Project 04, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Graphical User Interface and Exception

In this project, we will design a checkerboard graphical user interface (GUI), not necessarily functional, that can draw a board along with the checker pieces on that board. The main part of the project involves writing code to draw the checkerboard. We are not going to worry about making it functional that can be played by two players. You can make it functional, two human players can play the game, if you would like to receive **bonus credit**. This bonus credit is worth 50 points (a half of a complete assignment) that will give you an opportunity to improve your grade.

Please review the following resources, read the rules and watch a video, before you start working on it.

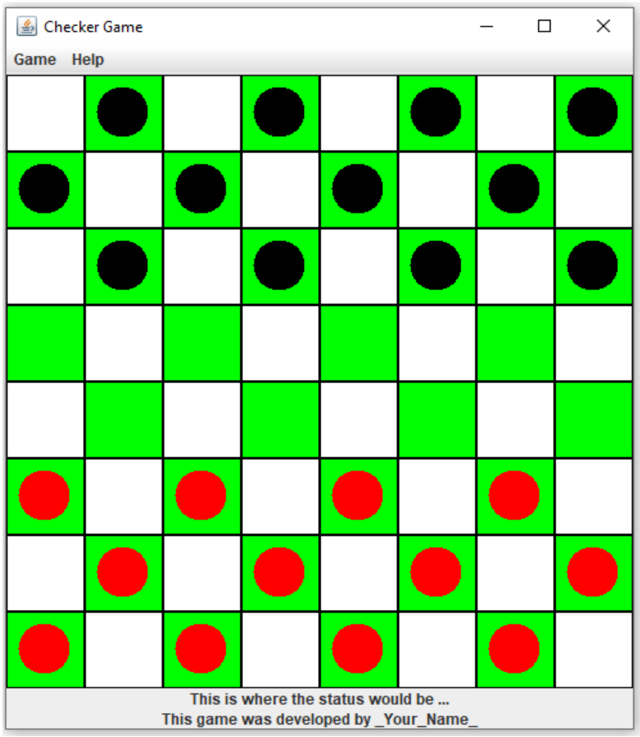
- https://www.wikihow.com/Play-Checkers
- https://youtu.be/ScKIdStgAfU

Problem Specification:

Develop a program in Java using GUI components that will draw a checkerboard and place the checker pieces on it at their starting position as shown in Figure 1. The board is made up of 64 alternating green and white squares which appear in 8 rows of 8. There are 32 green squares and 32 white squares. Each player places his pieces on the 12 green squares in the first three rows closest to the player. Each of these three rows should have a total of 4 checkers. Remember that checkers may only move in diagonal directions on the green squares. Since the board has 8 rows, 6 of the rows will be taken up by the players' checkers and the remaining two rows will be left open in the middle of the board.

The pieces are specified by an 8-by-8 char array called *boardStatus*. The status of any given square is either ***r*** (red), ***b*** (balck), or ***e*** (empty) as shown in the figure below.

```
private char[][] boardStatus
   = new char[][]{
   {'e','b','e','b','e','b','e','b'},
   {'b','e','b','e','b','e','b','e'},
   {'e','b','e','b','e','b','e','b'},
   {'e','e','e','e','e','e','e','e'},
   {'e','e','e','e','e','e','e','e'},
   {'r','e','r','e','r','e','r','e'},
   {'e','r','e','r','e','r','e','r'},
   {'r','e','r','e','r','e','r','e'}
};
```



This is where the status would be ...
This game was developed by _Your_Name_

Graphical User Interface and Exception

Figure 1: Board status array and associated board. The checkerboard on the right side represents the starting position of the checker pieces in a checkerboard game.

The above design of the GUI involves three classes, **CheckerPiece**, **CheckerBoard** and **CheckerGame**. The specifications for these three classes are discussed below. You need to write another class called **IllegalCheckerboardArgumentException**, the specification for which is given later in this document.

