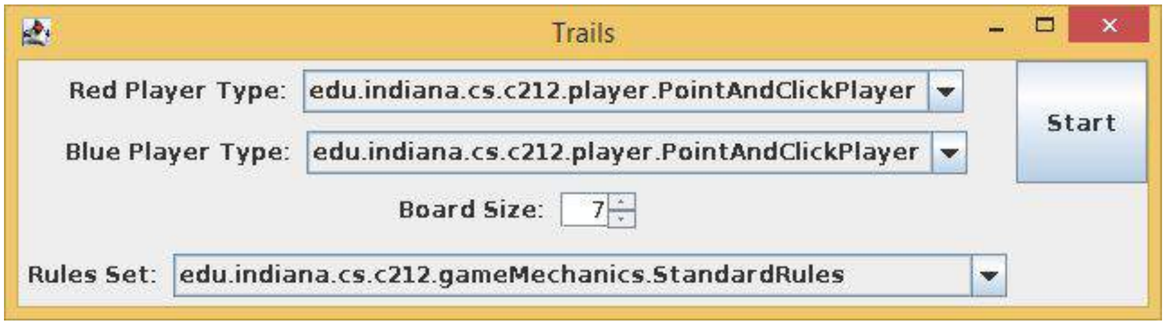
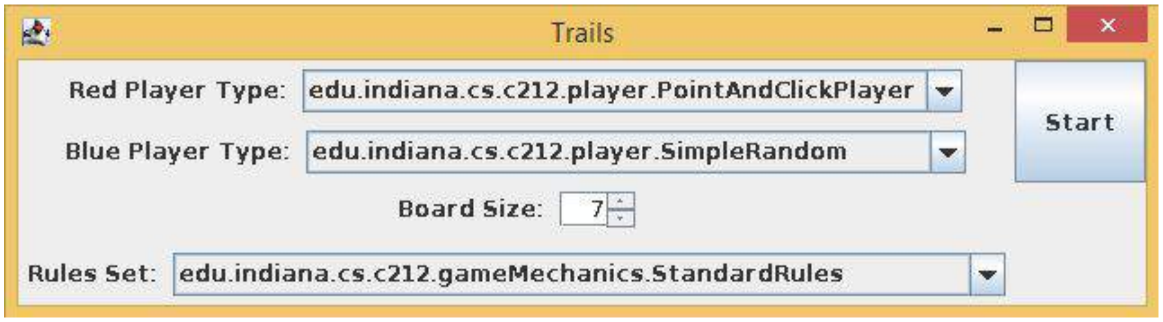
TWO PLAYER CHESS GAME

