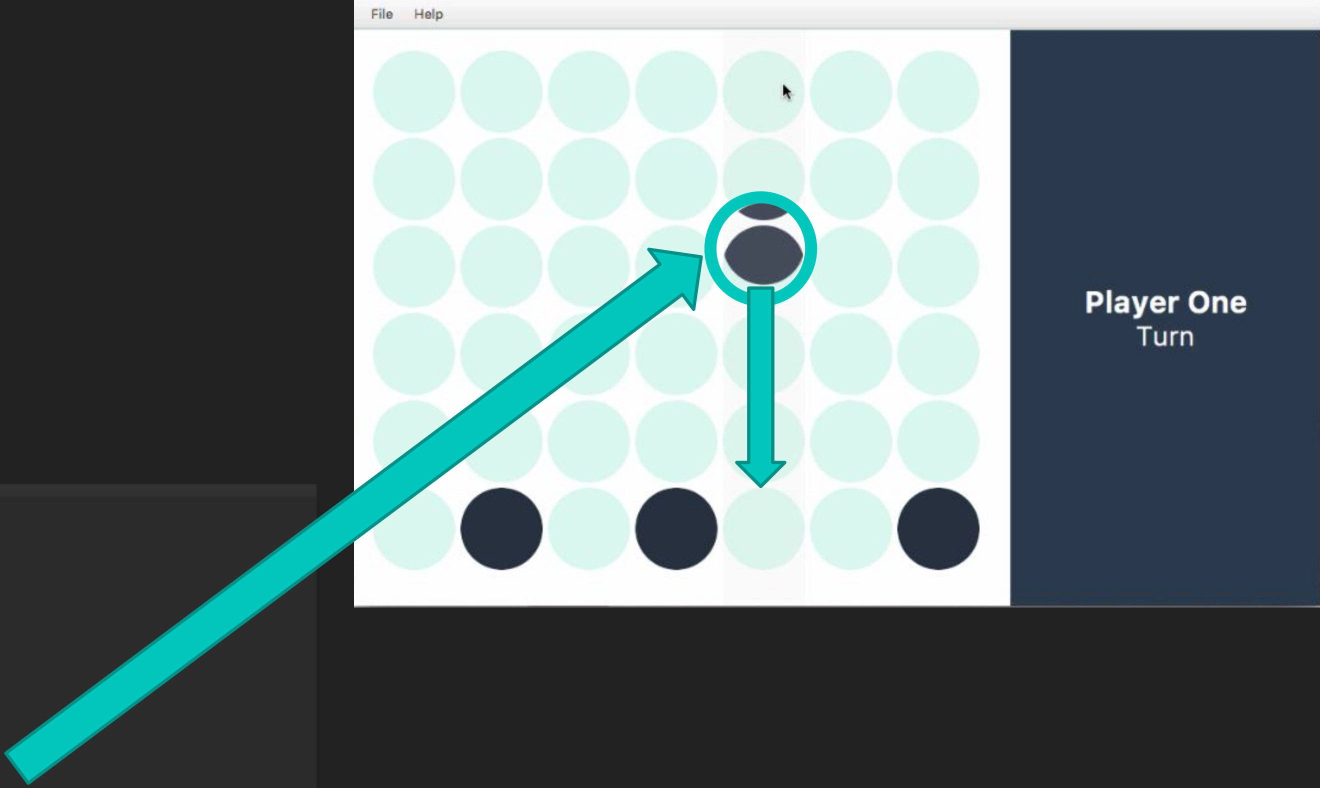


```
private static class Disc extends Circle {

    private final boolean isPlayerOneMove;

    public Disc(boolean isPlayerOneMove) {

        this.isPlayerOneMove = isPlayerOneMove;
        setRadius(CIRCLE_DIAMETER / 2);
        setFill(isPlayerOneMove? Color.valueOf(discColor1): Color.valueOf(discColor2));
        setCenterX(CIRCLE_DIAMETER/2);
        setCenterY(CIRCLE_DIAMETER/2);
    }
}
```

