Assignment #2

CSD 207
Prof. Sulabh Bansal

Question 1

Write a program that animates a hangman game swing, as shown in Figure below. Press the up-arrow key to increase the speed and the down-arrow key to decrease it. Press the S key to stop animation and the R key to resume.

Source

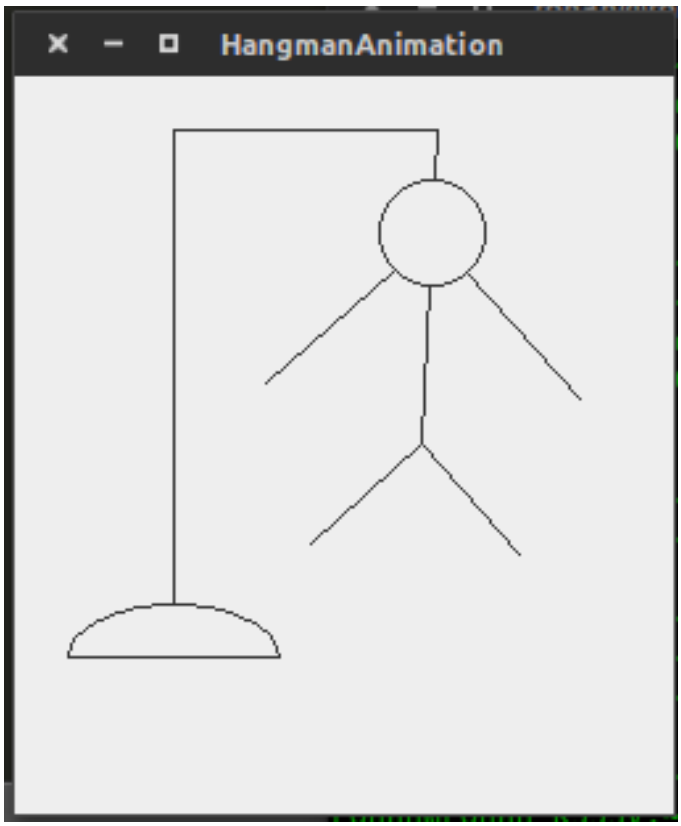
HangmanAnimation.java

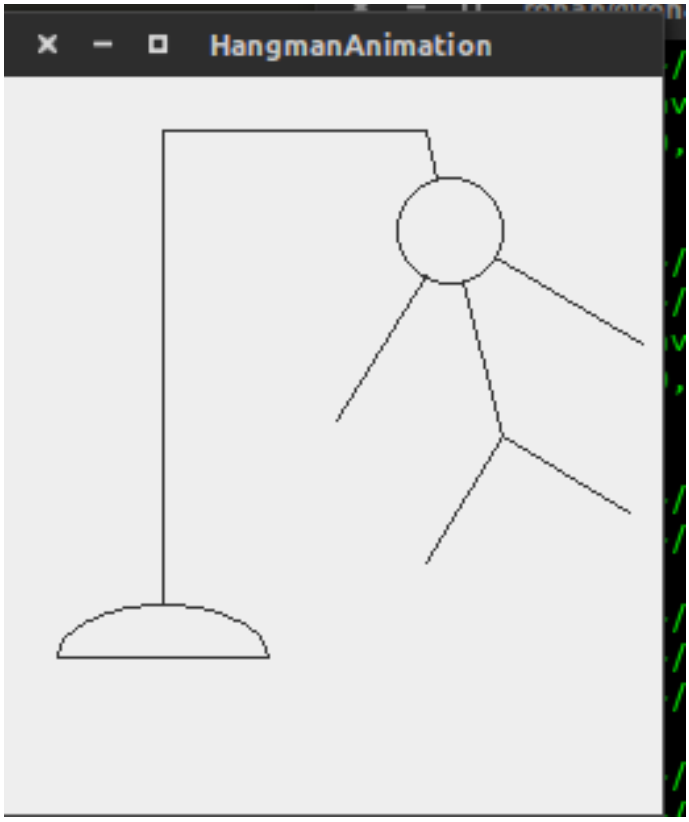
```
package Hangman;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class HangmanAnimation extends JFrame {
  private HangmanPanel canvas = new HangmanPanel();

  public HangmanAnimation() {
    this.add(canvas, BorderLayout.CENTER);
  }

  public static void main(String[] args) {
    JFrame frame = new HangmanAnimation();
    frame.setTitle("HangmanAnimation");
    frame.setLocationRelativeTo(null);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(250, 280);
    frame.setVisible(true);
  }

  class HangmanPanel extends JPanel {
    int leftAngle = 120;
    int rightAngle = 60;
    int angle = leftAngle;
    int angleDelta = 1;
    int delay = 100;

    Timer timer = new Timer(delay, new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        repaint();
      }
    });

    HangmanPanel() {
       timer.start();

      this.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
          switch (e.getKeyCode()) {
```

