

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 “Математичні та алгоритмічні основи комп’ютерної графіки”

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав  студент III курсу  групи КП-83  Мар’яненко Роман Геннадійович  (*прізвище, ім’я, по батькові*)  варіант № 11 |  |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладачем  Шкурат Оксаною Сергіївною (*прізвище, ім’я, по батькові*) |

Київ 2020

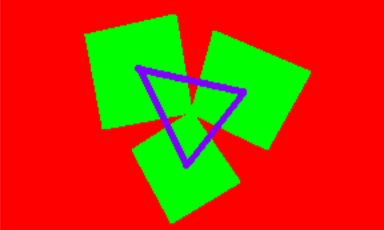
**Варіант завдання**

**Завдання**: За допомогою Java 2D намалювати картинку з лабораторної роботи №1 (за варіантом).

Додатково виконати:

1. Хоча б 1 стандартний примітив, та хоча б 1 фігуру, побудовану по точкам (ламаною).
2. Хоча б 1 фігуру залити градієнтною фарбою за вибором (в цьому випадку колір може не співпадати з варіантом із лабораторної роботи № 1).
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:**



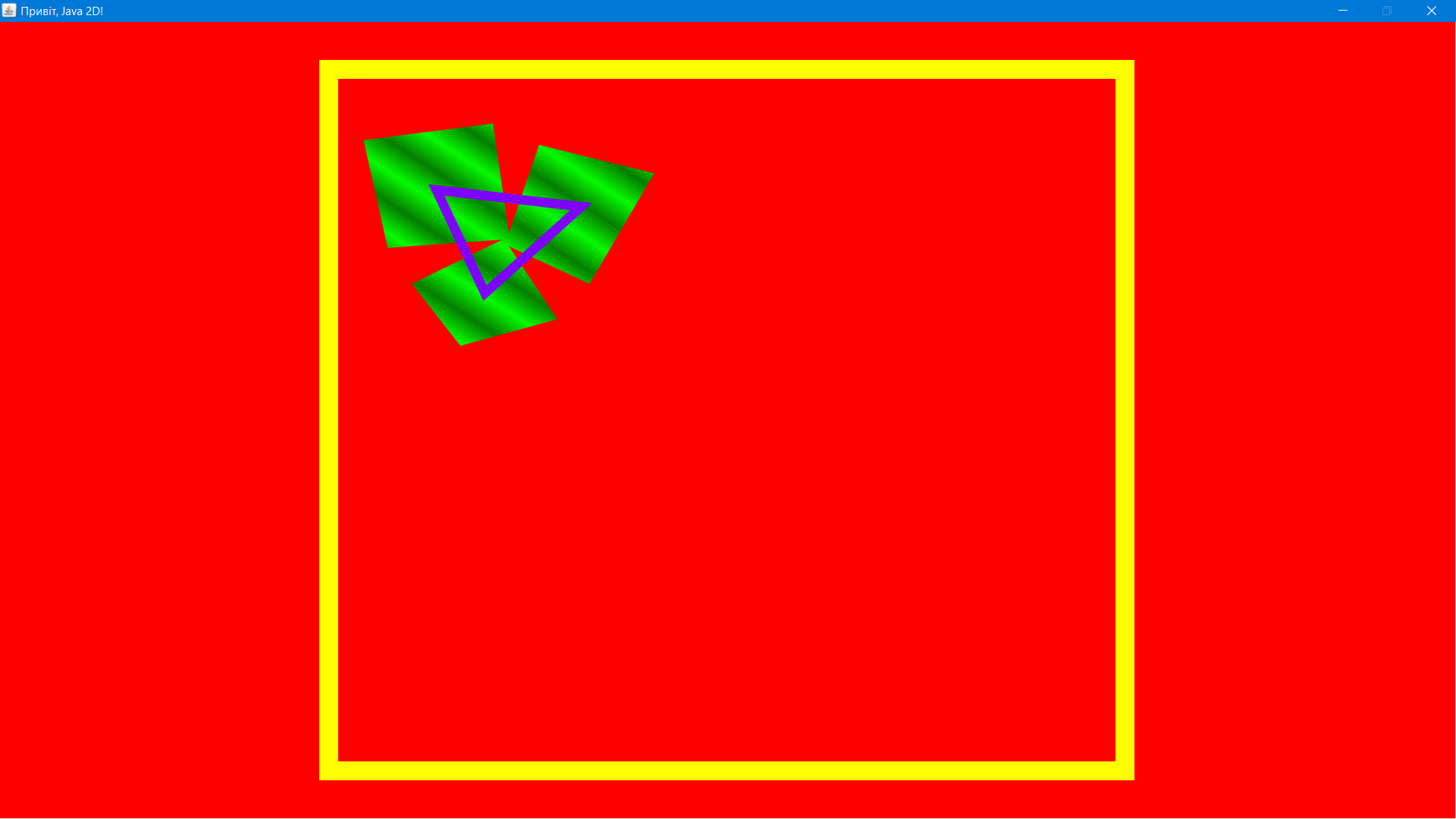


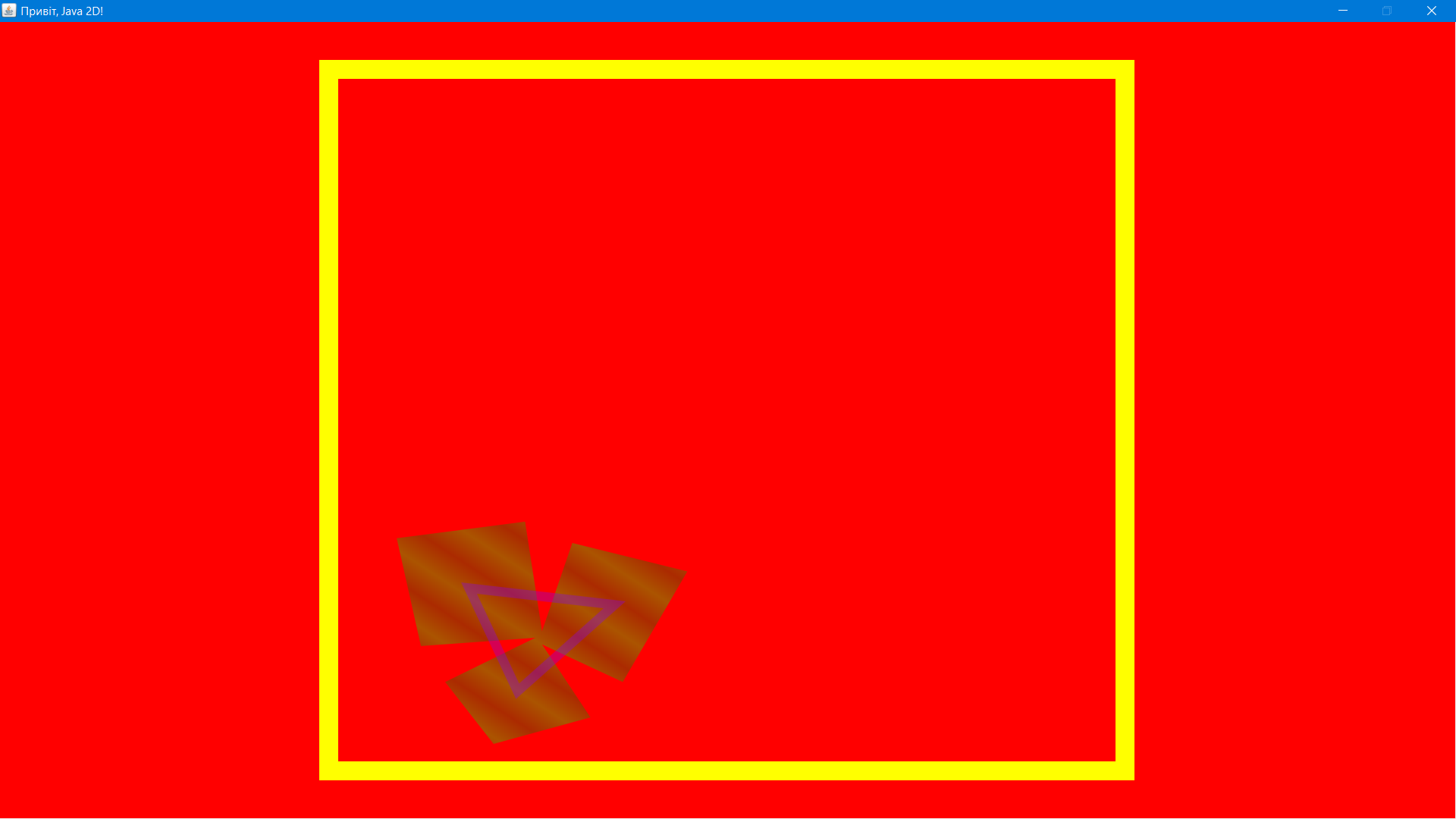


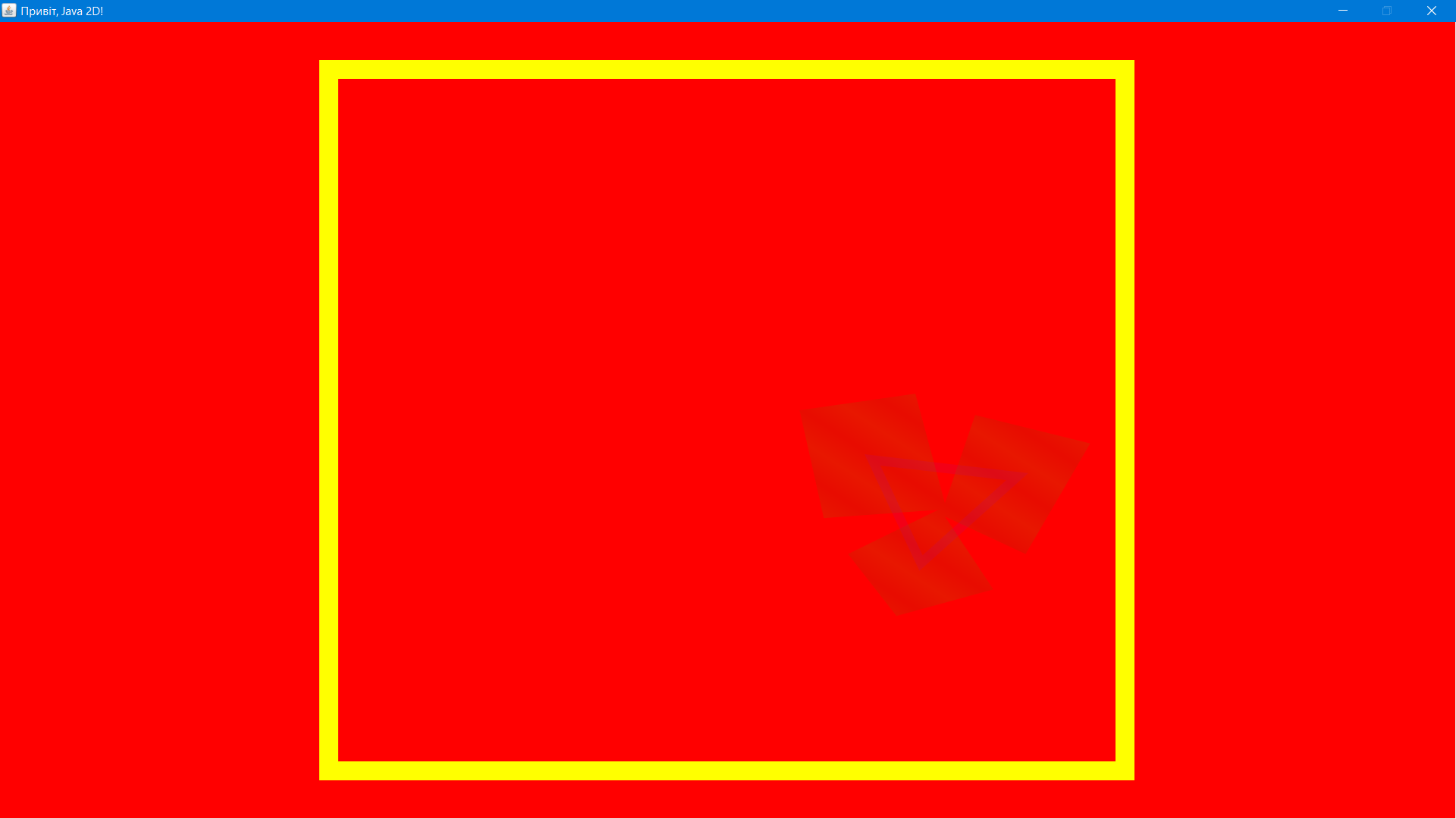
**Лістинг коду програми**

|  |
| --- |
| Skeleton.java |
