**Final Report**

**Checker game: - implement the rules of the game and you can play with a friend. This program is supposed to print it out on JPanel, so like another window, it allows the player to choose which possible moves it can make. This panel lets two users play checkers against each other. Red always starts the game and if a player can jump an opponent’s piece, then the player must jump. When a player can make no more moves, the game ends.**

First in order to make the game, we first need to create the board of the game. I used a different class other than my checkers class, which is called private class Boards extend JPanel, so I can use the JPanel. The panel displays a 160 by 160 checkerboard with a black border. We must use Java API-Oracle Documentation, for the package and predefined method within Java API.

**Package**

import java.awt.\*;  
import java.awt.event.\*;  
import javax.swing.\*;  
import java.util.ArrayList;

For now this is the following package we are going to use to import.

**What Each Package does**

Import java.awt=Contains all of the classes for creating user interfaces and for painting graphics and images.

Import java.awt.event= Provides interfaces and classes for dealing with different types of events fired by AWT components.

Import javax.swing= Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

Import java.util.ArrayList= Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list.

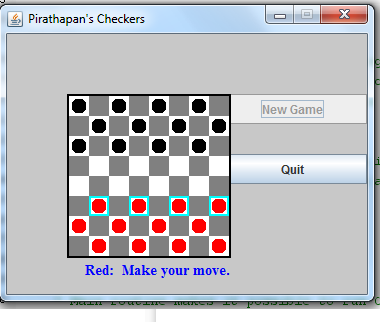
**How to make the program**

First, we have to make variables in our field class. So far, we are creating 3 variables.  
Private JButton newGameButton;  
Private JButton quitButton;  
Private JLabel message;

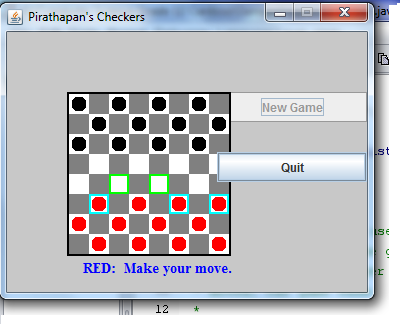
After, in our constructor Public checkers() we have to get the size of the window, so has to be size(350,250) and also set the background colour to any coloured that I prefer.

Also within the constructor you have to make a new object called board and also add the variable we made in our field class.   
 Board board = new Board();  
add(board);  
add(newGameButton);  
add(quitButton);  
add(message);

We also need to set the bound for the following variables we added, so it appears on the window and also be in the right order. We must use board.setBounds(60,60,164,164);

Within my program I have a nest of class, for example, checker pieces, board, legal move, and etc. Each class has specific function that it needs to do for in order the program to be successful.

As you can see, this is how my program finish starts off. This is when I have all the pieces set up and on the bottom, it says “Red: Make your move” because red starts off the game first.

As you can see here, when the red pieces user clicks on the checkers pieces that is baby blue, it then have option of where to move your pieces by the highlighted green colour around the small boxes. It then tells the user to make your move.

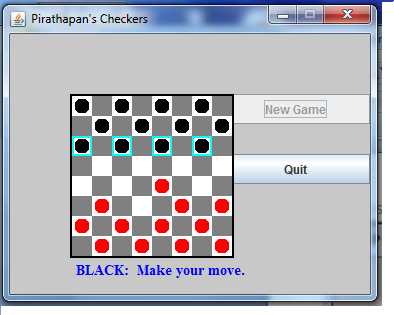
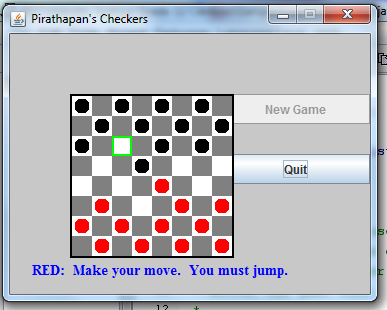
**Nested Class**

You are supposed to make multiple class in the checkers class. They can be called outside the program or inside the program. If they are called outside the program you must call the class, public Class but if you are calling it within, you must call the class, private Class. In our program we are going to call it within so it should look like this, private Class.

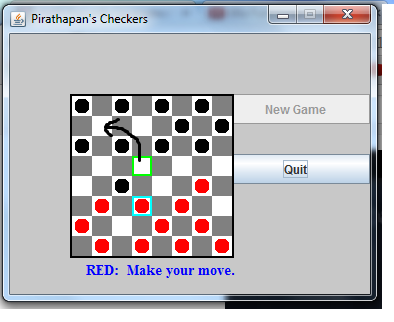
First off let’s start by making the board class. We must have a constructor in our board class, it will look like board (). This board is the main program for the buttons and label to be displayed. So first you have to set the background of the board, add the button within the constructor. We have to set the font size and type for our message so that our message will look good and it displays. There is one thing that we must include in our constructor, we must add a mouse listen, so it will look like this; addMouseListener(this);  
addActionListener(this);

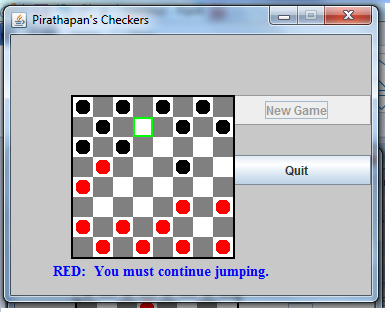
* private class Board extends JPanel implements ActionListener, MouseListener
* private static class CheckersData
* private static class CheckersMove

There are 4 different class I used include the main class which had everything inside of it.