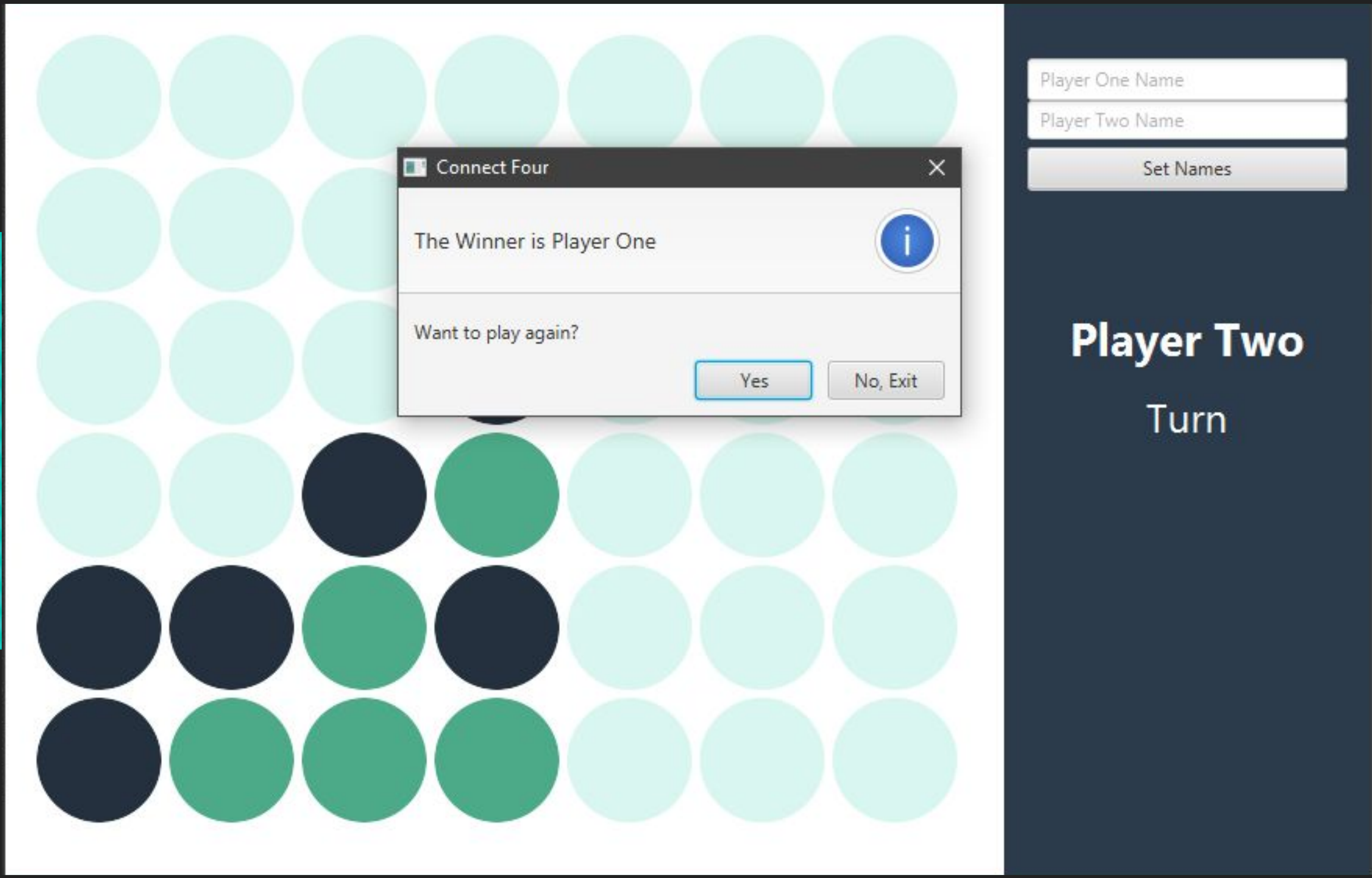
Inserting Discs Inside those clickable rectangles (Using Translation Animation)



```java
private void insertDisc(Disc disc, int column) {

    int row = ROWS - 1;
    while (row >= 0) {

        if (getDiscIfPresent(row, column) == null)
            break;
        row--;
    }
    if (row < 0)    // If it is full, we cannot insert anymore disc
        return;

    insertedDiscsArray[row][column] = disc;   // For structural Changes: For developers
    insertedDiscsPane.getChildren().add(disc);// For Visual Changes : For Players

    disc.setTranslateX(column * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);

    int currentRow = row;
    TranslateTransition translateTransition = new TranslateTransition(Duration.seconds(0.5), disc);
    translateTransition.setToY(row * (CIRCLE_DIAMETER + 5) + CIRCLE_DIAMETER / 4);
    translateTransition.setOnFinished(event -> {

        isAllowedToInsert = true;   // Finally, when disc is dropped allow next player to insert disc.
        if (gameEnded(currentRow, column)) {
            gameOver();
        }

        isPlayerOneTurn = !isPlayerOneTurn;
        playerNameLabel.setText(isPlayerOneTurn? PLAYER_ONE : PLAYER_TWO);
    });

    translateTransition.play();
}
```