| package com.company;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.geom.GeneralPath;  import javax.swing.\*;  enum Direction {  UP,  RIGHT,  DOWN,  LEFT  }  public class Skeleton extends JPanel implements ActionListener {  Graphics2D g2d;  Timer timer;  Direction dir = Direction.DOWN;  final Color borderColor = new Color(255, 255, 0);  final Color flowerColorLight = new Color(0, 255, 0);  final Color flowerColorDark = new Color(0, 128, 0);  final Color triangleColor = new Color(128, 0, 255);  private static double opacity = 0.5, ov = 0.01;  private static int a, w, h, cx, cy;  private static int fw, fh, x = 0, y = 0;  private final static int p = 7, v = 5, gap = 50;  public Skeleton() {  timer = new Timer(10, this);  timer.start();  }  public void paint(Graphics g) {  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(Color.RED);  g2d.clearRect(0, 0, w, h);  draw();  }  public void actionPerformed(ActionEvent e) {  update();  repaint();  }  public static void main(String[] args) {  JFrame frame = new JFrame("Привіт, Java 2D!");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(500, 500);  frame.setExtendedState(frame.getExtendedState() | JFrame.MAXIMIZED\_BOTH);  frame.setLocationRelativeTo(null);  frame.setResizable(false);  frame.add(new Skeleton());  frame.setVisible(true);  initScreenBounds(frame);  }  public static void initScreenBounds(JFrame frame) {  Dimension size = frame.getSize();  