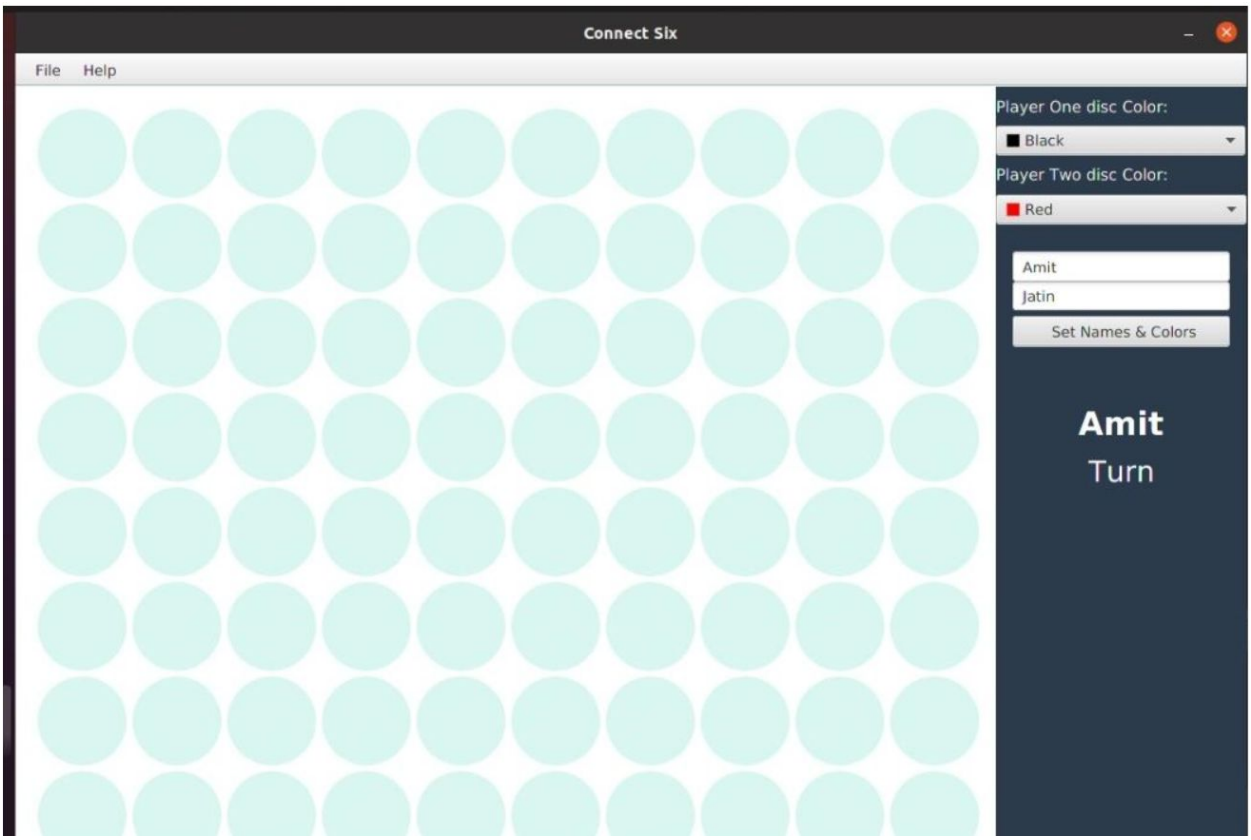
Core Java Contest Problem Statement

Hey! It was really a long journey that we successfully covered so far in this course.
Initially we covered all the core java concepts and did a wonderful project by
developing a game in JavaFX platform.

The Connect 4 game needs some improvement in terms of user experience. We will now make a Connect 6 game with a few more modifications. The modifications are listed below.

1. Both players should be able to enter their names and choose a color of the disc of their choice.
2. The names of players cannot be the same.
3. The colors picked by the players cannot be the same.
4. There should be 10 columns and 8 rows in the game.
5. A player wins when 6 discs of the same color are aligned in horizontal, vertical and diagonal (left diagonal and right diagonal).
6. Player names should be shown turn wise.
7. There should be a dialog box showing who won with Player Name and Color code.
8. The UI of the game should look something like the screenshot below.



Currently, the app displays either PLAYER ONE or PLAYER TWO
based on whose turn it is. So, you need to add two TextFields where users can enter their name and henceforth their names should be displayed in place of PLAYER ONE or PLAYER TWO text in playeNameLabel in the application.