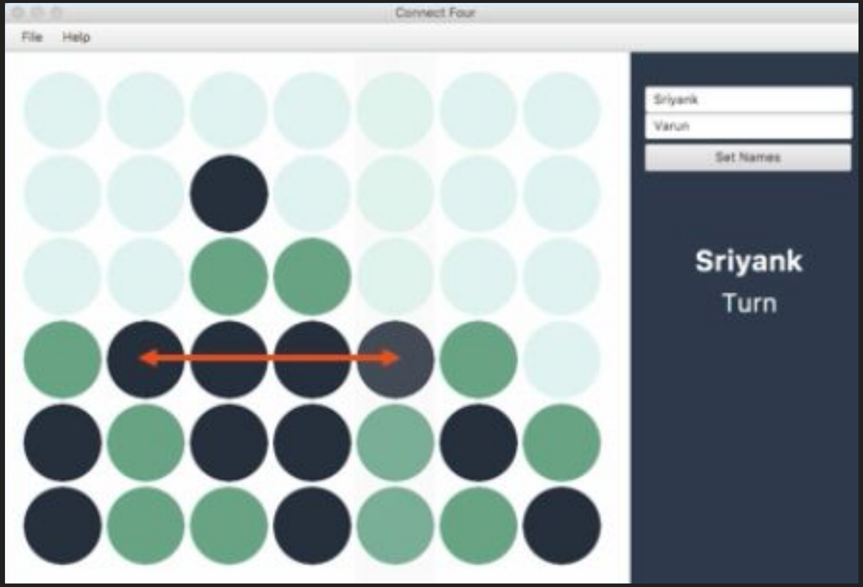
Decide The Winner

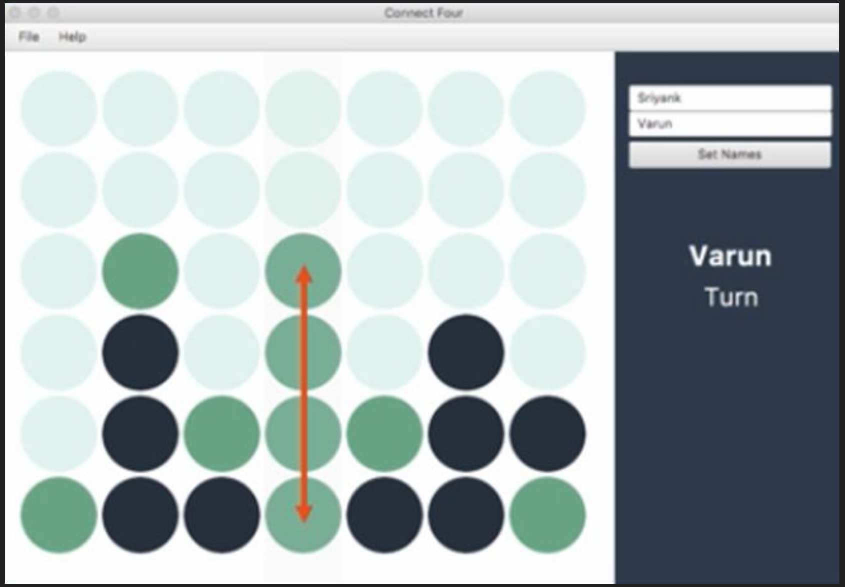


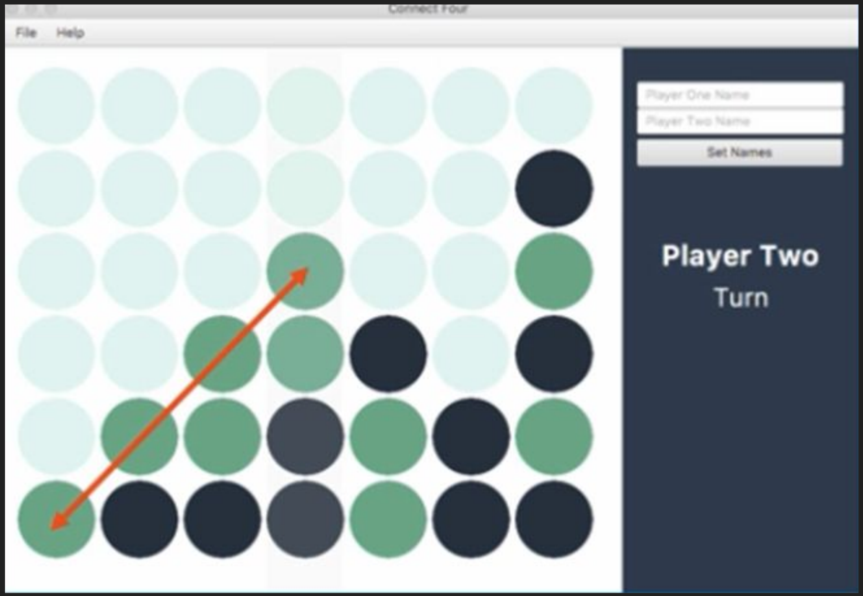
The Winner is Player One

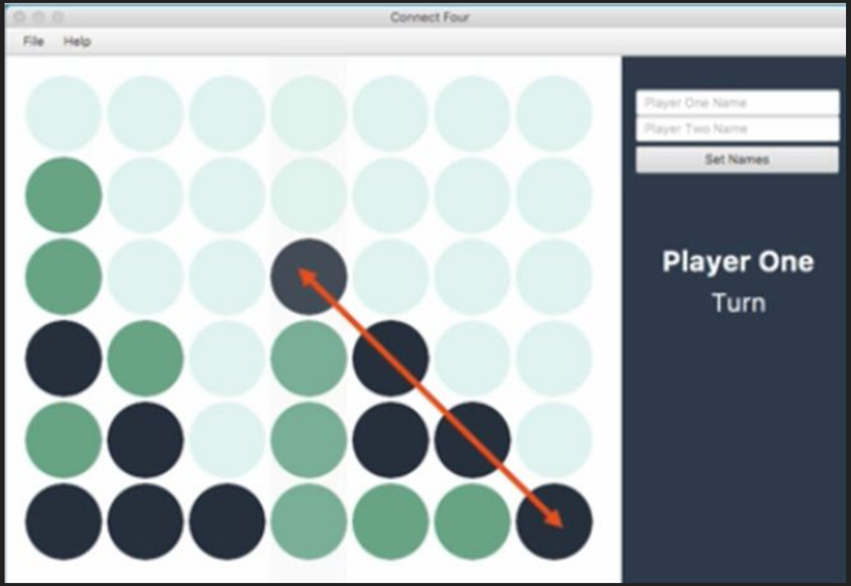
Want to play again?