Insets insets = frame.getInsets();  w = size.width - insets.left - insets.right - 1;  h = size.height - insets.top - insets.bottom - 1;  cx = w / 2;  cy = h / 2;  fw = w / 3;  fh = a = h / 3;  x = cx - a + gap;  y = cy - a + gap;  }  public void draw() {  double[][] flower = {  { x - p, y },  { x - fw / 4f, y + fh / 32f },  { x - 3 \* fw / 10f, y - 3 \* fh / 8f },  { x - x / 32f, y - 7 \* fh / 16f },  { x, y - p },  { x + fw / 16f, y - 5 \* fh / 14f },  { x + 3 \* fw / 10f, y - fh / 4f },  { x + fw / 6f, y + fh / 6f },  { x, y + p },  { x + fw / 10f, y + 3 \* fh / 10f },  { x - fw / 10f, y + 4 \* fh / 10f },  { x - 2 \* fw / 10f, y + fh / 6f }  };  double[][] triangle = {  { x - 3 \* fw / 20f, y - 3 \* fh / 16f },  { x + 3 \* fw / 20f, y - fh / 8f },  { x - fw / 20f, y + 2 \* fh / 10f }  };  drawFigure(flower, true, flowerColorLight, flowerColorDark);  drawFigure(triangle, false, triangleColor, null);  drawBorder();  }  public void drawFigure(double[][] points, boolean isFill, Color color1, Color color2) {  GeneralPath flower = new GeneralPath();  flower.moveTo(points[0][0], points[0][1]);  for (int i = 1; i < points.length; i++) {  flower.lineTo(points[i][0], points[i][1]);  }  flower.closePath();  