After you move the red pieces, it will then be black pieces turn. Even if you forget who’s turn or who is next you can tell by two ways. One way is that there is a friendly message that says it is “Black: make your move” or “Red: make your move”, so this is one way to identify whose turn it is. Another way to identify is by the colour of the baby blue box. If the blue box is shown on the black pieces then it is black pieces turn. If the blue box is shown on the red pieces then it will be red pieces turn.

This situation shows how when you have the chance to take out a piece just likes the actual game of checkers it lets the piece jump over to take that piece out of the game.

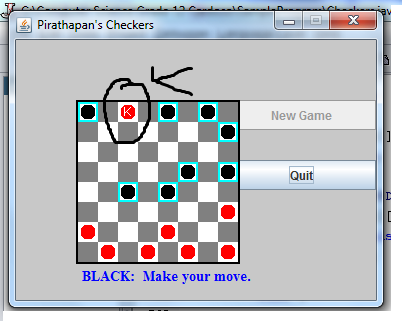


In this scenario, if you see closely the red piece can double jump the black piece and it will take two pieces instead of one pieces. In this situation this is like the actual game of checkers.

In this scenario, once you jump and you have another chance to jump again, it will then tell the user that they still have to continue jumping by the friendly message on the bottom.

**TO Print Out the Checker Board**

public void paintComponent(Graphics g)   
{   
 g.setColor(Color.black);   
 g.drawRect(0,0,getSize().width-1,getSize().height-1);  
 g.drawRect(1,1,getSize().width-3,getSize().height-3);  
 for (int row = 0; row < 8; row++)   
 {  
 for (int col = 0; col < 8; col++)   
 {  
 if ( row % 2 == col % 2 )  
 g.setColor(Color.WHITE);  
 else  
 g.setColor(Color.GRAY);  
 g.fillRect(2 + col\*20, 2 + row\*20, 20, 20);  
 switch (board.pieceAt(row,col))   
 {  
 case CheckersData.RED:  
 g.setColor(Color.RED);  
 g.fillOval(4 + col\*20, 4 + row\*20, 15, 15);  
 break;  
 case CheckersData.BLACK:  
 g.setColor(Color.BLACK);  
 g.fillOval(4 + col\*20, 4 + row\*20, 15, 15);  
 break;  
 case CheckersData.RED\_KING:  
 g.setColor(Color.RED);  
 g.fillOval(4 + col\*20, 4 + row\*20, 15, 15);  
 g.setColor(Color.WHITE);  
 g.drawString("K", 7 + col\*20, 16 + row\*20);  
 break;  
 case CheckersData.BLACK\_KING:  
 g.setColor(Color.BLACK);  
 g.fillOval(4 + col\*20, 4 + row\*20, 15, 15);  
 g.setColor(Color.WHITE);  
 g.drawString("K", 7 + col\*20, 16 + row\*20);  
 break;  
 }  
 }  
 }

****

In this scenario, you can see that when the red piece goes to the end towards the black pieces base, it then turns into a king and is able to move to any available spots it want.

**What I Used**

**Object**

This program contains many objects, these are few examples of object I have used in my program.

Board board = new Board();  
setBackground(Color.BLACK);  
addMouseListener(this);  
quitButton = new JButton("Quit");  
quitButton.addActionListener(this);  
newGameButton = new JButton("New Game");  
newGameButton.addActionListener(this);  
message = new JLabel("",JLabel.CENTER);  
message.setFont(new Font("Serif", Font.BOLD, 14));  
message.setForeground(Color.blue);  
board = new CheckersData();  
doNewGame();

**Array**

This program contains many arrays, these are few examples of array I have used in my program.

CheckersMove[] legalMoves;  
legalMoves[0].isJump()  
selectedRow = legalMoves[0].fromRow;  
selectedCol = legalMoves[0].fromCol;

**2D ARRAY**  
board = new int[8][8];  
return board[row][col];

**Reference**

<http://stackoverflow.com/questions/253492/static-nested-class-in-java-why>  
In response to why class has to have static in them, is because of the following reason when you open the webpage.

<http://www.dreamincode.net/forums/topic/291351-how-to-create-pieces-for-checkers/>  
This website helped me how to start my program and how to make the red pieces and black pieces.  
  
<https://github.com/jDramaix/droppable/blob/master/sample/src/main/java/gwtquery/plugins/droppable/client/draughtssample/CheckerBoard.java>  
This website helped me how to move the pieces in the available boxes.

<http://www.cs.berkeley.edu/~hilfingr/cs61b/f2004/public_html/hw/files/proj3-ui/game/Checkerboard.java>  
In this website, they showed me how to make legal moves and how to make the King pieces and how it can move all around the board.