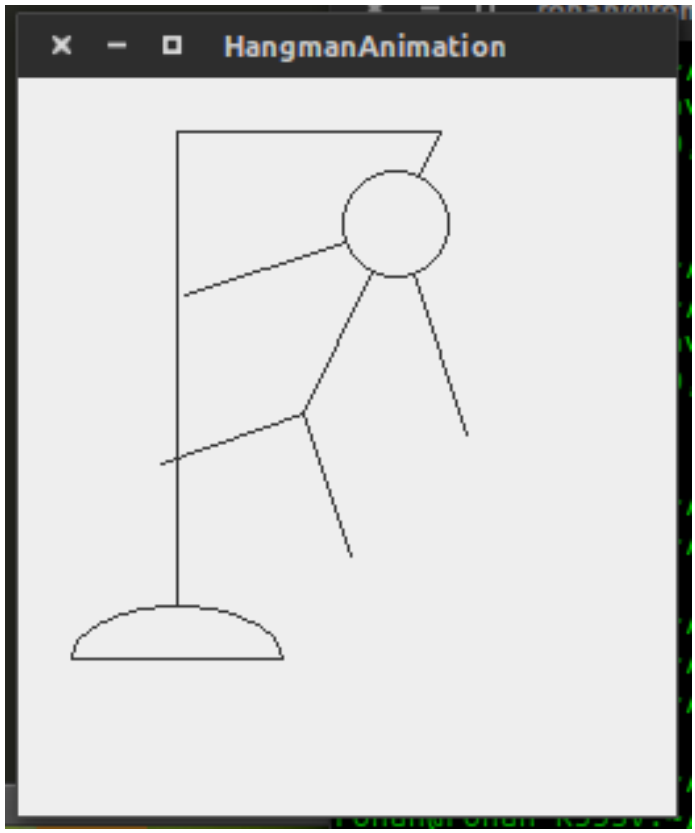
```
        case KeyEvent.VK_DOWN: delay += 5; break;
        case KeyEvent.VK_UP: delay -= 5; break;
        case KeyEvent.VK_S: timer.stop(); break;
        case KeyEvent.VK_R: timer.start(); break;
      }

      if (delay < 0) delay = 10;

      timer.setDelay(delay);
  }});


  setFocusable(true);
}

protected void paintComponent(Graphics g) {
  super.paintComponent(g);

  g.drawArc(20, 200, 80, 40, 0, 180);
  g.drawLine(20,220,100,220);
  g.drawLine(20 + 40, 200, 20 + 40, 20);
  g.drawLine(20 + 40, 20, 20 + 40 + 100, 20);

  if (angle < rightAngle)
    angleDelta = 1;
  else if (angle > leftAngle)
    angleDelta = -1;

  angle += angleDelta;

  int x1 = 20 + 40 + 100;
  int y1 = 20;
  int radius = 20;
  int x = x1 + (int)(radius * Math.cos(Math.toRadians(angle)));
  int y = y1 + (int)(radius * Math.sin(Math.toRadians(angle)));
  g.drawLine(20 + 40 + 100, 20, x, y);

  double length = 20 + 20;
  x = x1 + (int)(length * Math.cos(Math.toRadians(angle)));
  y = y1 + (int)(length * Math.sin(Math.toRadians(angle)));
  g.drawOval(x - radius, y - radius, 2 * radius, 2 * radius);


  length = distance(x1, y1,
      20 + 40 + 100 - (int)(radius * Math.cos(Math.toRadians(45))),
      40 + radius + (int)(radius * Math.sin(Math.toRadians(45))));
  double angle1 = Math.toDegrees(Math.asin(
      radius * Math.cos(Math.toRadians(45)) / length));
  int x2 = x1 + (int)(length * Math.cos(Math.toRadians(angle + angle1)));
  int y2 = y1 + (int)(length * Math.sin(Math.toRadians(angle + angle1)));

  length = (int)distance(x1, y1,
      20 + 40 + 100 - 60, 40 + radius + 60);
  angle1 = Math.toDegrees(Math.asin(60 / length));
  int x3 = x1 + (int)(length * Math.cos(Math.toRadians(angle + angle1)));
  int y3 = y1 + (int)(length * Math.sin(Math.toRadians(angle + angle1)));

  g.drawLine(x2, y2, x3, y3);


  length = distance(x1, y1,
      20 + 40 + 100 + (int)(radius * Math.cos(Math.toRadians(45))),
      40 + radius + (int)(radius * Math.sin(Math.toRadians(45))));
  angle1 = - Math.toDegrees(Math.asin(
```

```
            radius * Math.cos(Math.toRadians(45)) / length));
        x2 = x1 + (int)(length * Math.cos(Math.toRadians(angle + angle1)));
        y2 = y1 + (int)(length * Math.sin(Math.toRadians(angle + angle1)));

        length = (int)distance(x1, y1,
            20 + 40 + 100 + 60, 40 + radius + 60);
        angle1 = -Math.toDegrees(Math.asin(60 / length));
        x3 = x1 + (int)(length * Math.cos(Math.toRadians(angle + angle1)));
        y3 = y1 + (int)(length * Math.sin(Math.toRadians(angle + angle1)));

        g.drawLine(x2, y2, x3, y3);


        length = 40 + 20;
        x2 = x1 + (int)(length * Math.cos(Math.toRadians(angle)));
        y2 = y1 + (int)(length * Math.sin(Math.toRadians(angle)));

        length = 40 + 20 + 60;
        x3 = x1 + (int)(length * Math.cos(Math.toRadians(angle)));
        y3 = y1 + (int)(length * Math.sin(Math.toRadians(angle)));

        g.drawLine(x2, y2, x3, y3);


        length = distance(x1, y1,
            20 + 40 + 100 - 40, 40 + radius + 80 + 40);
        angle1 = Math.toDegrees(Math.asin(
            40.0 / length));
        int x4 = x1 + (int)(length * Math.cos(Math.toRadians(angle + angle1)));
        int y4 = y1 + (int)(length * Math.sin(Math.toRadians(angle + angle1)));
        g.drawLine(x3, y3, x4, y4);


        angle1 = -Math.toDegrees(Math.asin(
            40.0 / length));
        x4 = x1 + (int)(length * Math.cos(Math.toRadians(angle + angle1)));
        y4 = y1 + (int)(length * Math.sin(Math.toRadians(angle + angle1)));
        g.drawLine(x3, y3, x4, y4);
      }
    }


    public static double distance(
        double x1, double y1, double x2, double y2) {
      return Math.sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    }
}
```