1. **CheckerPiece**: This class represent a square in the checkerboard that extends the JComponent class. It draws a square if it is empty (e) or a square with a black(b) and red(r) checkers if it is red or black respectively. It overrides the paintComponent(Graphics g) method to draw. You can use fillRect(), fillOval(), drawRect() methods to draw squares and checkers. Write the following methods and instance variables:
   - Instance variables:
     - char status: It stores the status of a square in the checkerboard. The valid values are ‘r’, ‘b’ and ‘e’.
     - int row, int column: The row and column indices of the square in the checkerboard. For example, row is 0 and column is 0 for the top-left square in Figure 1. Also, the *row* is 7 and the *column* is 7 for the bottom-right square.
     - You can declare as many instance variables and constants as you think necessary. I have used constants for height and width of the squares and checkers, and x and y coordinates for the squares and checkers. In Figure 1, the length of a side of the squares is 60 pixels and the length of a side of the checkers is 40 pixels.
   - Methods:
     - `public CheckerPiece(int row, int column, char status)`
       The constructor that sets the row, column and status of a checker piece. It throws an `IllegalCheckerboardArgumentException` if the value of the row, column or status is invalid. For example, *row* and *column* need to be between 0 and 7, and the *status* can be either ‘e’, ‘b’ or ‘r’. Also, throw an `IllegalCheckerboardArgumentException` if there is an attempt to put a red or black checker on an invalid square i.e. checkers cannot be placed on white squares.
       `@Override`
     - `public void paintComponent(Graphics g)`
       Override the paintComponent() method from the JComponent class. Draw checkerboard squares with or without the checker piece based on the value of the *status*.
     - You can add as many helper methods or inner classes as you think necessary if you would like to make it functional and get bonus points.

2. **CheckerBoard**: This class represents a checkerboard that contains 64 CheckerPiece objects (squares with/without checkers). It extends the JPanel class and adds 64 CheckerPiece objects to the panel i.e. the CheckerBoard class. You can use the GridLayout,

8 x 8, to organize the CheckerPiece objects on the panel. Write the following methods and instance variables:

- Instance variables:
  - char [][] boardStatus: A two-dimensional character array, 8 x 8, that stores the status value for each of the squares.
  - You can declare as many instance variables and constants as you think necessary. I used two constants, ROW and COLUMN for the boardStatus array.
- Methods:
  - `public CheckerBoard(char [][] boardStatus)`

    The constructor that sets the boardStatus array using the received parameter. It also uses this array to instantiate 64 CheckerPiece objects based on the row, column and status values. It adds all these objects to the panel. As mentioned above, you can use the GridLayout, 8 x 8, to organize the CheckerPiece objects on the panel.
  - `public void setBoardStatus(char [][] boardStatus)`

    A mutator method that sets the boardStatus array using the received parameter. You can call repaint() method after setting the array to redraw the checker pieces based on the new value.
  - `public void setCheckerPiece(int row, int column, char status)`

    A mutator method that sets the *status* value for the square specified by *row* and *column.* You can call repaint() method after setting the status to redraw the checker pieces based on the new value.
  - You can add as many helper methods or inner classes as you think necessary if you would like to make it functional and get bonus points.

3. **CheckerGame**: This driver class represents the checker game window. The checker game window has a checkerboard along with the status bar and menu bar. The status bar is a JPanel object with two JLabels that show number of red and black checkers remaining during the game and the name of the programmer, YOUR NAME, who developed this game. That being said, CheckerGame class extends JFrame to create a window and adds a CheckerBoard object (a subclass of JPanel) and a status panel. It also adds a menu bar to the window as shown in Figure 2. Write the following methods and instance variables:
   - Instance variables:
     - You can declare as many instance variables and constants as you think necessary. I used only one instance variable, a char [][] boardStatus with values, see Figure 2, to store the initial position of the checker pieces non-functional GUI. Eventually, I passed it to CheckerBoard object.
   - Methods:
     - `public CheckerGame()`

       The constructor that sets the layout to organize CheckerBoard object (a JPanel), the status bar (a JPanel object) and the menu bar. You need to pass a two-dimensional char array with boardStatus, same as Figure 1, when you create CheckerBoard object. You can use the BorderLayout to set the layout of the JFrame to organize the components. Also, set the title, size or dimension, default close operation, etc. for the JFrame window. I have used 505 x 585 pixels dimension for the game window displayed in Figure 1 and 2. Add menus and menu items to the menu bar as shown in Figure 2.

- `public static void main(String[] args)`
  Create an object of the CheckerGame class and make the window visible.

4. Important: You can add as many helper classes and methods as you deem necessary to complete this project. The classes and methods specified above are good enough for the non-functional GUI of a checkerboard game.

```
private char[][] boardStatus
    = new char[][]{
      {'e','b','e','b','e','b','e','b'},
      {'b','e','b','e','b','e','b','e'},
      {'e','b','e','b','e','b','e','b'},
      {'e','e','e','e','e','e','e','e'},
      {'e','e','e','e','e','e','e','e'},
      {'r','e','r','e','r','e','r','e'},
      {'e','r','e','r','e','r','e','r'},
      {'r','e','r','e','r','e','r','e'}
};
```