Winning Criteria

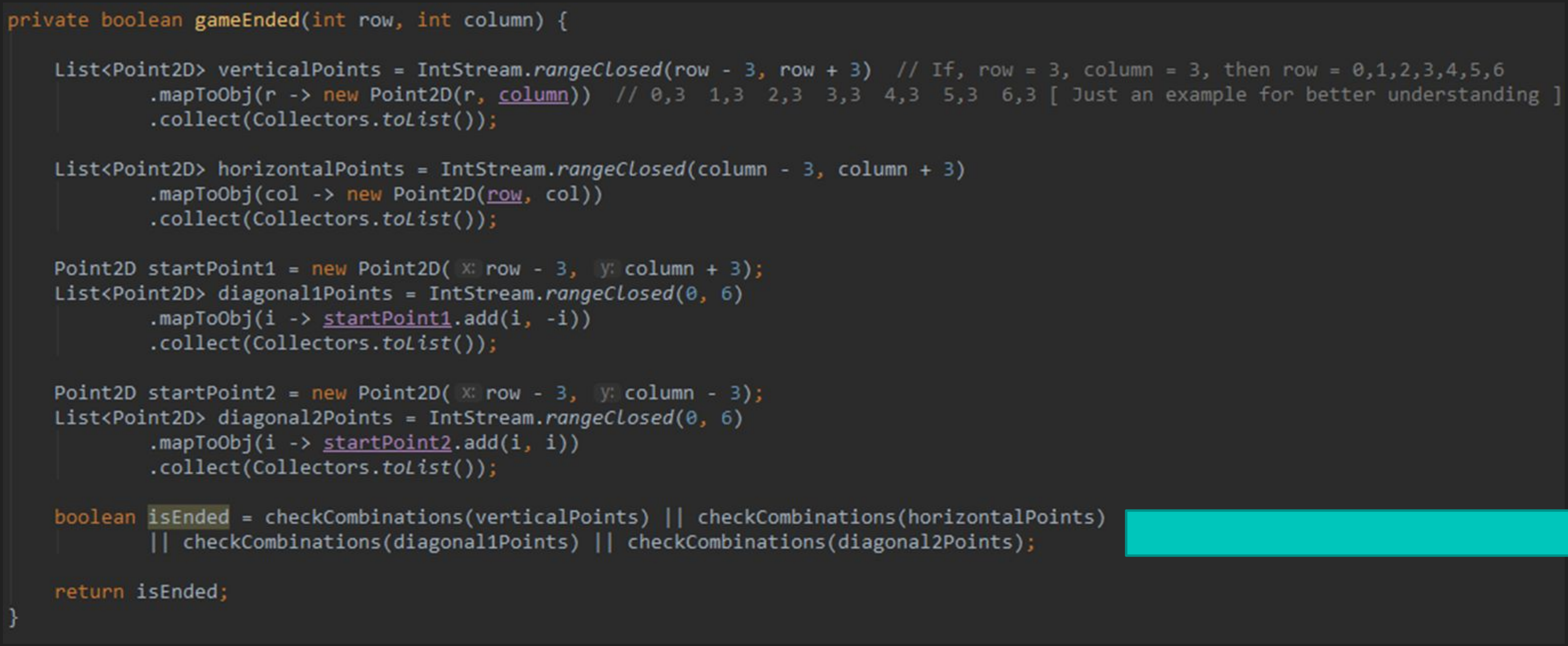






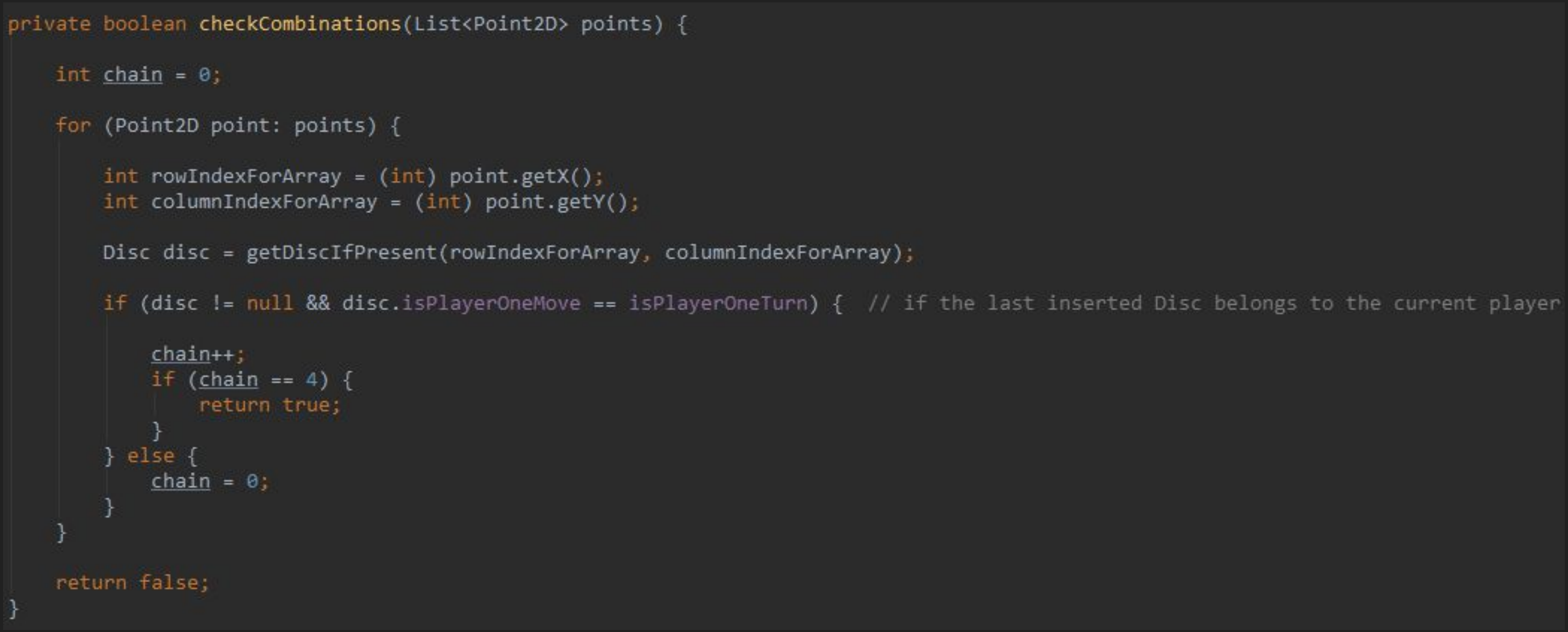


Ending The Game



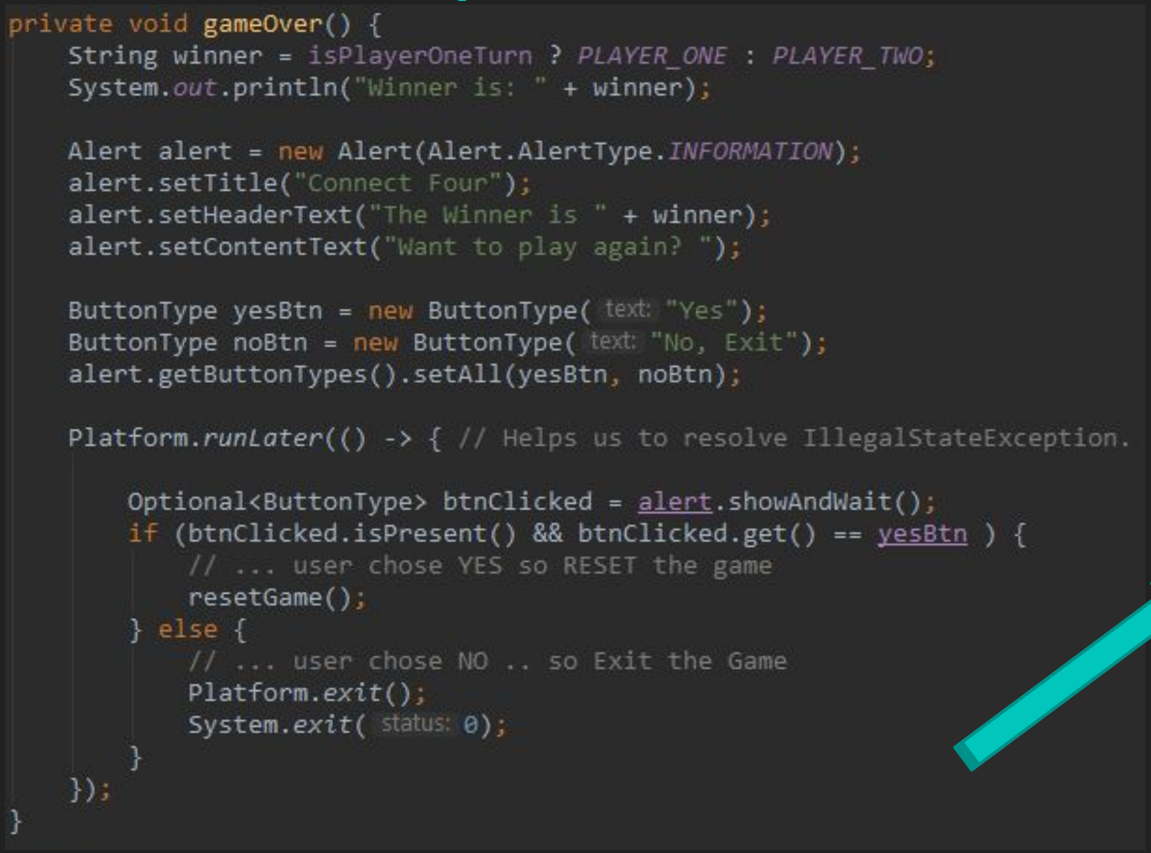
```java
private boolean gameEnded(int row, int column) {

    List<Point2D> verticalPoints = IntStream.rangeClosed(row - 3, row + 3)  // If, row = 3, column = 3, then row = 0,1,2,3,4,5,6
            .mapToObj(r -> new Point2D(r, column))  // 0,3  1,3  2,3  3,3  4,3  5,3  6,3 [ Just an example for better understanding ]
            .collect(Collectors.toList());

    List<Point2D> horizontalPoints = IntStream.rangeClosed(column - 3, column + 3)
            .mapToObj(col -> new Point2D(row, col))
            .collect(Collectors.toList());

    Point2D startPoint1 = new Point2D( x: row - 3,  y: column + 3);
    List<Point2D> diagonal1Points = IntStream.rangeClosed(0, 6)
            .mapToObj(i -> startPoint1.add(i, -i))
            .collect(Collectors.toList());

    Point2D startPoint2 = new Point2D( x: row - 3,  y: column - 3);
    List<Point2D> diagonal2Points = IntStream.rangeClosed(0, 6)
            .mapToObj(i -> startPoint2.add(i, i))
            .collect(Collectors.toList());

    boolean isEnded = checkCombinations(verticalPoints) || checkCombinations(horizontalPoints)
            || checkCombinations(diagonal1Points) || checkCombinations(diagonal2Points);

    return isEnded;
}
```

