

**Masters in Applied Software Technology**

**Object Oriented Software**

**Assignment 1: Drawing Pad**

**Date : 24/10/2016**

**Student Name : Ciarán Sweeney**

**Student Number : C12317616**

**Class DrawPad**

In this class this is where the layout is made out. All the buttons and menu bars are created here and as are the listeners too. In the constructor, we set this all up and in the end made it visible for the user. The DrawPad class is a child of JFrame.

**createPad()**

In this method, the container is created. All the button listeners are done in this method, these listeners communicate with the Canvas class. All the buttons are kept in a menu bar, while the canvas is under this bar. If a button is pressed then a string or data is send to the Canvas class to let the class know which button has been pressed, where by an action would be carried out based on the string or data send. An example of this is the red button which will change the colour of shapes that would be drawn next. The colours are listed in a dropdown menu.

**Class Canvas**

This class handles majority of the work that is done. The Canvas class is where all the painting done and all the functionality of the listeners from createPad, for example save() method which is called by the save button in the DrawPad class and its own class. The canvas mainly deals with how shapes are presented to the Canvas. Shapes are painted by this class and are previewed due to a mouse event. The preview is done with a mouse listener and keeps track of the current position of the cursor. Once the mouse is clicked a certain amount of times then the location of the points of the shape is displayed on the canvas. The shape is on the canvas once the points have been plotted and this is tracked with the click integer variable.

**mouseMoved(MouseEvent e)**

This mouse listener is used to find the coordinates of cursor which allows previews of shapes.

**mousePressed(MouseEvent e)**

This mouse listener is used for the coordinates of the click of the mouse, this is helpful for placing the shapes on to the canvas.

**Paint()**

This is a method which is overridden by me. The paint method is what allows the shapes to be put onto the screen. I used an ArrayList of shapes to populate the paint with shapes.

**reset()**

Used reset method to clear the canvas. I done this by removing all shapes out the ArrayList shapes.

**undo()**

This method allows you to remove the last shape drawn. This was done by removing the last element of the ArrayList shapes.

**Save()**

Writes all the shapes from the ArrayList into binary on a text file called “save.txt”.

**Load()**

Takes in a binary file from save.txt so that the array list can be loaded up again and which is displayed on the canvas.

**setColour()**

Set the colours of the shapes on the canvas.

**Shape Class**

The Shape class is the parent class of Line, Circle, Square, Triangle and Rectangle. The shape class is also an abstract class and works as a blue print for the other classes.

**Draw()**

This draw method, draws out the shape which is then painted in the Canvas class. There is a different draw method, made out for each child of the shape class.

**Distance()**

This method is found in the square class and is used to work the distance between points. I use the distance formula to work this out. Only needing to know once lines distance to be able to draw a square.

package drawpad;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

public class DrawPad extends JFrame{

private Canvas canvas;

public DrawPad(){

//making out the layout of the window

super(" JAVA Application!");

setLocation(300,200);

setSize(800,500);

setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

setContentPane(createPad());

setVisible(true);

}

//the layout which contains all the button,menu bar and listeners

private Container createPad(){

//setting up the layout

Container pane = new JPanel(new BorderLayout());

canvas = new Canvas("");

pane.add(canvas, BorderLayout.NORTH);

//the buttons and listeners for the shapes

//line

JMenuBar bar = new JMenuBar();

JButton line = new JButton ("Line");

line.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setShape("Line");

}

});

//circle

JButton circle = new JButton ("Circle");

circle.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setShape("Circle");

}

});

//triangle

JButton triangle = new JButton ("Triangle");

triangle.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setShape("Triangle");

}

});

//square

JButton square = new JButton ("Square");

square.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setShape("Square");

}

});

//rectangle

JButton rectangle = new JButton ("Rectangle");

rectangle.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setShape("Rectangle");

}

});

//setting up the drop down menu for colours and the listeners

JMenu colour = new JMenu ("Colour");

JMenuItem red=new JMenuItem("Red");

red.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(1);

}

});

JMenuItem blue=new JMenuItem("Blue");

blue.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(2);

}

});

