

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** **2**

з дисципліни “Побудова та анімація зображень за допомогою Java2D”

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-82  Новохацький Владислав  (*прізвище, ім’я, по батькові*)  варіант № 15 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2021

**Варіант завдання**

**Завдання**:

За допомогою Java 2D намалювати картинку з лабораторної роботи

Додатково виконати:

1. Хоча б 1 стандартний примітив, та хоча б 1 фігуру, побудовану по

точкам.

2. Хоча б 1 фігуру залити градієнтною фарбою за вибором

3. На достатній відстані від побудованого малюнку намалювати

прямокутну рамку, всередині якої відбуватиметься анімація. Тип лінії

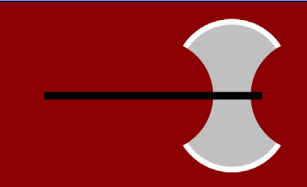
рамки задано за варіантом.

4. Виконати анімацію малюнку, за варіантом. При цьому рамка повинна

залишатися статичною. Взаємодія з рамкою не обов’язкова, якщо не

передбачено варіантом.

**Варіант:**









**Лістинг коду програми**

package lab2;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.geom.Ellipse2D;

import java.awt.geom.GeneralPath;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.Timer;

@SuppressWarnings("serial")

public class lab2 extends JPanel implements ActionListener {

Timer timer;

private static int maxWidth = 1600;

private static int maxHeight = 900;

private double angle = 360;

private double tx = -168;

private double ty = 54;

private int dx = 1;

private int dy = 1;

private boolean v = false;

private double speed = 5;

private double a = 300;

private double pos = a/2;

Color myRed = new Color(139, 0, 0);

public lab2() {

timer = new Timer(10, this);

timer.start();

}

public void paint(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

RenderingHints rh = new RenderingHints(RenderingHints.KEY\_ANTIALIASING,

RenderingHints.VALUE\_ANTIALIAS\_ON);

rh.put(RenderingHints.KEY\_RENDERING, RenderingHints.VALUE\_RENDER\_QUALITY);

g2d.setRenderingHints(rh);

g2d.setBackground(myRed);

g2d.clearRect(0, 0, maxWidth, maxHeight);

g2d.setColor(Color.WHITE);

BasicStroke bs1 = new BasicStroke(16, BasicStroke.CAP\_ROUND, BasicStroke.JOIN\_ROUND);

g2d.setStroke(bs1);

g2d.drawRect(20, 20, 1540, 820);

double points[][] = {

{ 0, 100 },

{ 280, 100 },

{ 280, 110 },

{ 0, 110 }

};

GeneralPath rect = new GeneralPath();

rect.moveTo(points[0][0], points[0][1]);

for (int k = 1; k < points.length; k++)

rect.lineTo(points[k][0], points[k][1]);

rect.closePath();

g2d.translate(maxWidth / 2, maxHeight / 2);

g2d.translate(tx, ty);

g2d.rotate(Math.toRadians(angle), rect.getBounds2D().getCenterX(), rect.getBounds2D().getCenterY());

g2d.setColor(Color.WHITE);

g2d.fillOval(220-55-8, 105-100, 176, 200);

GradientPaint gp = new GradientPaint(200, 105, Color.WHITE, 400, 105, Color.BLACK, true);

g2d.setPaint(gp);

g2d.fillOval(220-55, 105-92, 160, 184);

g2d.setColor(myRed);

g2d.fillArc(268,105-70,140,140,90,180);

g2d.fillArc(220-140,105-70,140,140,90,-180);

g2d.setColor(Color.BLACK);

g2d.fill(rect);

}

public void actionPerformed(ActionEvent e) {

angle -= 3.6;

if (angle == 0)

angle = 360;

pos+=speed;

if (pos==a) {

if (v) {

dx=-dx;

} else {

dy=-dy;

}

v=!v;

pos=0;

}

if (v) {

ty+=dy\*speed;

} else {

tx+=dx\*speed;

}

repaint();

}

public static void main(String[] args) {

JFrame frame = new JFrame("lab2");

frame.add(new lab2());

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(1600, 900);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setVisible(true);

Dimension size = frame.getSize();