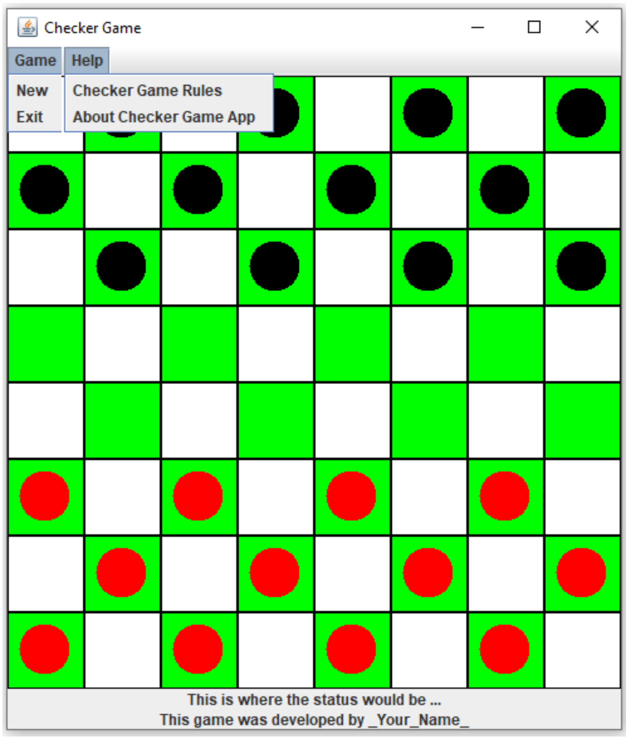

Checker Game
Game
Help
New
Exit
Checker Game Rules
About Checker Game App
This is where the status would be ...
This game was developed by _Your_Name_

Figure 2: Board status array and associated board. The checkerboard on the right side shows the menu bar with menus and menu items. You don't need to make these functional unless you would like to get bonus credits.

Define Customized Exception Class for Checkerboard Game:

5. First, watch the new lecture video on exception posted on Canvas. Here is the link: https://miamioh.instructure.com/courses/117194/discussion_topics/874858

6. **IllegalCheckerboardArgumentException**
   Write a java class called **IllegalCheckerboardArgumentException** extends **Exception** class to define a customized exception class for our checkerboard game. Please follow the example exception class designed in the Exception lecture. Define a default constructor and a constructor that receives a string message.
   - Methods:
     - `public IllegalCheckerboardArgumentException()`
     - `public IllegalCheckerboardArgumentException(String message)`

Bonus Credit:
You will receive bonus credit if you complete the followings:

1. **(30 Points)** Make the game work following the game rules described in the link above that is summarized in the following major rules.
   - It should be a two-player game where the player with black checkers will play first and then the player with Red. These turns will continue until the game is over.
   - A player can move one space diagonally to an empty square or jump over the opponent's checker piece.
   - You need to update the board with every valid move the players make. To move a checker piece, the player has to click on that piece and then click the empty square where she/he wants to move that piece.
   - Don't move if a player makes an invalid move.

2. **(10 Points)** You need to make the menus from the menu bar functional.
   - New: It starts a new game. Resets the checkerboard to initial configuration.
   - Exit: Exit the game or program.
   - Checker Game Rules: Display a JOptionPane message box that has the link to the game rules.
   - About Checker Game App: Display a JOptionPane message box that has your name, email and the university name.

3. **(10 Points)** You need to update the status label with the current count of black and red checkers as the game goes on. Also, declare the winner (result) once the game is over.

Hint for the bonus credit:
You can use MouseListener and MouseMotionListener to listen to the coordinate of the button clicked and determine which checker piece has been selected. Also, use variables to store and determine player turns and invalid moves.

Important Note:
Make sure the file names are correct and code is well commented. No late submission. You will get zero for late submission.

Submission:
Submit the java files to the appropriate submission folder on the Canvas by the due time. You can zip all the java files together and submit one zip file.

Graphical User Interface and Exception

Grading Rubric:

CheckerPiece extends JComponent	2
Instance variables are used as described	3
public CheckerPiece(**int** row, **int** column, **char** status)	5
public void paintComponent(Graphics g)	20
CheckerBoard extends JPanel	2
Instance variables are used as described	3
public CheckerBoard(**char** [][] boardStatus)	20
public void setBoardStatus(**char** [][] boardStatus)	3
public void setCheckerPiece(**int** row, **int** column, **char** status)	2
CheckerGame extends JFrame	2
Passes a char[][] to initialize the checkerboard correctly	2
Frame title, dimension and layout are correct	3
public CheckerGame() adds checkerboard object, status panel and menu bar	10
GUI looks similar to Figure 1.	10
public static void main(String[] args)	3
IllegalCheckerboardArgumentException extends Exception	5
Throws Appropriate exception object	5
Total	**100**
Bonus Credit	**50**
Functional game following the rules	30
Menus are functional	10
Status panel shows game status	10