Checking The Combinations

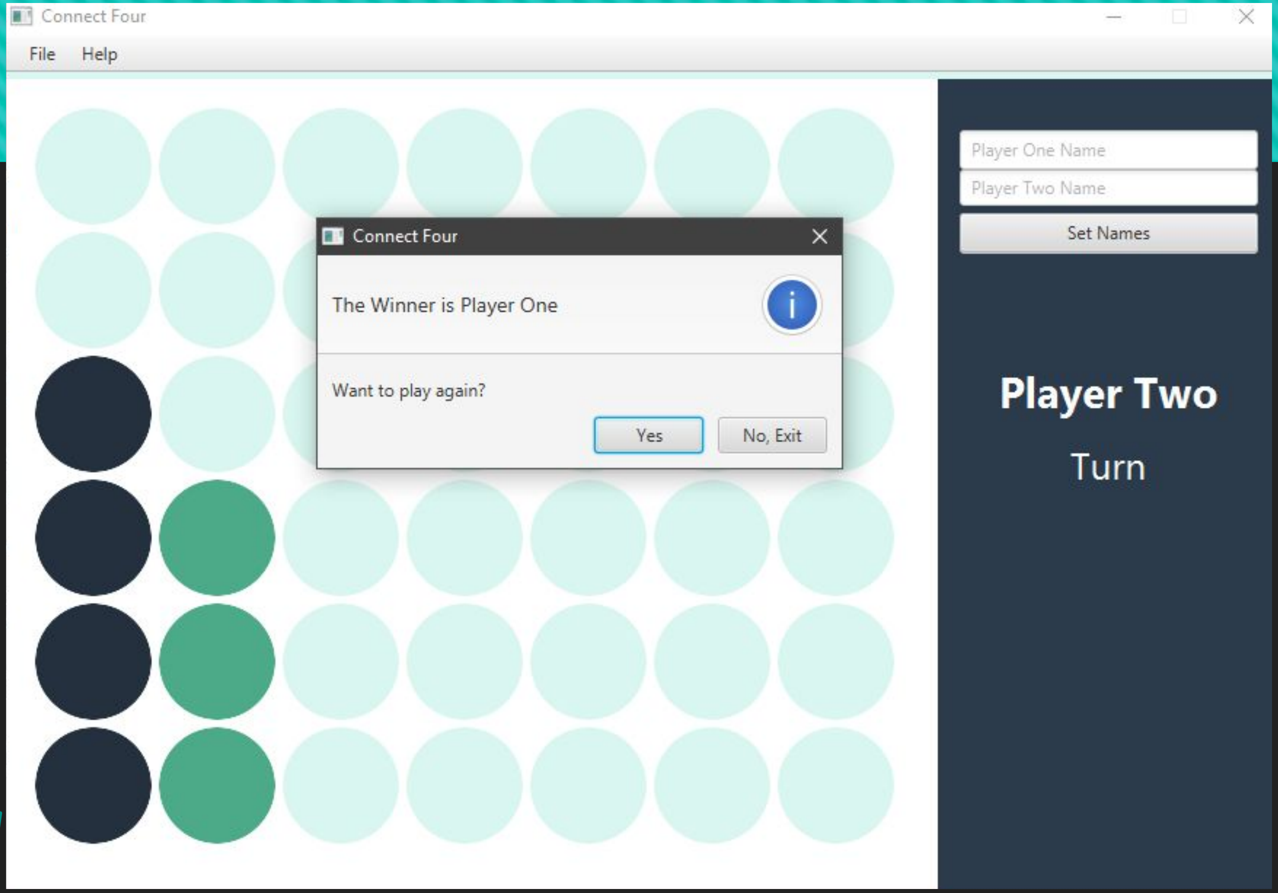


```java
private boolean checkCombinations(List<Point2D> points) {

    int chain = 0;

    for (Point2D point: points) {

        int rowIndexForArray = (int) point.getX();
        int columnIndexForArray = (int) point.getY();

        Disc disc = getDiscIfPresent(rowIndexForArray, columnIndexForArray);

        if (disc != null && disc.isPlayerOneMove == isPlayerOneTurn) {  // if the last inserted Disc belongs to the current player

            chain++;
            if (chain == 4) {
                return true;
            }
        } else {
            chain = 0;
        }
    }

    return false;
}
```

GAME OVER



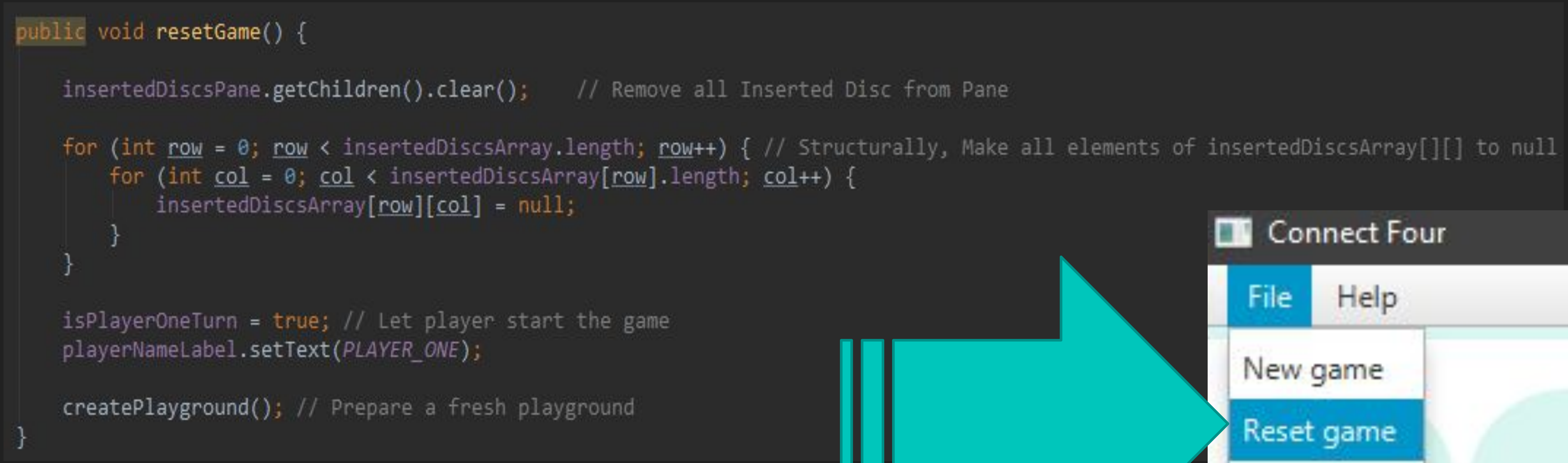
```java
private void gameOver() {
    String winner = isPlayerOneTurn ? PLAYER_ONE : PLAYER_TWO;
    System.out.println("Winner is: " + winner);

    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Connect Four");
    alert.setHeaderText("The Winner is " + winner);
    alert.setContentText("Want to play again? ");

    ButtonType yesBtn = new ButtonType( text: "Yes");
    ButtonType noBtn = new ButtonType( text: "No, Exit");
    alert.getButtonTypes().setAll(yesBtn, noBtn);

    Platform.runLater(() -> { // Helps us to resolve IllegalStateException.

        Optional<ButtonType> btnClicked = alert.showAndWait();
        if (btnClicked.isPresent() && btnClicked.get() == yesBtn ) {
            // ... user chose YES so RESET the game
            resetGame();
        } else {
            // ... user chose NO .. so Exit the Game
            Platform.exit();
            System.exit( status: 0);
        }
    });
}
```



