**РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ“**

Катедра: Компютърни системи и технологии

**КУРСОВА ЗАДАЧА**

**по дисциплината**

**„Програмни езици“**

Изготвил: Даниел Ивелинов Панайотов

фак. номер: 233001

група: 25А

курс: 2

специалност: КСТ

Дата:................... Приел:.....................................

/доц. д-р Г. Георгиев/

1. **Задание**

**Да се разработи елементарен векторен графичен редактор на програмния език**

**Java**

Графичният редактор да позволява избор на цвят за изчертаване и

евентуално на цвят за запълване и да поддържа следните графични

примитиви:

* **Запълнен овал**
* **Незапълнен овал**
* **Незапълнен четириъгълник**

Изборът на примитив, който ще се въвежда, да става чрез меню.

Потребителският интерфейс и самите графични примитиви да са

разположени в един и същ прозорец. По време на работа с редактора обектите от графичните примитиви да се съхраняват LinkedList. В потребителския интерфейс да се предвиди команда за извеждане на съдържанието му в конзола. В програмата може да се използват различни класове по усмотрение на студента, но задължително за всеки графичен примитив, зададен в A, трябва да има отделен клас.

Всички тези класове трябва да реализират следния интерфейс:

public interface DrawableShape{

public abstract void drawShape(Graphics g);

// методът изчертава графичен примитив

}

За базов тип на списъка трябва да се използва интерфейса DrawableShape, а като елементи да се съхраняват обекти от класовете за отделните фигури.

Как се задават фигурите (например четириъгълник може да бъде зададен по две точки или по точка, ширина и височина) и как се въвеждат

стойности (например чрез работа с мишката, както в комерсиалните

редактори, или чрез въвеждане на числа в някакъв потребителски

интерфейс) е по избор на студента.

Изборът на пакет за графика/GUI между java.awt и javax.swing е също

според предпочитанията на студента.

1. **Разпечатка на кода**

**Main.java**

import classes.\*;

class Main {

public static void main(String[] args) {

new Window();

}

}

**BaseShape.java**

package classes;

import java.awt.Color;

public class BaseShape {

int x\_left;

int y\_top;

int x\_right;

int y\_bottom;

Color color;

public BaseShape (int x1, int y1, int x2, int y2, Color color) {

this.x\_left = Math.min(x1 , x2);

this.y\_top = Math.min(y1 , y2);

this.x\_right = Math.max(x1 , x2);

this.y\_bottom = Math.max(y1 , y2);

this.color = color;

}

public void move(int x1, int y1, int x2, int y2) {

this.x\_left = Math.min(x1 , x2);

this.y\_top = Math.min(y1 , y2);

this.x\_right = Math.max(x1 , x2);

this.y\_bottom = Math.max(y1 , y2);

}

public String toString () {

return this.getClass().getName()

+ " #" + Integer.toHexString(this.color.getRGB())

+ " (" + x\_left + "," + y\_top + ")"

+ " (" + x\_right + "," + y\_bottom + ")";

}

}

**DrawableShape.java**

package classes;

import java.awt.Graphics ;

public interface DrawableShape {

void drawShape (Graphics g);

void move (int x1 , int y1 , int x2 , int y2);

}

**FilledOval.java**

package classes;

import java.awt.Color;

import java.awt.Graphics;

public class FilledOval extends BaseShape implements DrawableShape {

public FilledOval(int x1, int y1, int x2, int y2, Color fill) {

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

}

@Override

public void drawShape(Graphics g) {

g.setColor(color);

g.fillOval(x\_left, y\_top, x\_right - x\_left, y\_bottom - y\_top);

}

}

**Oval.java**

package classes;

import java.awt.Color;

import java.awt.Graphics;

public class Oval extends BaseShape implements DrawableShape {

public Oval(int x1 , int y1 , int x2 , int y2 , Color fill\_color ) {

super(x1 , y1 , x2 , y2 , fill\_color );

}

@Override

public void drawShape(Graphics g) {

g.setColor(color);

g.drawOval(x\_left, y\_top, x\_right - x\_left, y\_bottom - y\_top);

}

}

**Rectangle.java**

package classes;

import java.awt.Color;

import java.awt.Graphics;

public class Rectangle extends BaseShape implements DrawableShape {

public Rectangle(int x1, int y1, int x2, int y2, Color fill) {

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

}

@Override

public void drawShape(Graphics g) {

g.setColor(color);

g.drawRect(x\_left, y\_top, x\_right - x\_left, y\_bottom - y\_top);

}

}

**Window.java**

package classes;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.LinkedList;

public class Window extends JFrame {

enum Shape { filledOval, rectangle, oval };

Shape shape = Shape.filledOval;

Color color = Color.BLACK;

boolean mousePressed = false;

LinkedList<DrawableShape> shapeStorage = new LinkedList<DrawableShape>();

int startingX;

int startingY;

public Window() {

setTitle("Bad");

setSize(600, 400);

setResizable(false);

setJMenuBar(createMenuBar());

add(createCanvas());

setVisible(true);

}

public JPanel createCanvas() {

JPanel canvas = new JPanel() {

@Override

public void paint(Graphics g) {

g.setColor(Color.WHITE);

g.fillRect(0, 0, getWidth(), getHeight());

shapeStorage.forEach((DrawableShape s) -> {s.drawShape(g);});

}

};

registerEvents(canvas);

return canvas;

}

public void registerEvents(JPanel canvas) {

canvas.addMouseListener(new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

startingX = e.getX();

startingY = e.getY();

DrawableShape tmp = switch (shape) {

case filledOval -> new FilledOval(

startingX, startingY, startingX, startingY, color);

case oval -> new Oval(

startingX, startingY, startingX, startingY, color);

case rectangle -> new Rectangle(

startingX, startingY, startingX, startingY, color);

};

shapeStorage.addLast(tmp);

mousePressed = true;

}

@Override

public void mouseReleased(MouseEvent e) {

mousePressed = false;

}

});

canvas.addMouseMotionListener(new MouseMotionAdapter() {

@Override

public void mouseDragged(MouseEvent e) {

if (!mousePressed) return;

shapeStorage.getLast()

.move(startingX, startingY, e.getX(), e.getY());

repaint();

}

});

}

public JMenuBar createMenuBar() {

JMenuBar mb = new JMenuBar();

JMenu shapesMenu = new JMenu("Shapes");

JMenu colorsMenu = new JMenu("Colors");

JMenu functions = new JMenu("Functions");

JMenuItem filledOval = new JMenuItem("Oval");

JMenuItem oval = new JMenuItem("Empty oval");

JMenuItem rectangle = new JMenuItem("Empty rectangle");

//set global shape variable

filledOval.addActionListener(e -> shape = Shape.filledOval);

oval.addActionListener(e -> shape = Shape.oval);

rectangle.addActionListener(e -> shape = Shape.rectangle);

JMenuItem clBlack = new JMenuItem("Black");

JMenuItem clRed = new JMenuItem("Red");

//set global color variable

clBlack.addActionListener(e -> color = Color.BLACK);

clRed.addActionListener(e -> color = Color.RED);

JMenuItem print = new JMenuItem("Print");

print.addActionListener(e -> {

shapeStorage.forEach((DrawableShape s) -> System.out.println(shapeStorage.indexOf(s) + ". " + s.toString()));

});

shapesMenu.add(filledOval);

shapesMenu.add(oval);

shapesMenu.add(rectangle);

colorsMenu.add(clBlack);

colorsMenu.add(clRed);

functions.add(print);

mb.add(shapesMenu);

mb.add(colorsMenu);

mb.add(functions);

return mb;

}

}