Кафедра Вычислительной техники



Лабораторная работа №4

по дисциплине «Разработка клиент-серверных приложений»

на тему: «Работа с файлами»

Группа: АВТ-918

Студент: Ванин К.Е.

Преподаватель: Токарев В.Г.

Содержание

[Цель работы 3](#_Toc55463600)

[Проектирование программы 3](#_Toc55463602)

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

[Пример работы программы 10](#_Toc55463606)

[Вывод 1](#_Toc55463607)1

# Цель работы

В базовом классе реализовать (объявить) методы (абстрактные) чтения/записи объекта в двоичный и текстовый потоки. Переопределить во всех производных классах. В оконном классе реализовать методы сохранения и загрузки вектора объектов в тестовый, двоичный и сериализуемый файл (двоичная сериализация средствами Java и XML-сериализация с использованием библиотеки xstream). Формат двоичного и текстового файла содержит начальный счетчик объектов, а затем для каждого их них – имя и содержимое. При загрузке из файла создавать объект файла произвольного класса с помощью средств рефлексии (методы Class.forName и Class.newInstance)

# Проектирование программы

* **public class Main – Основной класс запуска программы**
* **public class Screen extends javax.swing.JPanel implements MouseListener, KeyListener – 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 implements Serializable – абстрактный класс**
* **public class SmileHappy extends Smile implements Serializable – класс объекта – веселый смайл**
* **public class SmileUnhappy extends Smile implements Serializable – 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.addKeyListener(s);  
 frame.addWindowListener(new HelpListener());  
  
 while (!s.isExit())  
 s.Update();  
 }  
  
}

**Класс Screen**

package java\_laba\_2;  
  
import com.thoughtworks.xstream.XStream;  
import com.thoughtworks.xstream.io.xml.DomDriver;  
import java.awt.Graphics;  
import java.awt.Graphics2D;  
import java.awt.RenderingHints;  
import java.awt.event.KeyEvent;  
import java.util.ArrayList;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.awt.event.KeyListener;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.io.OutputStreamWriter;  
import java.util.Random;  
  
public class Screen extends javax.swing.JPanel implements MouseListener, KeyListener {  
  
 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;  
 }  
  
 void DrawRects(Graphics g) {}  
  
 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);  
 }  
  
 public void deleteObject(Smile o)  
 {  
 objects.remove(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;}  
  
 void WriteText(String filename)  
 {  
 System.*out*.println("SAVING AS TEXT");  
 try {  
  
 BufferedWriter bwriter = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(filename), "UTF-8"));  
 int count = objects.size();  
 bwriter.write(Integer.*toString*(count) + '\n');  
 for (Smile o : objects)  
 {  
 String name = o.getClass().getName();  
 bwriter.write(name + '\n');  
 o.writeText(bwriter);  
 }  
 bwriter.flush();  
 bwriter.close();  
 }  
 catch (IOException e) { e.printStackTrace(); }  
 }  
  
 void ReadText(String filename)  
 {  
 System.*out*.println("LOADING FROM TEXT");  
 try {  
 objects.clear();  
 BufferedReader breader = new BufferedReader(new InputStreamReader(new FileInputStream(filename), "UTF-8"));  
 String scount = breader.readLine();  
 int count = Integer.*parseInt*(scount);  
 System.*out*.println(Integer.*toString*(count));  
  
 for (int i = 0; i < count; i++)  
 {  
 String name = breader.readLine();  
  
 Class<?> clazz = Class.*forName*(name);  
 Smile o = (Smile)clazz.newInstance();  
 o.readText(breader);  
 objects.add(o);  
 }  
  
 breader.close();  
 }  
 catch (Exception e) { e.printStackTrace(); }  
  
 }  
  
 void WriteBinary(String filename)  
 {  
 System.*out*.println("SAVING AS BINARY");  
  
 try {  
 DataOutputStream dos = new DataOutputStream(new FileOutputStream(filename));  
 int count = objects.size();  
 dos.writeInt(count);  
 for (Smile o : objects)  
 {  
 String name = o.getClass().getName();  
 dos.writeInt(name.length());  
 dos.writeChars(name);  
 o.writeBinary(dos);  
 }  
 dos.close();  
 }  
 catch (Exception e) { e.printStackTrace(); }  
 }  
  
 void ReadBinary(String filename)  
 {  
 System.*out*.println("LOADING FROM BINARY");  
 try {  
 objects.clear();  
 DataInputStream dis = new DataInputStream(new FileInputStream(filename));  
 int count = dis.readInt();  
  
 for (int i = 0; i < count; i++)  
 {  
 int nameLength = dis.readInt();  
 StringBuilder sb = new StringBuilder();  
 for (int j = 0; j < nameLength; j++)  
 sb.append(dis.readChar());  
 String name = sb.toString();  
  