Screenshots







Question 2

Implement a Game of TicTacToe which can be played by two players. Each player takes chance one by one. A player wins if his symbol is present in consecutive three positions either horizontly, vertically or diagonally. The game is stopped either if a player wins or all 9 places are filled.

Source

TicTacToe.java

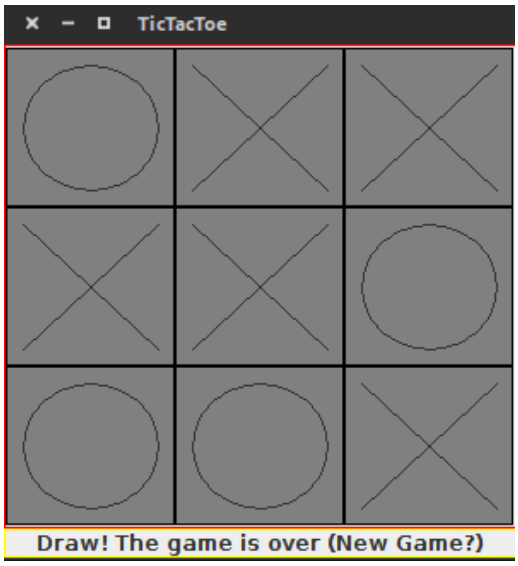
```
package ttt;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.LineBorder;

@SuppressWarnings("serial")

public class TicTacToe extends JApplet {
  private char whoseTurn = 'X';

  private Cell[][] cells =  new Cell[3][3];

  private JLabel jlblStatus = new JLabel("X's turn to play",
SwingConstants.CENTER);


  public void resetColor(){
    for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++){
        cells[i][j].setBackground(Color.gray);
      }
  }

  public TicTacToe() {
    JPanel p = new JPanel(new GridLayout(3, 3, 0, 0));
    for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++){
        cells[i][j] = new Cell();
        cells[i][j].setBackground(Color.gray);
        p.add(cells[i][j]);
      }



    jlblStatus.addMouseListener(new MouseAdapter()
    {
            public void mouseClicked(MouseEvent e)
            {
                if(whoseTurn == ' '){
                        for (int i = 0; i < 3; i++)
                       for (int j = 0; j < 3; j++){
                        cells[i][j].setToken(' ');
                resetColor();
                        jlblStatus.setText("X's turn to play");
```