Approach for Contest Problem Statement:

Follow the following steps and complete the task.

****The following steps are only to guide you on JavaFX and do not cover the logical part of the project.**

1. Go to Scene Builder of your game.fxml, select the Region present in VBox and
delete it. In place of Region we'll insert our Labels,TextFields and a Button.

2. Once Region is deleted, in place of that add Two labels Player One disc Color and Player Two disc Color, Two TextFields, and a Button. Now your layout structure should look similar to this:

3. Modify the properties of the Two TextFields and the Button. Stretch the TextFields
and Button so that they occupy the whole space from left to right within the VBox.
Along with this modify the Prompt Text and the Button Text.

4. Along with names, the player can also choose a color of their choice of the disc. The colors can be chosen from color picker in PLAYER ONE DISC COLOR and PLAYER TWO DISC COLOR. The color picker can be fetched from the control functionality of game.xfml. Refer to the screenshot below to know how the

UI of the application should look like and screenshot in the above mentioned point to see the control functionality.

4. Give the required Margin for TextFields, Button and PLAYER ONE Labe and Player Disc Color. You can use your own margin values or you can take help from the following image:

5. Finally, assign the fx:id to the Two TextFields and Colors and the Button as playerOneTextField, playerTwoTextField, playerOneColor, playerTwoColor and setNamesandColors Button respectively. In case of any confusion check the above mentioned image for help.

So far the things look perfect in our game.fxml file.

6. Now, let's come to Controller.java and declare all the Controls such as playerOneTextField, playerTwoTextField, playerOneColor, playerTwo Color and setNamesandColorsButton with @FXML annotation. Make sure they all have public modifier. For example:

```
@FXML
public TextField playerOneTextField, playerTwoTextField;    // Part of Assignment Solution

@FXML
public Button setNamesAndColorButton;   // Part of Assignment Solution

@FXML
public ColorPicker playerOneColor,playerTwoColor;
```

7. Within the createPlayground() method use setOnAction() method to handle the click event of setNamesButton. When click event on the button is fired write your logic to set the user input names to the String variable of PLAYER_ONE and PLAYER_TWO and color picker for Player One and Player Two.

```
setNamesAndColorButton.setOnAction(event -> {
```

8. Run your app and test your code.

Final Deliverables:

1. Jar file of your new Connect 6 app.
2. Final Connect 6 game source code.

Before you submit, make sure to zip all the above mentioned deliverables within a single folder.

Evaluation Rubric

Element	Categories	Sub-categories	Below Average	Average	Good Work
Code	**Naming Convention**		Class Names not in capital Variable names not in lower_case	Cases of class name and variables taken into account but meaningful names not used	Meaningful names taking into consideration the casing of letters
	Use of Methods		Less than 10 methods used	Number of methods lie in the range of (10, 13)	Number of methods are in the range [13, ...)
	Java8 features			No use of lambda function	Use of lambda function
Concepts	**OOPS Concepts**	**Classes**	No use of controller or disc class	Use of controller class but missed disc class	Use of both controller and disc class
		Objects	Less than 10 objects	Objects in the range of (10,12)	Objects in range [12, ...)
		Inheritance	No Inheritance	Inheritance but not correctly	Correct Inheritance
	Collections		No use of Arraylist and 2D array	Used Arraylists and 2D arrays but incorrectly	Arraylists and 2D arrays used to perfection
	Keywords		Incorrect usage of static, new and this	Used the keywords correctly but not at appropriate places.	Good use of keywords.
Application	**Flow**	**Players Name**	Default names, no names picked	One one name is picked	Both names picked

		Disc Color	Disc color cannot be chosen	Only one player can choose disc color	Both players can choose disc color
		Disc Insertion	Discs do not insert	Discs inserted with some exceptions	Discs insert smoothly
		Winner Decision	Winner not decided	Incorrect winner decision	Correct winner decision
		Double Insertion		Double insertion take place	No Double insertion
	Assignment		Textbox and Color Labels not made	Textboxes made but names' sequence not correct Color Label works but is not assigned to the chosen player.	Textbox and Color Labels work perfectly
			Same name can be chosen	Only one name can be chosen	Same name for both players cannot be chosen
			Same disc color for both players can be chosen	Only one disc color can be chosen for one player	Same disc color cannot be chosen for both players.
	Jar File			Not made	Made