Кафедра Вычислительной техники



Лабораторная работа №2-3

по дисциплине «Разработка клиент-серверных приложений»

на тему: «Разработка абстрактного базового класса. Параллелизм и потоки»

Группа: АВТ-918

Студент: Ванин К.Е.

Преподаватель: Токарев В.Г.

Содержание

[Цель работы 3](#_Toc55463600)

[Задание 3](#_Toc55463601)

[Проектирование программы 3](#_Toc55463602)

[«Составные части» программы 4](#_Toc55463604)

[Пример работы программы 10](#_Toc55463606)

[Вывод 1](#_Toc55463607)1

# Цель работы

1. Разработать абстрактный базовый класс графического объекта с данными и функционалом (абстрактными и переопределяемыми методами): координаты центра и размеры охватывающего прямоугольника, цвет, рисование объекта, проверка на принадлежность точки объекту, запись/чтение в различные потоки, а также все остальные, необходимые для реализации функционала в последующих работах.

2. Разработать два производных класса (по вариантам). Разработать оконное приложение с созданием и уничтожением объектов по координатам клика мыши. Объекты сохраняются в векторе в виде ссылок на базовый класс.

3. Реализовать процесс движения объектов (вид движения по вариантам для каждого типа), остановки и возобновления движения по отдельности и всех вместе. Внести в базовый класс функционал движения, в производный – алгоритм движения для конкретного типа.

# Задание

Объект – Смайлик. Бывают двух видов: грустный и веселый. Грустный смайлик: вращается по окружности со случайным радиусов. Веселый смайлик: вращение вокруг своей оси и синусоидальная пульсация размера фигуры.

# Проектирование программы

* **public class Main – Основной класс запуска программы**
* **public class Screen extends javax.swing.JPanel implements MouseListener – параметры окна, класс управлением объектами**
* **public class Threads extends Thread – класс управления потоками**
* **public class HelpListener implements WindowListener – вспомогательный класс для Windows**
* **abstract public class Smile – абстрактный класс**
* **public class SmileHappy extends Smile – класс объекта – веселый смайл**
* **public class SmileUnHappy extends Smile – класс объекта – не веселый смайл**

# «Составные части» программы

**Класс Main**

package java\_laba\_2;  
  
import java.awt.Dimension;  
import javax.swing.JFrame;  
  
public class Main {  
  
 public static void main(String []args){  
 Screen s = new Screen();  
 JFrame frame = new JFrame();  
 frame.setVisible(true);  
 frame.setSize(new Dimension(800, 600));  
 frame.add(s);  
 frame.addMouseListener(s);  
 frame.addWindowListener(new HelpListener());  
   
 while (!s.isExit())  
 s.Update();  
 }  
}

**Класс Screen**

package java\_laba\_2;  
  
import java.awt.Graphics;  
import java.awt.Graphics2D;  
import java.awt.RenderingHints;  
import java.util.ArrayList;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.util.Random;  
  
public class Screen extends javax.swing.JPanel implements MouseListener {  
  
 private ArrayList<Smile> objects = new ArrayList<>();  
 public long timestep = 1000/60;  
 private long prev\_t = 0;  
   
 void DrawObjects(Graphics g) { //Описание в консоли  
 Graphics2D g2d = (Graphics2D) g;  
   
 long t = System.*currentTimeMillis*();  
 while (t-prev\_t < timestep) t = System.*currentTimeMillis*();  
   
 float dt = (float)(t-prev\_t)/1000.f;  
 for (Smile o : objects)  
 {  
 Threads td = new Threads(o,dt);  
 td.start();  
 try { td.join(); } catch(Exception ex) {}  
 g2d.drawImage(td.get(), 0, 0, null);  
 }  
   
 prev\_t = t;  
 }  
   
 public Smile pickObject(float x, float y) { //Координаты объетка  
 for (Smile o : objects)  
 {  
 if (o.isInside(x, y))  
 return o;  
 }  
 return null;  
 }  
   
 public void addObject(Smile o) { //Добавление объекта  
 objects.add(o);  
 }  
   
 @Override  
 public void paintComponent(Graphics g) { //Рисование компонентов  
 ((Graphics2D)g).setRenderingHint(RenderingHints.*KEY\_ANTIALIASING*, RenderingHints.*VALUE\_ANTIALIAS\_ON*);  
 super.paintComponent(g);  
 DrawObjects(g);  
 }  
  