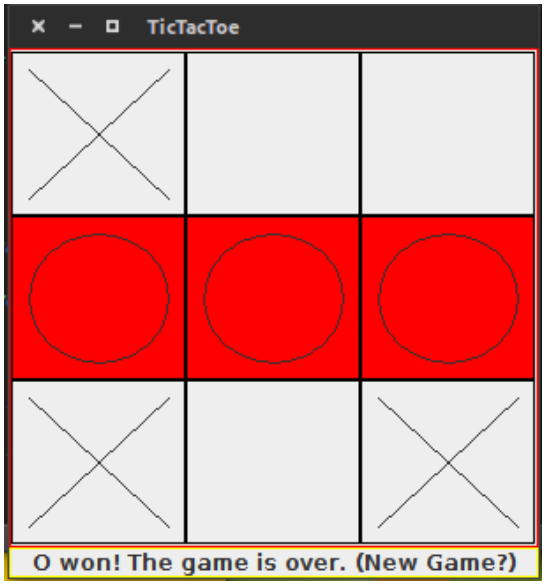
```
                    whoseTurn = 'X';
                  }
                }
              }
            });

    p.setBorder(new LineBorder(Color.red, 1));
    jlblStatus.setBorder(new LineBorder(Color.yellow, 1));

    add(p, BorderLayout.CENTER);
    add(jlblStatus, BorderLayout.SOUTH);
  }


  public boolean isFull() {
    for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
        if (cells[i][j].getToken() == ' ')
          return false;

    return true;
  }

  public boolean isWon(char token) {
    for (int i = 0; i < 3; i++)
      if ((cells[i][0].getToken() == token)
          && (cells[i][1].getToken() == token)
          && (cells[i][2].getToken() == token)) {

          cells[i][0].setBackground(Color.red);
          cells[i][1].setBackground(Color.red);
          cells[i][2].setBackground(Color.red);

        return true;
      }

    for (int j = 0; j < 3; j++)
      if ((cells[0][j].getToken() ==  token)
          && (cells[1][j].getToken() == token)
          && (cells[2][j].getToken() == token)) {

                cells[0][j].setBackground(Color.red);
                cells[1][j].setBackground(Color.red);
                cells[2][j].setBackground(Color.red);

        return true;
      }

    if ((cells[0][0].getToken() == token)
        && (cells[1][1].getToken() == token)
        && (cells[2][2].getToken() == token)) {

                cells[0][0].setBackground(Color.red);
                cells[1][1].setBackground(Color.red);
                cells[2][2].setBackground(Color.red);

      return true;
    }

    if ((cells[0][2].getToken() == token)
        && (cells[1][1].getToken() == token)
        && (cells[2][0].getToken() == token)) {

                cells[0][2].setBackground(Color.red);
```