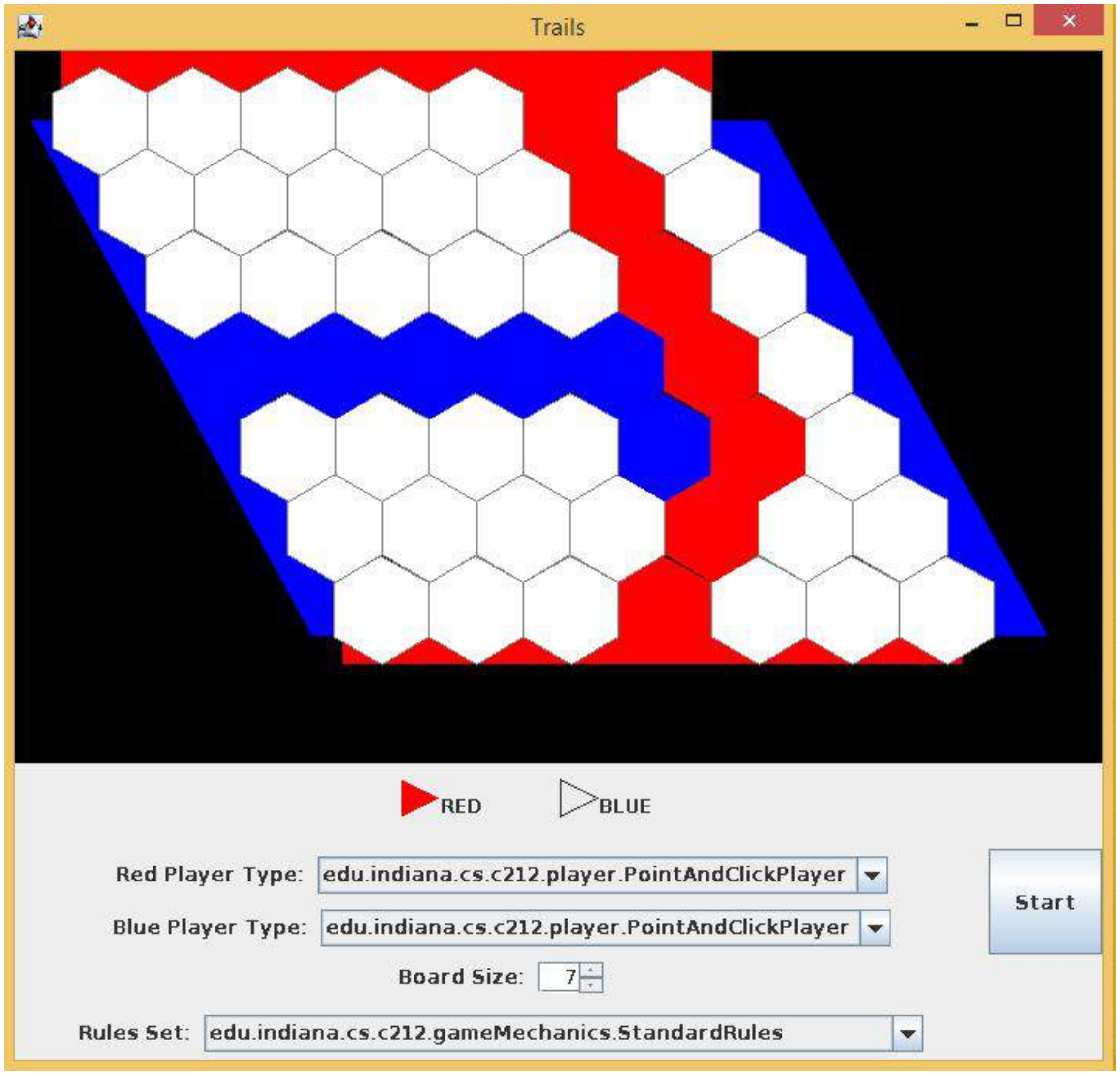
For this project, my partner John Binzer and I wrote the main panel and the game board from provided source code. The rule for the game is that when the player connects the tiles between two sides of the board, this player wins. In the main panel for the game, you can choose to play as red player click and blue player click step by step. Or you can choose to play as red player click and blue player is a randomly generated move by computer. After choosing the player type, the size of board, the type of rule you want, the player can click start button to start play the game and that brings up the game panel.



Red Player Type: edu.indiana.cs.c212.player.PointAndClickPlayer
Blue Player Type: edu.indiana.cs.c212.player.PointAndClickPlayer
Board Size: 7
Rules Set: edu.indiana.cs.c212.gameMechanics.StandardRules



Red Player Type: edu.indiana.cs.c212.player.PointAndClickPlayer
Blue Player Type: edu.indiana.cs.c212.player.SimpleRandom
Board Size: 7
Rules Set: edu.indiana.cs.c212.gameMechanics.StandardRules