 @Override  
 public void mouseClicked(MouseEvent e) { //Создание и удаление объекта  
 int x = e.getX() - 10;  
 int y = e.getY() - 10;  
   
 if (e.getButton() == MouseEvent.*BUTTON1*)  
 {  
   
 System.*out*.println("CLICKED ON: " + Integer.*toString*(x) + "X " + Integer.*toString*(y) + "Y");  
   
 Smile o = this.pickObject(x,y);  
 if (o!=null) {  
 objects.remove(o);  
 System.*out*.println(" > DELETED SMILE");  
 }  
 else   
 {  
 Random r = new Random();  
 boolean i = r.nextBoolean();  
 if (i) {  
 SmileHappy s = new SmileHappy();  
 s.centerX = x;  
 s.centerY = y;  
 this.addObject(s);}  
 else  
 {  
 SmileUnhappy s = new SmileUnhappy();  
 s.centerX = x;  
 s.centerY = y;  
 this.addObject(s);  
 }  
 System.*out*.println(" > ADDED SMILE");  
 }  
 }  
   
 this.repaint();  
 }  
   
 void Update() { repaint(); }  
 boolean isExit() { return false;}  
   
 @Override  
 public void mousePressed(MouseEvent e) {}  
  
 @Override  
 public void mouseReleased(MouseEvent e) {}  
  
 @Override  
 public void mouseEntered(MouseEvent e) {}  
  
 @Override  
 public void mouseExited(MouseEvent e) {}  
   
}

**Класс Smile**

package java\_laba\_2;  
  
import java.awt.Color;  
import java.awt.Graphics2D;  
  
abstract public class Smile {  
   
 public abstract void Draw(Graphics2D g, float t);  
   
 public boolean isInside(float x, float y) {  
 return (x > rectX1()) && (x < rectX1() + animW)   
 && (y > rectY1()) && (y < rectY1() + animH);  
 }  
   
 // исходные положение и размеры  
 protected float centerX = 50, centerY = 50;  
   
 protected float rectWidth = 50, rectHeight = 50;  
 // анимированинные положение и размеры  
 protected float animW, animH, animCX, animCY;  
   
 public float rectX1() { return centerX - animW/2;}  
 public float rectY1() { return centerY - animH/2;}  
  
 protected Color color = Color.*RED*;  
}

**Класс SmileHappy**

package java\_laba\_2;  
  
import java.awt.Color;  
import java.awt.Graphics2D;  
import java.awt.geom.AffineTransform;  
import java.awt.geom.Ellipse2D;  
  
public class SmileHappy extends Smile {  
   
 public SmileHappy() { color = Color.*GREEN*;}  
   
 @Override  
 public void Draw(Graphics2D g, float t)   
 {  
 Graphics2D g2d = g;  
 x+=t;  
 animH = Math.*round*(rectHeight\*( 1f - 0.5f\*Math.*sin*(2\*x)));//размер и сжатие  
 animW = Math.*round*(rectHeight\*( 1f - 0.5f\*Math.*sin*(3\*x)));  
  
 g2d.setTransform(AffineTransform.*getRotateInstance*(x,centerX,centerY));  
 g2d.fillOval((int)rectX1(), (int)(rectY1()), (int)animW, (int)animH);  
 Color c = (Color)g2d.getPaint();  
 g2d.setPaint(Color.*BLACK*);  
 float margin = 0.2f;  
 // рот  
 g2d.drawArc((int)(rectX1()+ animW\*margin), (int)(rectY1()+ animH\*margin),   
 (int)(animW\*(1.0f-2\*margin)), (int)(animH\*(1.0f-2\*margin)), 180, 180);  
 // глаза  
 g2d.drawLine((int)(rectX1()+ 1.5\*animW\*margin),   
 (int)(rectY1()+ animH\*margin),   
 (int)(rectX1()+ 1.5\*animW\*margin),   
 (int)(rectY1()+ 2\*animH\*margin));  
 g2d.drawLine((int)(rectX1()+ animW \*(1-1.5\*margin)) ,   
 (int)(rectY1()+ animH\*margin),   
 (int)(rectX1()+ animW \*(1-1.5\*margin)),   
 (int)(rectY1()+ 2\*animH\*margin));  
 g2d.draw(new Ellipse2D.Float(rectX1() + 16\*animW\*margin/7,  
 rectY1() + 2\*animH\*margin,  
 animW\*margin/3,  
 animH\*margin/3));  
 g2d.setPaint(c);  
 }  
 float x = 0.0f;  
}