 Class<?> clazz = Class.*forName*(name);  
 Smile o = (Smile)clazz.newInstance();  
 o.readBinary(dis);  
 objects.add(o);  
 }  
 }  
 catch (Exception e) { e.printStackTrace(); }  
 }  
  
 void Serialize(String filename)  
 {  
 System.*out*.println("SERIALIZING");  
 try {  
 ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename));  
 oos.writeObject(objects);  
 oos.flush();  
 oos.close();  
 }  
 catch (Exception e) { e.printStackTrace(); }  
 }  
  
 void Deserialize(String filename)  
 {  
 System.*out*.println("DESERIALIZING");  
 try {  
 ObjectInputStream oin = new ObjectInputStream(new FileInputStream(filename));  
 objects = (ArrayList<Smile>)oin.readObject();  
 }  
 catch (Exception e) { e.printStackTrace(); }  
 }  
  
 void SerializeXML(String filename)  
 {  
 System.*out*.println("SERIALIZING TO XML");  
 try {  
 XStream x = new XStream(new DomDriver());  
 x.toXML(objects, new FileOutputStream(new File(filename)));  
 }  
 catch (Exception e) { e.printStackTrace(); }  
  
 }  
  
 void DeserializeXML(String filename)  
 {  
 System.*out*.println("DESERIALIZING FROM XML");  
 try {  
 XStream x = new XStream(new DomDriver());  
 objects = (ArrayList<Smile>)x.fromXML(new File(filename));  
 }  
 catch (Exception e) { e.printStackTrace(); }  
 }  
 @Override  
 public void mousePressed(MouseEvent e) {}  
  
 @Override  
 public void mouseReleased(MouseEvent e) {}  
  
 @Override  
 public void mouseEntered(MouseEvent e) {}  
  
 @Override  
 public void mouseExited(MouseEvent e) {}  
  
 @Override  
 public void keyTyped(KeyEvent e) {  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 int code = e.getKeyCode();  
 if (code == KeyEvent.*VK\_F1* ) { WriteText("text.txt"); }  
 else if(code ==KeyEvent.*VK\_F2* ) { ReadText("text.txt"); }  
 else if(code ==KeyEvent.*VK\_F3*) { WriteBinary("bin.dat"); }  
 else if(code ==KeyEvent.*VK\_F4*) { ReadBinary("bin.dat"); }  
 else if(code ==KeyEvent.*VK\_F5*) { Serialize("ser.dat"); }  
 else if(code ==KeyEvent.*VK\_F6*) { Deserialize("ser.dat"); }  
 else if(code ==KeyEvent.*VK\_F7*) { SerializeXML("ser.xml"); }  
 else if(code ==KeyEvent.*VK\_F8*) { DeserializeXML("ser.xml"); }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 }  
  
}

**Класс Smile**

package java\_laba\_2;  
  
import java.awt.Color;  
import java.awt.Graphics;  
import java.awt.Graphics2D;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.IOException;  
import java.io.Serializable;  
  
abstract public class Smile implements Serializable {  
  
 public abstract void Draw(Graphics2D g, float t);  
  
 abstract void readText(BufferedReader breader) throws IOException;  
 abstract void writeText(BufferedWriter bwriter) throws IOException;  
  
 abstract void writeBinary(DataOutputStream dos) throws IOException;  
 abstract void readBinary(DataInputStream dis) throws IOException;  
  
 public boolean isInside(float x, float y) {  
 return (x > rectX1()) && (x < rectX1() + animW)  
 && (y > rectY1()) && (y < rectY1() + animH);  
 }  
  
 public void DrawRect(Graphics g)  
 {  
 Graphics2D g2d = (Graphics2D) g;  
  
  
 int l = 10;  
 Color c = (Color)g2d.getPaint();  
 g2d.setPaint(Color.*BLACK*);  
  
 g2d.drawLine((int)rectX1(), (int)rectY1(), (int)rectX1() + l, (int)rectY1());  
 g2d.drawLine((int)rectX1(), (int)rectY1(), (int)rectX1(), (int)rectY1() + l);  
  