JMenuItem yellow=new JMenuItem("Yellow");

yellow.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(3);

}

});

JMenuItem green=new JMenuItem("Green");

green.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(4);

}

});

JMenuItem pink=new JMenuItem("Pink");

pink.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(5);

}

});

JMenuItem orange=new JMenuItem("Orange");

orange.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(6);

}

});

JMenuItem gray=new JMenuItem("Gray");

gray.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(7);

}

});

JMenuItem darkGray=new JMenuItem("Dark Gray");

darkGray.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(8);

}

});

JMenuItem lightGray=new JMenuItem("Light Gray");

lightGray.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(9);

}

});

JMenuItem cyan=new JMenuItem("Cyan");

cyan.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(10);

}

});

JMenuItem white=new JMenuItem("White");

white.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(11);

}

});

JMenuItem black=new JMenuItem("Black");

black.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.setColour(12);

}

});

colour.add(red);

colour.add(blue);

colour.add(yellow);

colour.add(green);

colour.add(pink);

colour.add(orange);

colour.add(gray);

colour.add(darkGray);

colour.add(lightGray);

colour.add(cyan);

colour.add(white);

colour.add(black);

// buttons and event listeners

JButton undo = new JButton ("Undo");

undo.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.undo();

}

});

JButton reset = new JButton ("Reset");

reset.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.reset();

}

});

JButton save = new JButton ("Save");

save.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.save();

}

});

JButton load = new JButton ("Load");

load.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {;

canvas.load();

}

});

bar.add(line);

bar.add(circle);

bar.add(triangle);

bar.add(square);

bar.add(rectangle);

bar.add(colour);

bar.add(undo);

bar.add(reset);

bar.add(save);

bar.add(load);

pane.add(bar,BorderLayout.PAGE\_START);

canvas.setBackground(Color.white);

pane.add(canvas,BorderLayout.CENTER);

return pane;

}

//main method

public static void main(String args[]){

SwingUtilities.invokeLater(new Runnable(){

public void run(){

new DrawPad();

}

});

}

}

package drawpad;

import java.util.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

//import java.io.Serializable;

import javax.swing.JPanel;

public class Canvas extends JPanel{

private String shape="";

private Line line;

//instance of the shape objects

private Triangle triangle;

private Square square;

private Circle circle;

private Rectangle rectangle;

//used saves the mouse position

private int mouseX1;

private int mouseY1;

private int mouseX2;

private int mouseY2;

private int mouseX3;

private int mouseY3;

//used to count the amount of times the mouse is used

private int click=0;

//the default colour is black

private Color colour=Color.black;

//an array of shapes

private ArrayList<Shape> shapes=new ArrayList<Shape>();

public Canvas(String shape){

this.setShape(shape);

addMouseMotionListener(new MouseAdapter(){

//used to keep track of the cursors position

public void mouseMoved(MouseEvent e) {

//used for the dynamically display with the mouse cursor

//line

if(getShape().equals("Line") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

}

//Triangle

else if(getShape().equals("Triangle") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

mouseX3=e.getX();

mouseY3=e.getY();

}

else if(getShape().equals("Triangle") && click==2){

mouseX3=e.getX();

mouseY3=e.getY();

}

//Square

else if(getShape().equals("Square") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

}

//Circle

else if(getShape().equals("Circle") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

}

//Rectangle

else if(getShape().equals("Rectangle") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

}

else if(getShape().equals("Rectangle") && click==2){

mouseX3=e.getX();

mouseY3=e.getY();

}

repaint();

}

});

addMouseListener(new MouseAdapter(){

//this keeps track of the mouse being pressed, create my objects here and add them to the arraylist

public void mousePressed(MouseEvent e){

//Line

if(getShape().equals("Line") && click==0){

mouseX1=e.getX();

mouseY1=e.getY();

click++;

}

//the shape is done so it will be save to shapes and painted

else if(getShape().equals("Line") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

line=new Line(colour,mouseX1,mouseY1,mouseX2,mouseY2);

shapes.add(line);

repaint();

click=0;

}

//the shape is done so it will be save to shapes and painted

else if(getShape().equals("Triangle") && click==0){

mouseX1=e.getX();

mouseY1=e.getY();

click++;