**Класс SmileUnhappy**

package java\_laba\_2;  
  
import java.awt.\*;  
import java.awt.geom.AffineTransform;  
import java.awt.geom.Ellipse2D;  
  
public class SmileUnhappy extends Smile {  
   
 public SmileUnhappy() { color = Color.*ORANGE*;}  
 double x = 0;double x1 = 0;  
 double rand = Math.*random*();  
 double r = rand\*100;  
 @Override  
 public void Draw(Graphics2D g, float t)   
 {  
 x += t;  
 Graphics2D g2d = g;  
 animH = rectHeight;  
 animW = rectWidth;  
 x1=1.f - 0.3f\*Math.*atan*(x);  
  
 g2d.setTransform(AffineTransform.*getRotateInstance*(x,centerX+r,centerY+r));  
 g2d.fillOval((int)rectX1(), (int)rectY1(), (int)rectWidth, (int)rectHeight);  
 Color c = (Color)g2d.getPaint();  
 g2d.setPaint(Color.*BLACK*);  
   
 float margin = 0.2f;  
 // рот  
 g2d.drawArc((int)(rectX1()+ rectWidth\*margin), (int)(centerY+ rectHeight\*margin),   
 (int)(rectWidth\*(1.0f-2\*margin)), (int)(rectHeight\*(1.0f-2\*margin)/2), 30, 120);  
   
 // глаза  
 g2d.drawLine((int)(rectX1()+ 1.5\*rectWidth\*margin),   
 (int)(rectY1()+ rectHeight\*margin),   
 (int)(rectX1()+ 1.5\*rectWidth\*margin),   
 (int)(rectY1()+ 2\*rectHeight\*margin));  
 g2d.drawLine((int)(rectX1()+ rectWidth \*(1-1.5\*margin)) ,   
 (int)(rectY1()+ rectHeight\*margin),   
 (int)(rectX1()+ rectWidth \*(1-1.5\*margin)),   
 (int)(rectY1()+ 2\*rectHeight\*margin));  
 g2d.draw(new Ellipse2D.Float(rectX1() + 16\*animW\*margin/7,  
 rectY1() + 2\*animH\*margin,  
 animW\*margin/3,  
 animH\*margin/3));  
 g2d.setPaint(c);  
 }  
}