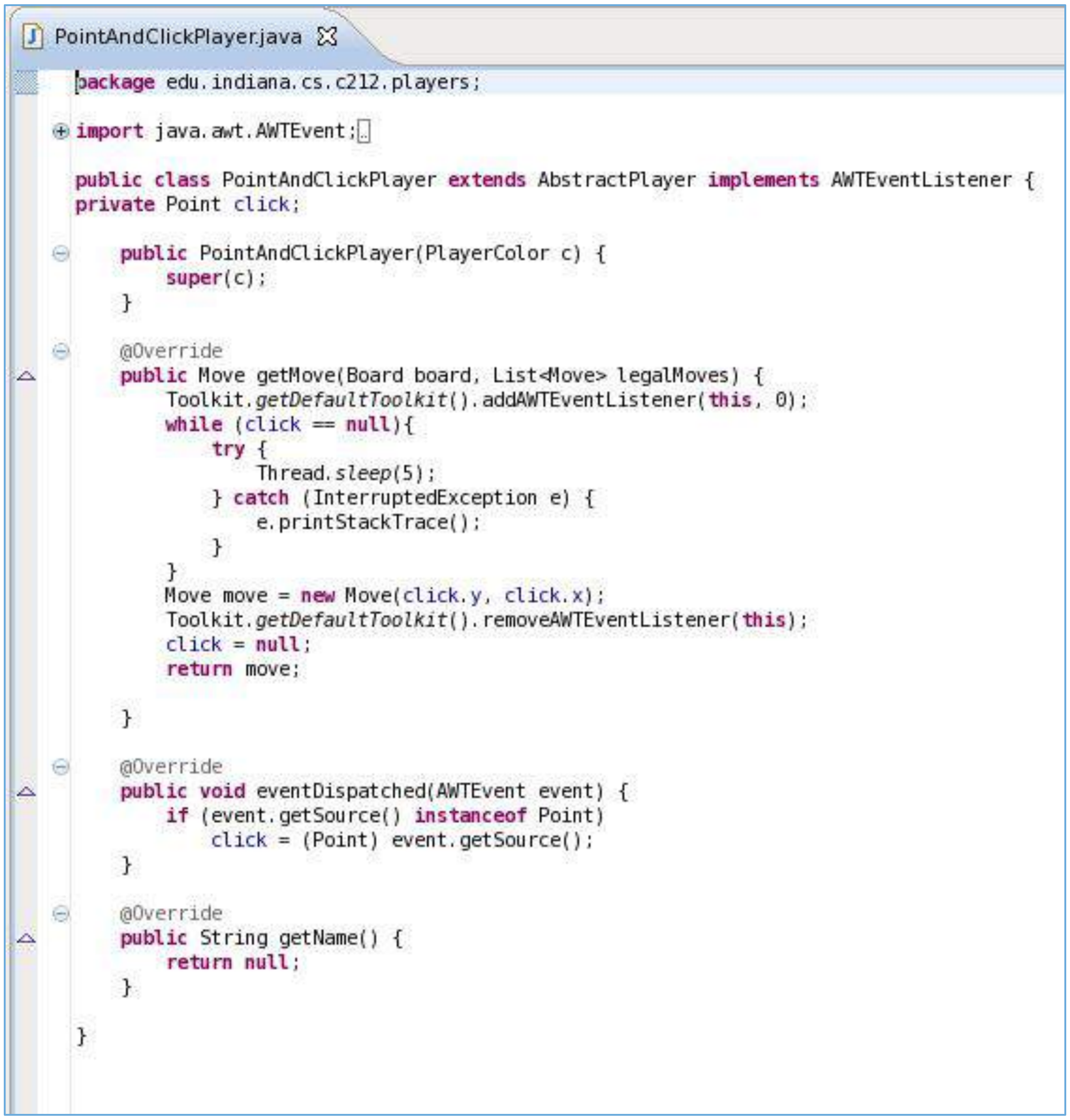
Red Player Type: edu.indiana.cs.c212.player.PointAndClickPlayer

