Checkers Checkerboard

Description: Create a Checkerboard class that generates a board for the game of checkers in JavaFX using an AnchorPane as the container for the Rectangles that make up the "squares" on the board. Place the AnchorPane containing the checker board in a UI built in FXML that has menus to select the grid size and color scheme. Allow the Stage size to be changed by the user causing the size of the board to change with the size of the Stage.

Purpose: This challenge provides experience in building a class to generate a JavaFX UI. It provides experience with JavaFX UI hierarchies and in creating an algorithm to generate the checkerboard layout.

Requirements:

Project Name: Checkers

For all challenges in this course you are to use a naming scheme that incorporates your pawprint into the name of the project. The project name is to be preceded by your pawprint with the first letter capitalized. Example if pawprint is abcxyz9 then the project name is to be: Abcxyz9Checkers. You must follow this naming scheme to receive credit for your work. The name of the project in Java will also be the name of the main class. Class names are camel-cased with the first letter upper-case.

IDE: NetBeans
Language: Java
JDK: 8
UI: JavaFX – JavaFX objects for the CheckerBoard class and FXML/Scene Builder to create UI that displays the checkerboard and allows the user to make selections.

In this challenge, you are to create a CheckerBoard class that builds a UI hierarchy that is the board in the game of checkers based on an AnchorPane that contains Rectangles. The following are examples of boards generated by the CheckerBoard class.

Checkers Checkerboard

Figure 1. Default Colored Board (Color.RED / Color.BLACK)

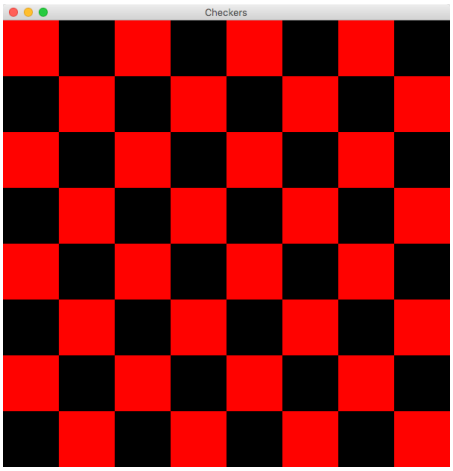
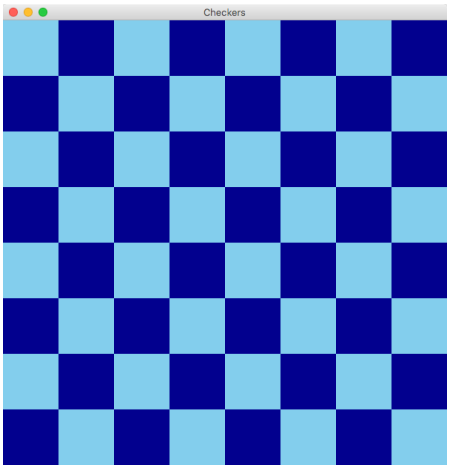


Figure 2. Custom Color Scheme (Color.SKYBLUE/Color.DARKBLUE)



Checkers / English draughts: https://en.wikipedia.org/wiki/English_draughts

The default colored board is generated when colors are not specified via the constructor. Boards are comprised of squares that are a dark color or a light color. For the default colored board the light color is Color.RED and the dark color is Color.BLACK.

A normal board is 8 x 8. The top left square is a light color and in each direction the squares alternate between a light color and a dark color. CheckerBoard instances, via two constructors, are to be configurable in the following ways:

- Number of rows and number of columns.
- Board width and the board height in pixels.
- Colors for the light color and the dark color.
- One constructor allows the colors to be specified and the other does not. If the colors
- are not specified, the default board colors are to be used: Color.RED and Color.BLACK.

Checkers Checkerboard

CheckerBoard Class Requirements:

The following are requirements for the CheckerBoard class.

- Its fields are to be private.
- It has two constructors.
    - First constructor receives: int numRows, int numCols, double boardWidth, double boardHeight
    - Second constructor receives: int numRows, int numCols, double boardWidth double boardHeight, Color lightColor, Color darkColor
- Code is not to be duplicated in the constructors. One constructor is to use the other constructor.
- It has a public build() method that builds the board UI and returns an AnchorPane as the root object.
- It has a public getBoard() method that returns the AnchorPane generated by build() or null if one hasn't been built yet.
- It has the following additional public getters: getNumRows(), getNumCols(), getWidth(), getHeight(), getLightColor(), getDarkColor(), getRectangleWidth(), getRectangleHeight()

Building the board UI:

The following is information about building the board UI.

The board is to be procedurally generated from the data held in the CheckerBoard fields and is to fill the height and width. The UI is comprised of an AnchorPane that is the root object that contains Rectangle objects that represent the "squares." The colors of the Rectangle objects follow the rules for a checker board: the top left "square" is a light color and in each direction the "squares" alternate between a light color and a dark color. The result is to look like the examples shown in Figures 1 and 2. The algorithm for determining the color choice for each "square" (light or dark) as it is generated should be intelligently implemented. Hint: there is no need for flags or flip-flopping states.

Building the host UI:

The CheckerBoard class is used to build an AnchorPane that needs to be displayed in the UI of the application. The host UI for the CheckBoard AnchorPane is to be created using FXML/Scene Builder. The UI is to contain a MenuBar with the following Menus and MenuItems:

- Grid
    - 16 x 16
    - 10 x 10
    - 8 x 8
    - 3 x 3
- Colors

- Default (The default which is Color.RED / Color.BLACK)
- Blue (Color.SKYBLUE / Color.DARKBLUE)

Note: the info in parentheses are not be shown in the MenuItem. They are there to indicate what colors you are to use for the Rectangles.

Below the MenuBar the AnchorPane created by CheckerBoard is to be displayed. Typically, the way to do this is to use a VBox to hold the MenuBar and the AnchorPane.

When the user chooses a grid size from the Grid Menu that size checkerboard is to be displayed. When the user chooses a color scheme from the Colors menu the board is to be rendered with Rectangles with the colors indicated in the parentheses after the color scheme.

The size of the AnchorPane for the checker board is to be based on the size of the area in the Stage below the MenuBar. When the Stage resizes the checkerboard is to resize.

Submission:

Submit a zip file of your NetBeans project. This means your are to submit a zip file of the **entire project folder**. The name of the zip file is to be the name of the project folder. So, if your pawprint is abcxyz9 then the zip file would be named Abcxyz9Checkers.zip.