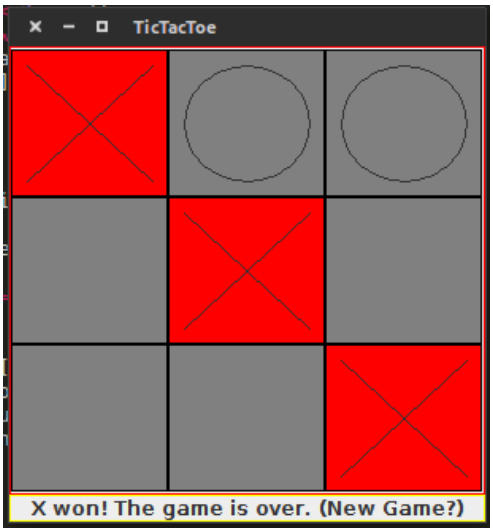
```
                cells[1][1].setBackground(Color.red);
                cells[2][0].setBackground(Color.red);

      return true;
    }

    return false;
  }

  // An inner class for a cell
  public class Cell extends JPanel {
    // Token used for this cell
    private char token = ' ';

    public Cell() {
      setBorder(new LineBorder(Color.black, 1));
      addMouseListener(new CellMouseListener());
    }


    public char getToken() {
      return token;
    }


    public void setToken(char c) {
      token = c;
      repaint();
    }


    protected void paintComponent(Graphics g) {
      super.paintComponent(g);

      if (token == 'X') {
        g.drawLine(10, 10, getWidth() - 10, getHeight() - 10);
        g.drawLine(getWidth() - 10, 10, 10, getHeight() - 10);
      }
      else if (token == 'O') {
        g.drawOval(10, 10, getWidth() - 20, getHeight() - 20);
      }
    }

    private class CellMouseListener extends MouseAdapter {

      public void mouseClicked(MouseEvent e) {

        if (token == ' ' && whoseTurn != ' ') {
          setToken(whoseTurn);

          if (isWon(whoseTurn)) {
            jlblStatus.setText(whoseTurn + " won! The game is over. (New
Game?)");
            //resetColor();
            whoseTurn = ' ';
          }
          else if (isFull()) {
            jlblStatus.setText("Draw! The game is over (New Game?)");
            //resetColor();
            whoseTurn = ' ';
          }
          else {

            whoseTurn = (whoseTurn == 'X') ? 'O': 'X';
```

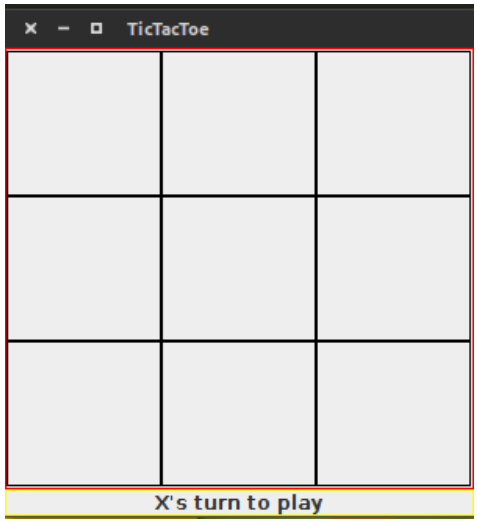
```
            jlblStatus.setText(whoseTurn + "'s turn");
          }
        }
      }
    }
  }

  public static void main(String[] args) {
    JFrame frame = new JFrame("TicTacToe");

    TicTacToe applet = new TicTacToe();

    frame.add(applet, BorderLayout.CENTER);

    frame.setSize(300, 300);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

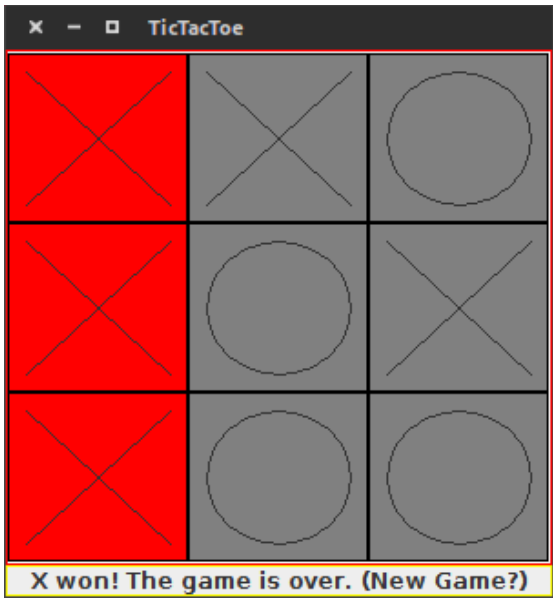
Screenshots






TicTacToe
X won! The game is over. (New Game?)


TicTacToe
X's turn to play


TicTacToe
X won! The game is over. (New Game?)