 g2d.drawLine((int)rectX2(), (int)rectY1(), (int)rectX2() - l, (int)rectY1());  
 g2d.drawLine((int)rectX2(), (int)rectY1(), (int)rectX2(), (int)rectY1() + l);  
  
 g2d.drawLine((int)rectX1(), (int)rectY2(), (int)rectX1() + l, (int)rectY2());  
 g2d.drawLine((int)rectX1(), (int)rectY2(), (int)rectX1(), (int)rectY2() - l);  
  
 g2d.drawLine((int)rectX2(), (int)rectY2(), (int)rectX2() - l, (int)rectY2());  
 g2d.drawLine((int)rectX2(), (int)rectY2(), (int)rectX2(), (int)rectY2() - l);  
  
  
  
 g2d.setPaint(c);  
 }  
  
 // 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;}  
 public float rectX2() { return centerX + animW/2;}  
 public float rectY2() { 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.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.IOException;  
import java.io.Serializable;  
  
public class SmileHappy extends Smile implements Serializable {  
  
 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.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.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.setPaint(c);  
 }  
  
 float x = 0.0f;  
  
 @Override  
 void writeText(BufferedWriter bwriter) throws IOException {  
 bwriter.write(Float.*toString*(x) + '\n');  
 bwriter.write(Float.*toString*(centerX)+ '\n');  
 bwriter.write(Float.*toString*(centerY)+ '\n');  
 }  
  
 @Override  
 void readText(BufferedReader breader) throws IOException {  
  
 String str\_x = breader.readLine();  
 String str\_centerX = breader.readLine();  
 String str\_centerY = breader.readLine();  
 x = Float.*parseFloat*(str\_x);  
 centerX = Float.*parseFloat*(str\_centerX);  
 centerY = Float.*parseFloat*(str\_centerY);  
 }  
  
 @Override  
 void writeBinary(DataOutputStream dos) throws IOException {  
 dos.writeFloat(x);  
 dos.writeFloat(centerX);  
 dos.writeFloat(centerY);  
  
 }  
  
 @Override  
 void readBinary(DataInputStream dis) throws IOException {  
 x = dis.readFloat();  
 centerX = dis.readFloat();  
 centerY = dis.readFloat();  
 }  
}

**Класс SmileUnhappy**

package java\_laba\_2;  
  
import java.awt.Color;  
import java.awt.Graphics2D;  
import java.awt.geom.AffineTransform;  
import java.awt.geom.Ellipse2D;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.IOException;  
import java.io.Serializable;  
  
public class SmileUnhappy extends Smile implements Serializable {  
   
 public SmileUnhappy() { color = Color.*RED*;}  
 float x = 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;  
  
 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);  
 }  
   
   
 @Override  
 void writeText(BufferedWriter bwriter) throws IOException {  
 bwriter.write(Float.*toString*(x) + '\n');  
 bwriter.write(Float.*toString*(centerX)+ '\n');  
 bwriter.write(Float.*toString*(centerY)+ '\n');  
 }  
  
   
 @Override  
 void readText(BufferedReader breader) throws IOException {  
   
 String str\_x = breader.readLine();  
 String str\_centerX = breader.readLine();  
 String str\_centerY = breader.readLine();  
 x = Float.*parseFloat*(str\_x);  
 centerX = Float.*parseFloat*(str\_centerX);  
 centerY = Float.*parseFloat*(str\_centerY);  
 }  
  
 @Override  
 void writeBinary(DataOutputStream dos) throws IOException {  
 dos.writeFloat(x);  
 dos.writeFloat(centerX);  
 dos.writeFloat(centerY);  
 }  
  
 @Override  
 void readBinary(DataInputStream dis) throws IOException {  
 x = dis.readFloat();  
 centerX = dis.readFloat();  
 centerY = dis.readFloat();  
 }  
   