//repaint();

}

else if(getShape().equals("Triangle") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

click++;

//repaint();

}

//the shape is done so it will be save to shapes and painted

else if(getShape().equals("Triangle") && click==2){

mouseX3=e.getX();

mouseY3=e.getY();

triangle=new Triangle(colour,mouseX1,mouseY1,mouseX2,mouseY2,mouseX3,mouseY3);

shapes.add(triangle);

click=0;

repaint();

}

//Square

else if(getShape().equals("Square") && click==0){

mouseX1=e.getX();

mouseY1=e.getY();

click++;

}

//the shape is done so it will be save to shapes and painted

else if(getShape().equals("Square") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

square=new Square(colour,mouseX1,mouseY1,mouseX2,mouseY2);

shapes.add(square);

repaint();

click=0;

}

//Circle

else if(getShape().equals("Circle") && click==0){

mouseX1=e.getX();

mouseY1=e.getY();

click++;

}

//the shape is done so it will be save to shapes and painted

else if(getShape().equals("Circle") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

circle=new Circle(colour,mouseX1,mouseY1,mouseX2,mouseY2);

shapes.add(circle);

repaint();

click=0;

}

//Rectangle

else if(getShape().equals("Rectangle") && click==0){

mouseX1=e.getX();

mouseY1=e.getY();

click++;

}

else if(getShape().equals("Rectangle") && click==1){

mouseX2=e.getX();

mouseY2=e.getY();

click++;

}

//the shape is done so it will be save to shapes and painted

else if(getShape().equals("Rectangle") && click==2){

mouseX3=e.getX();

mouseY3=e.getY();

rectangle=new Rectangle(colour,mouseX1,mouseY1,mouseX2,mouseY2,mouseX3,mouseY3);

shapes.add(rectangle);

repaint();

click=0;

}

}

});

}

//paints my shapes to the screen

public void paint(Graphics g){

super.paint(g);

//Line

if(getShape().equals("Line") && click==1){

line=new Line(colour,mouseX1,mouseY1,mouseX2,mouseY2);

line.draw(g);

}

//Triangle

else if(getShape().equals("Triangle")&& click>0){

triangle=new Triangle(colour,mouseX1,mouseY1,mouseX2,mouseY2,mouseX3,mouseY3);

triangle.draw(g);

}

//Square

else if(getShape().equals("Square") && click==1){

square=new Square(colour,mouseX1,mouseY1,mouseX2,mouseY2);

square.draw(g);

}

//Circle

else if(getShape().equals("Circle") && click==1){

circle=new Circle(colour,mouseX1,mouseY1,mouseX2,mouseY2);

circle.draw(g);

}

//Rectangle

else if(getShape().equals("Rectangle") && click>0){

rectangle=new Rectangle(colour,mouseX1,mouseY1,mouseX2,mouseY2,mouseX3,mouseY3);

rectangle.draw(g);

}

//go through the array to put the objects on the screeen

for(Shape s:shapes)

s.draw(g);

}

public String getShape() {

return shape;

}

//used to change what shape to use

public void setShape(String shape) {

click=0;

this.shape = shape;

}

//clears all the shapes on screen by clearing the arraylist

public void reset(){

click=0;

shapes.clear();

repaint();

}

//Removes the last element of the array which clears the last drawing

public void undo(){

if(shapes.size()>0)

shapes.remove(shapes.size()-1);

repaint();

}

//saves all the objects on a text file

public void save(){

try{

FileOutputStream fout = new FileOutputStream("sava.txt");

ObjectOutputStream oos = new ObjectOutputStream(fout);

oos.writeObject(shapes);

oos.close();

}

catch (IOException e) {

e.printStackTrace();

}

repaint();

}

//Loads all the objects from a text file save.txt

public void load(){

try{

FileInputStream fin = new FileInputStream("sava.txt");

ObjectInputStream ois = new ObjectInputStream(fin);

shapes = (ArrayList<Shape>) ois.readObject();

ois.close();

}catch(Exception e){

e.printStackTrace();