Blue Player Type: edu.indiana.cs.c212.player.PointAndClickPlayer

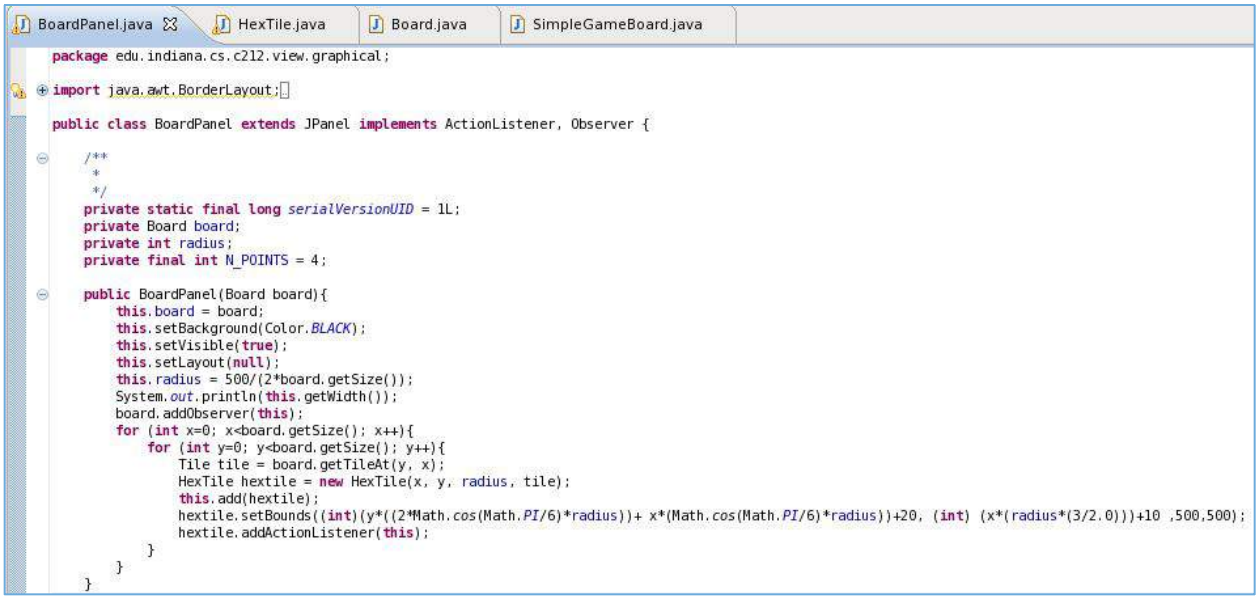
Board Size: 7

Rules Set: edu.indiana.cs.c212.gameMechanics.StandardRules

After click the start button in the main panel. The board panel shows up. You can start play the game! When any player wins the game, the game ends. The picture shows the red player wins. In the panel bellow, you can use the same player type or change the player type and start again.