g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_OVER, (float) opacity));  if (color2 == null) g2d.setColor(color1);  else g2d.setPaint(new GradientPaint(5, 25, flowerColorLight, 20, 2, flowerColorDark, true));  if (isFill) {  g2d.fill(flower);  }  else {  BasicStroke bs = new BasicStroke(10, BasicStroke.CAP\_ROUND, BasicStroke.JOIN\_MITER);  g2d.setStroke(bs);  g2d.draw(flower);  }  }  public void drawBorder() {  BasicStroke bs = new BasicStroke(20, BasicStroke.CAP\_ROUND, BasicStroke.JOIN\_MITER);  g2d.setStroke(bs);  g2d.setColor(borderColor);  int size = h - 2 \* gap;  g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_OVER, 1f));  g2d.drawRect((w - size) / 2 - gap, gap, size + gap \* 2, size);  }  public void update() {  updateOpacity();  updatePosition();  }  public void updateOpacity() {  double raw = opacity + ov;  if (raw < 0 || raw > 1) {  ov = -ov;  }  opacity += ov;  }  public void updatePosition() {  switch (dir) {  case UP -> {  y -= v;  if (y <= cy - a + gap) dir = Direction.LEFT;  }  case RIGHT -> {  x += v;  if (x >= cx + a - gap) dir = Direction.UP;  }  case DOWN -> {  y += v;  if (y >= cy + a - gap) dir = Direction.RIGHT;  }  case LEFT -> {  x -= v;  if (x <= cx - a + gap) dir = Direction.DOWN;  }  }  }  } |

**Результат**

****

****

****