}

repaint();

}

//used to change the colour of the shapes

public void setColour(int x){

if(x==1)

colour=Color.red;

else if(x==2)

colour=Color.blue;

else if(x==3)

colour=Color.yellow;

else if(x==4)

colour=Color.green;

else if(x==5)

colour=Color.pink;

else if(x==6)

colour=Color.orange;

else if(x==7)

colour=Color.gray;

else if(x==8)

colour=Color.darkGray;

else if(x==9)

colour=Color.lightGray;

else if(x==10)

colour=Color.cyan;

else if(x==11)

colour=Color.white;

else if(x==12)

colour=Color.black;

repaint();

}

}

**package** drawpad;

**import** java.awt.\*;

**public** **class** Line **extends** Shape {

**protected** **int** x1;

**protected** **int** y1;

**protected** **int** x2;

**protected** **int** y2;

**public** Line(Color c,**int** x1, **int** y1,**int** x2, **int** y2){

**super**(c);

**this**.x1=x1;

**this**.y1=y1;

**this**.x2=x2;

**this**.y2=y2;

}

**public** **void** draw(Graphics g){

g.setColor(colour);

g.drawLine(x1, y1, x2, y2);

}

}

package drawpad;

import java.awt.Color;

import java.awt.Graphics;

public class Triangle extends Line {

private int x3;

private int y3;

public Triangle(Color c,int x1, int y1,int x2, int y2,int x3, int y3){

super(c,x1,y1,x2,y2);

this.x3=x3;

this.y3=y3;

}

//drawing out a triangle

public void draw(Graphics g){

g.setColor(colour);

g.drawLine(x1, y1, x2, y2);

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

g.drawLine(x1, y1, x3, y3);

}

}

**package** drawpad;

**import** java.awt.Color;

**import** java.awt.Graphics;

**public** **class** Circle **extends** Shape {

**private** **int** x1;

**private** **int** y1;

**private** **int** x2;

**private** **int** y2;

**public** Circle(Color c,**int** x1, **int** y1,**int** x2, **int** y2){

**super**(c);

**this**.x1=x1;

**this**.y1=y1;

**this**.x2=x2;

**this**.y2=y2;

}

//draws out circle

**public** **void** draw(Graphics g){

g.setColor(colour);

g.drawOval(x1-(distance()/2),y1-(distance()/2),distance(),distance());

}

//distance formula

**public** **int** distance(){

**double** x= Math.*pow*(x1-x2,2);

**double** y= Math.*pow*(y1-y2,2);

**double** distance=Math.*sqrt*(x+y);

distance=Math.*round*(distance);

**return** (**int**)distance;

}

}

package drawpad;

import java.awt.Color;

import java.awt.Graphics;

public class Square extends Line {

public Square(Color c,int x1, int y1,int x2, int y2){

super(c,x1,x1,x2,y2);

}

//Draws a square

public void draw(Graphics g){

g.setColor(colour);

g.drawLine(x1, y1, x2, y2);

g.drawLine(x1, y1-distance(), x2, y2-distance());

g.drawLine(x1, y1-distance(), x1, y1);

g.drawLine(x2, y2, x2, y2-distance());

}

//The distance formula

public int distance(){

double x= Math.pow(x1-x2,2);

double y= Math.pow(y1-y2,2);

double distance=Math.sqrt(x+y);

distance=Math.round(distance);

return (int)distance;

}

}

**package** drawpad;

**import** java.awt.Color;

**import** java.awt.Graphics;

**public** **class** Rectangle **extends** Line {

**private** **int** x3;

**private** **int** y3;

**public** Rectangle(Color c,**int** x1, **int** y1,**int** x2, **int** y2,**int** x3,**int** y3){

**super**(c,x1,y1,x2,x2);

**this**.x3=x3;

**this**.y3=y3;

}

//drawing out a rectangle

**public** **void** draw(Graphics g){

g.setColor(colour);

g.drawLine(x1, y1, x1, y2);

g.drawLine(x1, y1, x3, y1);

g.drawLine(x1, y2, x3, y2);

g.drawLine(x3, y1, x3, y2);

}

}