}

**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;  
 }  
}

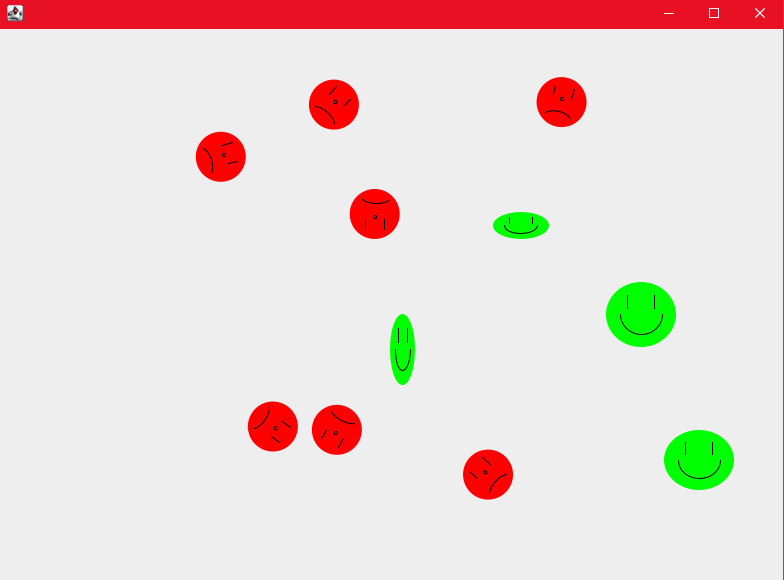
**Класс HelpListener**

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) {  
 }  
  
}

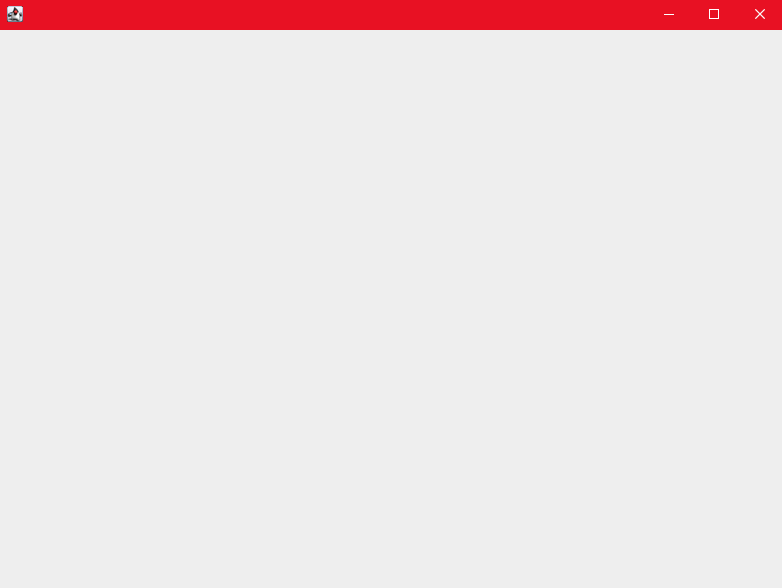
# 

**Пример работы программы**

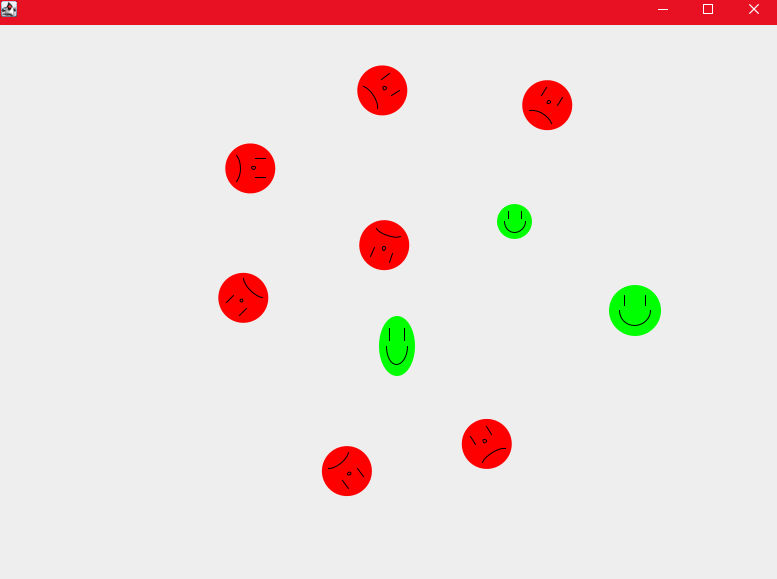
Сохраняем объекты программы (F1).



Открываем новый проект.



Выгружаем объекты (F2).



# Вывод

В результате выполнения лабораторной работы были реализованы методы чтения/записи объекта в двоичном и текстовом потоках.

Переопределены во всех производных классах.

В оконном классе реализованы методы сохранения и загрузки вектора объектов в тестовый, двоичный и сериализуемый файл.

Формат двоичного и текстового файла содержит начальный счетчик объектов, и для каждого их имена и содержимое.

При загрузке из файла создан объект файла произвольного класса с помощью средств рефлексии.