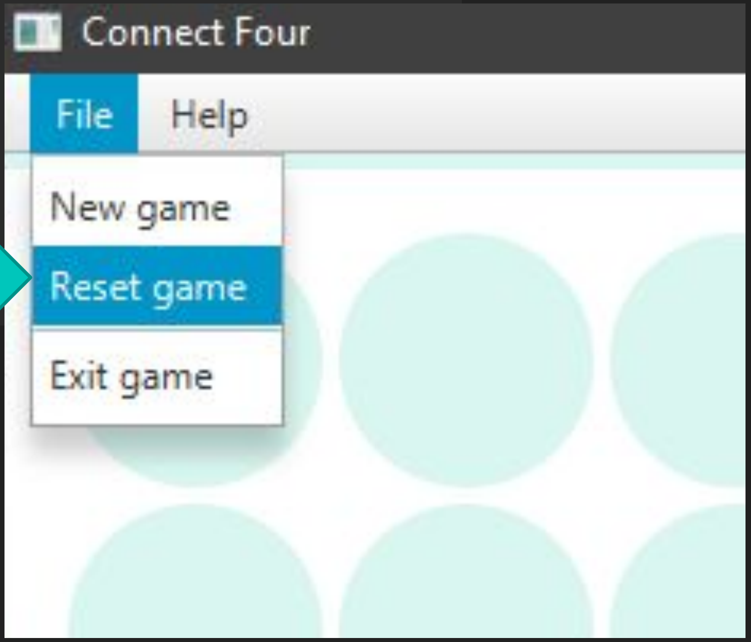
The Winner is Player One

Want to play again?