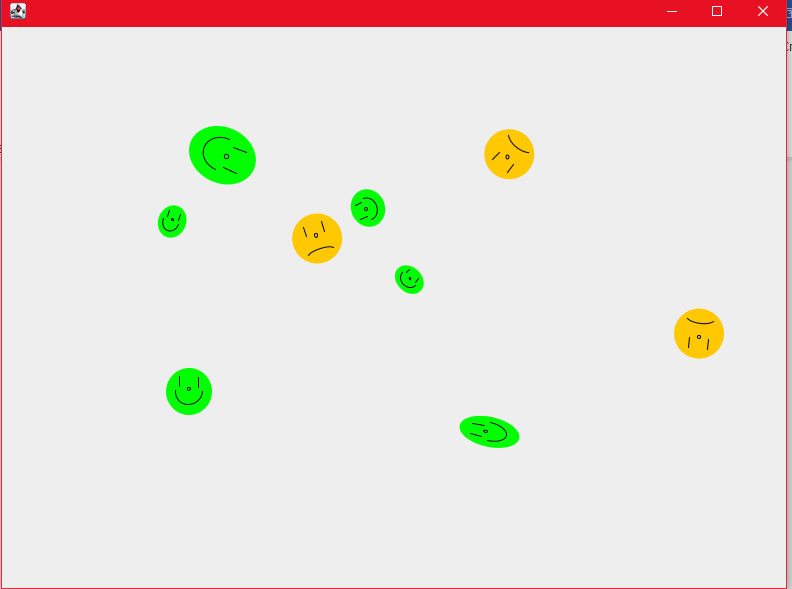
**Класс Treads**

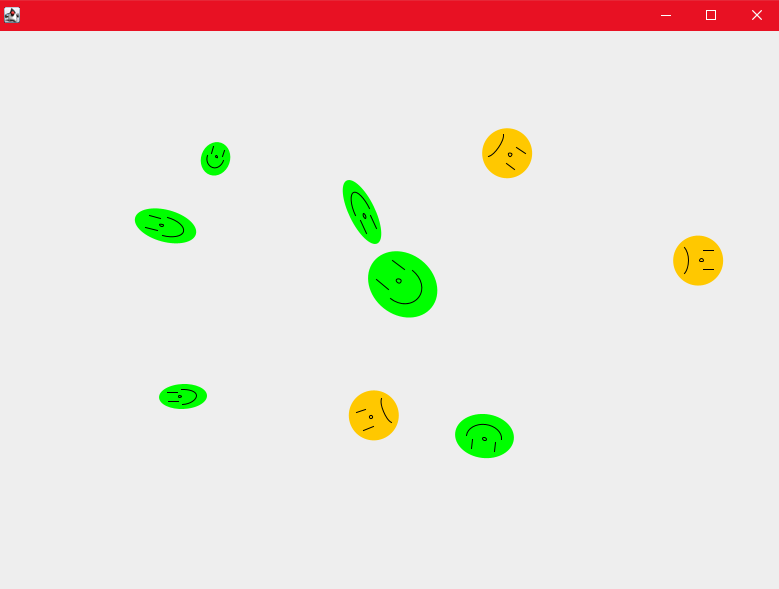
package java\_laba\_2;  
  
import java.awt.Graphics2D;  
import java.awt.RenderingHints;  
import java.awt.image.BufferedImage;  
  
public class Threads extends Thread {  
  
 private Smile \_d;  
 float dt;  
 private BufferedImage img;  
 public Threads(Smile d, float delta) {  
 super();  
 \_d = d;  
 dt = delta;  
 }  
 public void paint() {  
 BufferedImage out = new BufferedImage(800,600,BufferedImage.*TYPE\_INT\_ARGB*);  
 Graphics2D g = out.createGraphics();  
 RenderingHints rh = new RenderingHints(  
 RenderingHints.*KEY\_ANTIALIASING*,  
 RenderingHints.*VALUE\_ANTIALIAS\_ON*);  
 g.setRenderingHints(rh);  
 g.setPaint(\_d.color);  
 \_d.Draw(g, dt);  
   
 img = out;  
 }  
  
 @Override  
 public void run() {  
 paint();  
 }  
   
 public BufferedImage get()  
 {  
 return img;  
 }  
}

**Класс HelpListener**

package java\_laba\_2;  
  
import java.awt.event.WindowEvent;  
import java.awt.event.WindowListener;  
  
public class HelpListener implements WindowListener {  
  
 @Override  
 public void windowOpened(WindowEvent e) {  
 }  
  
 @Override  
 public void windowClosing(WindowEvent e) {  
 System.*exit*(0);  
 }  
  
 @Override  
 public void windowClosed(WindowEvent e) {  
 }  
  
 @Override  
 public void windowIconified(WindowEvent e) {  
 }  
  
 @Override  
 public void windowDeiconified(WindowEvent e) {  
 }  
  
 @Override  
 public void windowActivated(WindowEvent e) {  
 }  
  
 @Override  
 public void windowDeactivated(WindowEvent e) {  
 }  
   
}

# Пример работы программы





# Вывод

В результате выполнения лабораторной работы была получена программа для создания объектов, принадлежащих к двум разным классам, обладающими разными характеристиками.