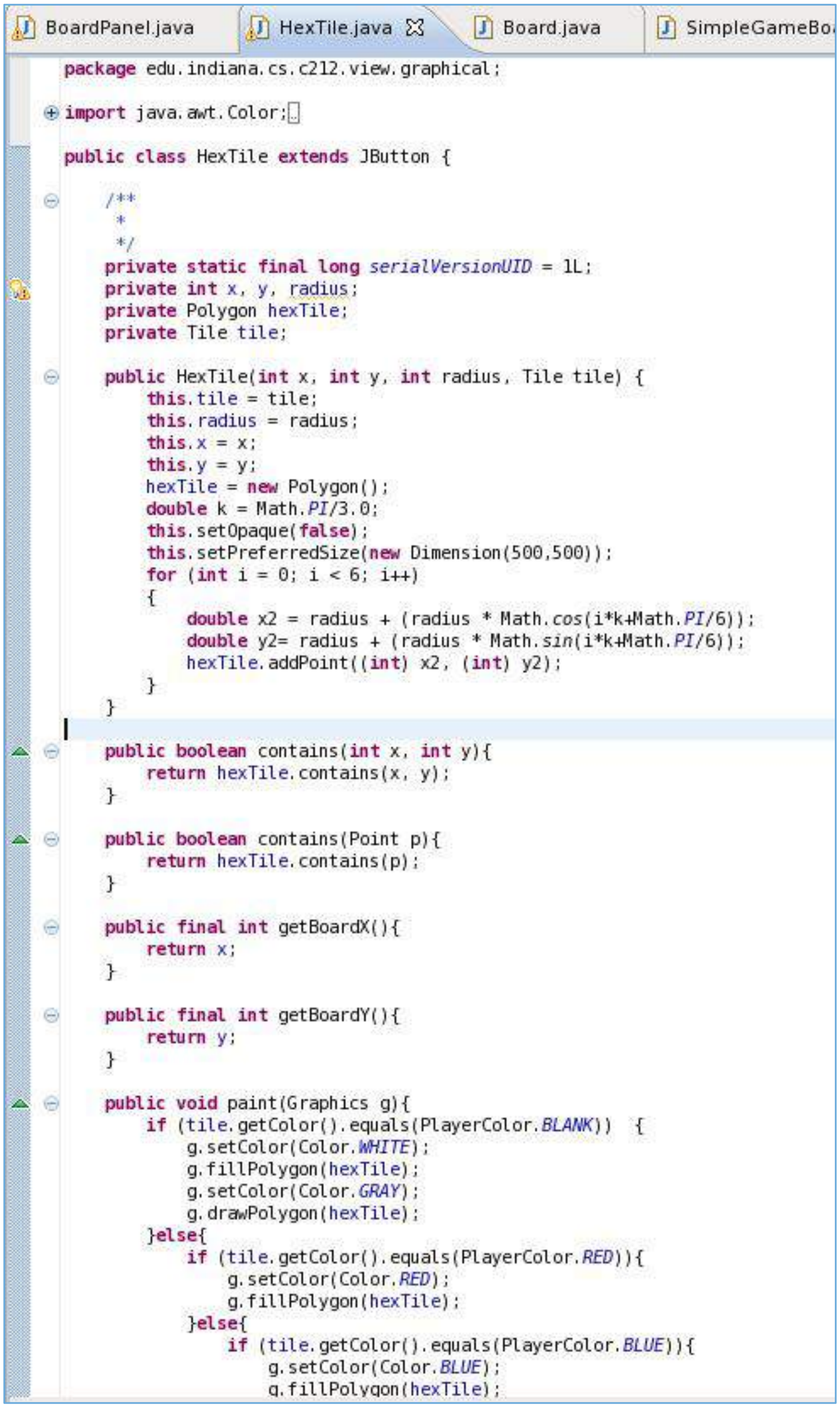

PointAndClickPlayer.java

```java
package edu.indiana.cs.c212.players;

import java.awt.AWTEvent;

public class PointAndClickPlayer extends AbstractPlayer implements AWTEventListener {
private Point click;

    public PointAndClickPlayer(PlayerColor c) {
        super(c);
    }

    @Override
    public Move getMove(Board board, List<Move> legalMoves) {
        Toolkit.getDefaultToolkit().addAWTEventListener(this, 0);
        while (click == null){
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        Move move = new Move(click.y, click.x);
        Toolkit.getDefaultToolkit().removeAWTEventListener(this);
        click = null;
        return move;

    }

    @Override
    public void eventDispatched(AWTEvent event) {
        if (event.getSource() instanceof Point)
            click = (Point) event.getSource();
    }

    @Override
    public String getName() {
        return null;
    }

}
```



```java
package edu.indiana.cs.c212.view.graphical;

import java.awt.BorderLayout;

public class BoardPanel extends JPanel implements ActionListener, Observer {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private Board board;
    private int radius;
    private final int N_POINTS = 4;

    public BoardPanel(Board board){
        this.board = board;
        this.setBackground(Color.BLACK);
        this.setVisible(true);
        this.setLayout(null);
        this.radius = 500/(2*board.getSize());
        System.out.println(this.getWidth());
        board.addObserver(this);
        for (int x=0; x<board.getSize(); x++){
            for (int y=0; y<board.getSize(); y++){
                Tile tile = board.getTileAt(y, x);
                HexTile hextile = new HexTile(x, y, radius, tile);
                this.add(hextile);
                hextile.setBounds((int)(y*((2*Math.cos(Math.PI/6)*radius))+ x*(Math.cos(Math.PI/6)*radius))+20, (int) (x*(radius*(3/2.0)))+10 ,500,500);
                hextile.addActionListener(this);
            }
        }
    }
```

This class set the board panel. This part of code draws the tiles at the needed location.



```java
package edu.indiana.cs.c212.view.graphical;

import java.awt.Color;

public class HexTile extends JButton {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private int x, y, radius;
    private Polygon hexTile;
    private Tile tile;

    public HexTile(int x, int y, int radius, Tile tile) {
        this.tile = tile;
        this.radius = radius;
        this.x = x;
        this.y = y;
        hexTile = new Polygon();
        double k = Math.PI/3.0;
        this.setOpaque(false);
        this.setPreferredSize(new Dimension(500,500));
        for (int i = 0; i < 6; i++)
        {
            double x2 = radius + (radius * Math.cos(i*k+Math.PI/6));
            double y2= radius + (radius * Math.sin(i*k+Math.PI/6));
            hexTile.addPoint((int) x2, (int) y2);
        }
    }

    public boolean contains(int x, int y){
        return hexTile.contains(x, y);
    }

    public boolean contains(Point p){
        return hexTile.contains(p);
    }

    public final int getBoardX(){
        return x;
    }

    public final int getBoardY(){
        return y;
    }

    public void paint(Graphics g){
        if (tile.getColor().equals(PlayerColor.BLANK))  {
            g.setColor(Color.WHITE);
            g.fillPolygon(hexTile);
            g.setColor(Color.GRAY);
            g.drawPolygon(hexTile);
        }else{
            if (tile.getColor().equals(PlayerColor.RED)){
                g.setColor(Color.RED);
                g.fillPolygon(hexTile);
            }else{
                if (tile.getColor().equals(PlayerColor.BLUE)){
                    g.setColor(Color.BLUE);
                    g.fillPolygon(hexTile);
```

This class set the shape for the hex tile and can paint the color for it.