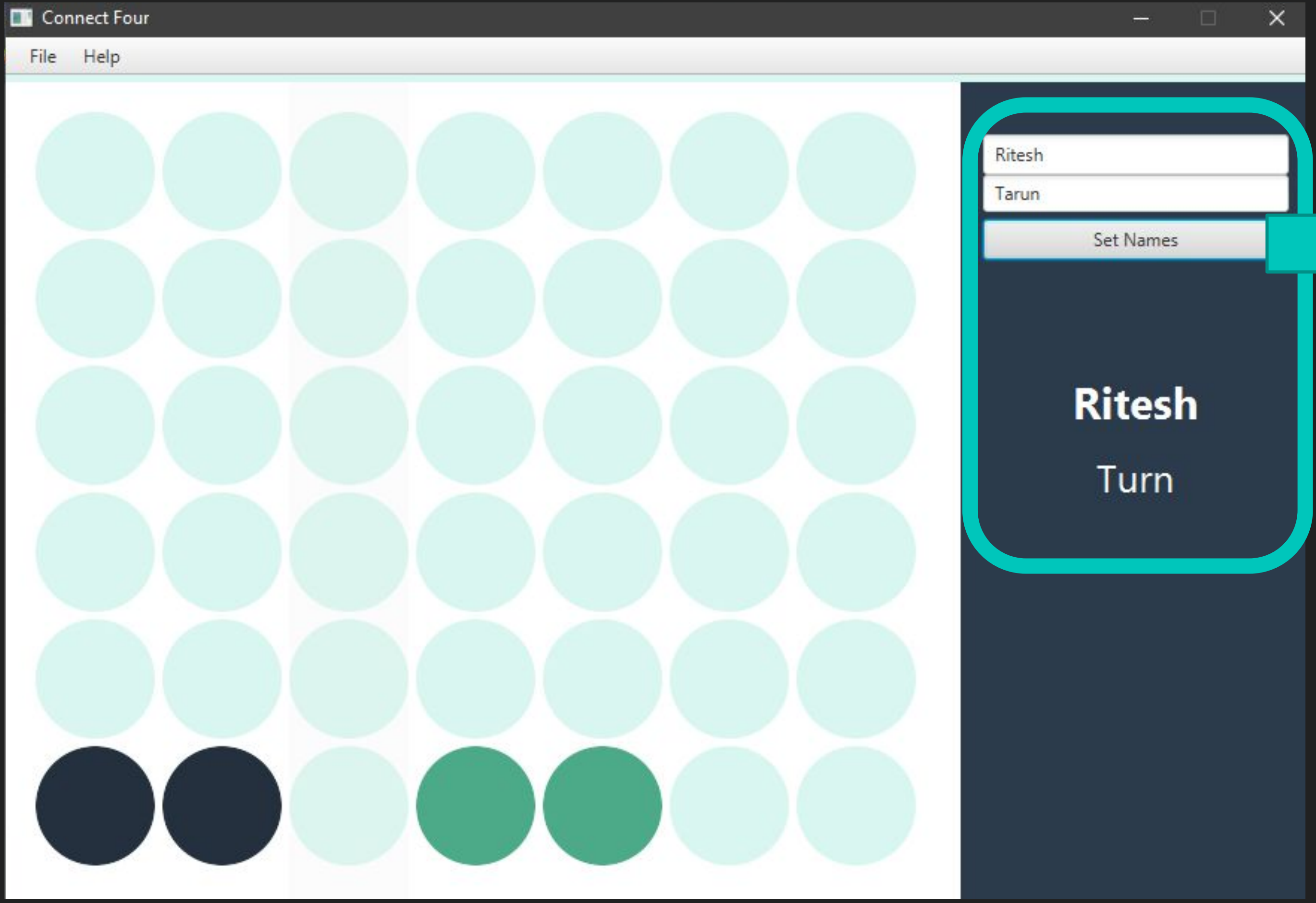
New Game & Reset Game

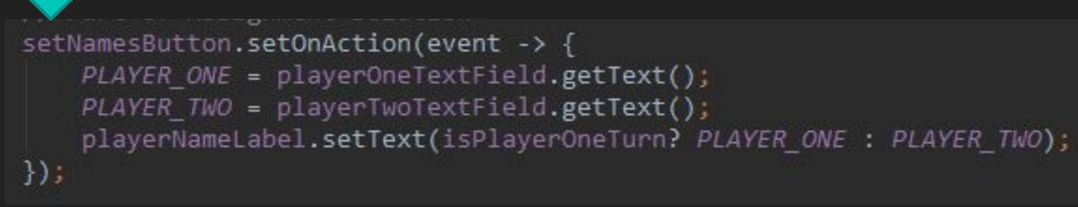


```
public void resetGame() {

    insertedDiscsPane.getChildren().clear();    // Remove all Inserted Disc from Pane

    for (int row = 0; row < insertedDiscsArray.length; row++) { // Structurally, Make all elements of insertedDiscsArray[][] to null
        for (int col = 0; col < insertedDiscsArray[row].length; col++) {
            insertedDiscsArray[row][col] = null;
        }
    }

    isPlayerOneTurn = true; // Let player start the game
    playerNameLabel.setText(PLAYER_ONE);

    createPlayground(); // Prepare a fresh playground
}
```



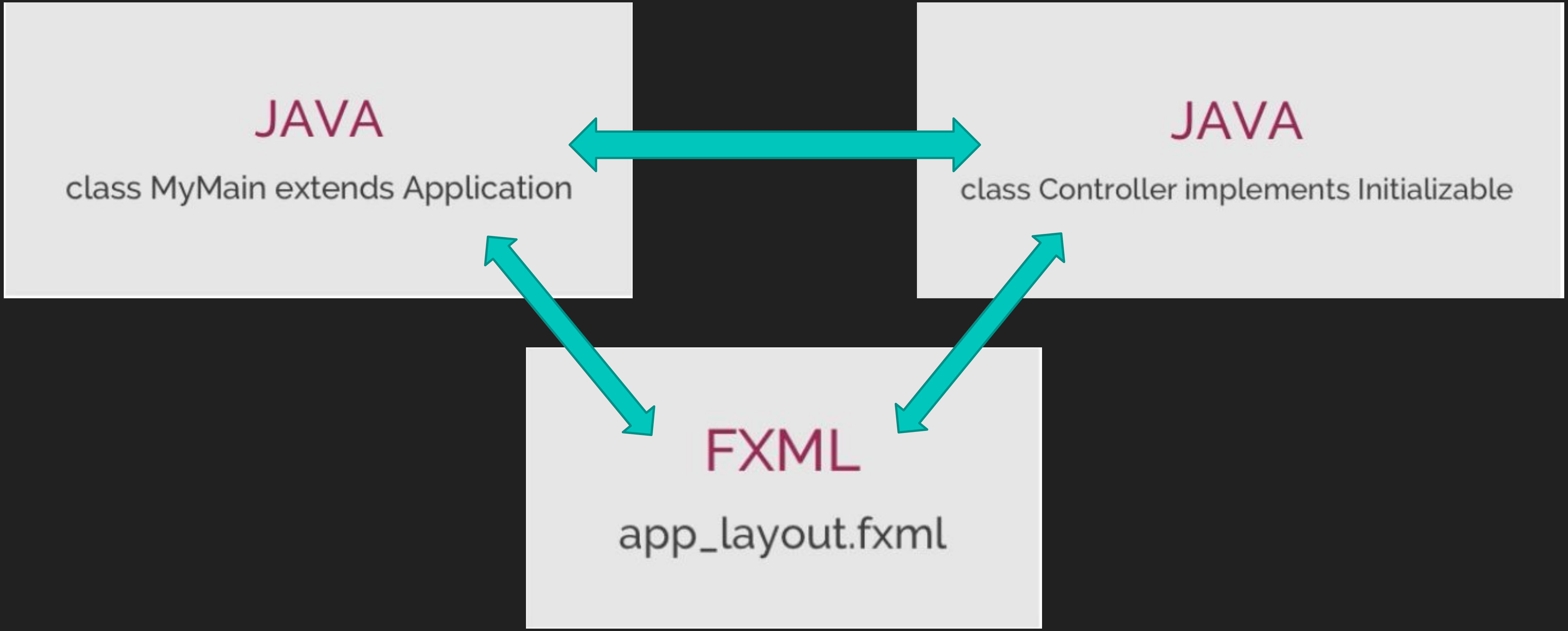
Setting Players Name





```
setNamesButton.setOnAction(event -> {
    PLAYER_ONE = playerOneTextField.getText();
    PLAYER_TWO = playerTwoTextField.getText();
    playerNameLabel.setText(isPlayerOneTurn? PLAYER_ONE : PLAYER_TWO);
});
```

JavaFX Basic Project Structure


JAVA
class MyMain extends Application
JAVA
class Controller implements Initializable
FXML
app_layout.fxml