Insets insets = frame.getInsets();

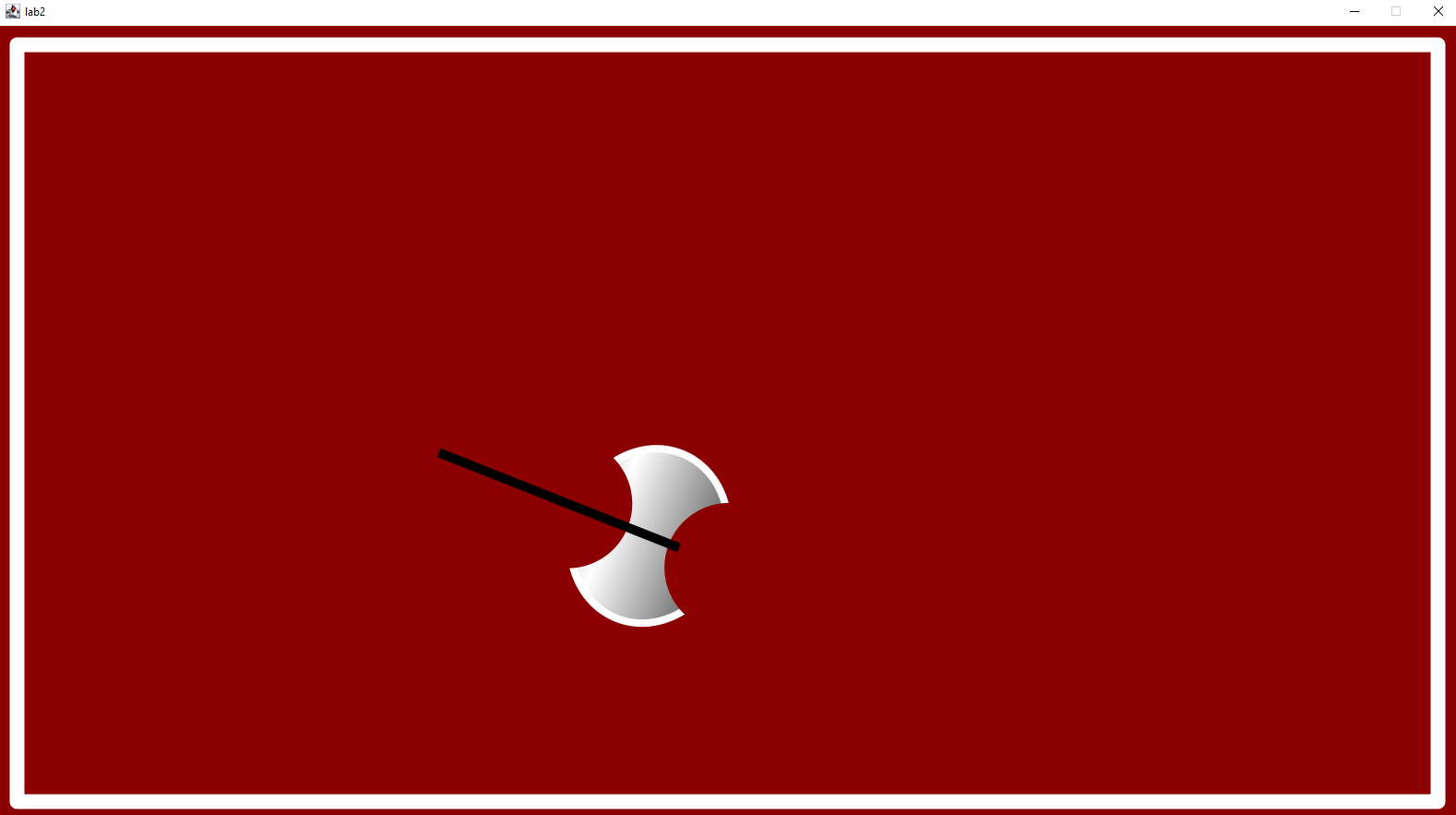
maxWidth = size.width - insets.left - insets.right - 1;

maxHeight = size.height - insets.top - insets.bottom - 1;

}

}

**Результат**

****