КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. А.Н.Туполева - КАИ

(КНИТУ-КАИ)

Отделение среднего профессионального образования института

Компьютерных технологий и защиты информации

«Колледж информационных технологий»

Лабораторная работа №4

по предмету:

Инструментальные средства разработки программного обеспечения

Выполнил:

студент 4 курса

Группы 4441

Зарипов Ислам Радифович

Казань 2019 г.

**Задание:**

1. В зависимости от варианта выполняется разработка приложения. Пример

приложения похожего на то, которое нужно разработать см. в примере 1.

2. Разрабатываемое приложение должно состоять из двух окон: управляющего (УО) и дочернего, т.е. демонстрационного (ДО). В первом должны отображаться интерфейсные элементы (кнопки, текстовые поля, выпадающие списки, элементы выбора цвета), во втором в зависимости от настроек, сделанных в первом окне, должны передвигаться фигурки или простые объекты (ФиО).

3. При выставлении в УО некоторых настроек запуска очередной ФиО и нажатии на кнопку «Пуск» в ДО из левого верхнего угла в случайном направлении должен начать двигаться ФиО (!!! т.е. приращение координат «икс» и «игрек» ФиО может быть таким, что оно будет отличаться от траектории, совпадающей с биссектрисой угла вылета ФиО).

4. ФиО при движении в ДО должны отражаться от сторон окна по правилу: «угол отражения равен углу падения».

5. В УО должна быть предусмотрена возможность выбора уже запущенного ФиО и изменения его параметров (например, цвет, скорость).

6. Номер, который присваивается ФиО должен отображаться рядом с ним (ФИО), когда он (ФИО) перемещается в ДО.

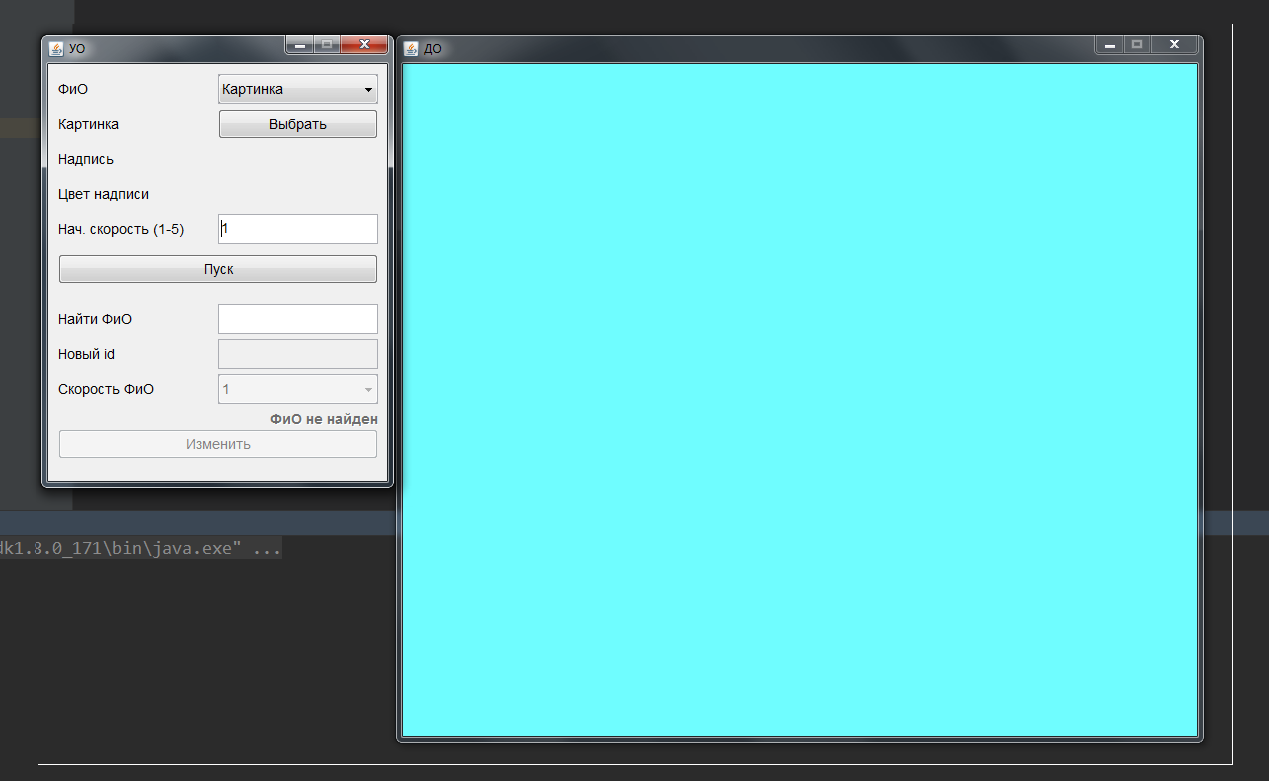
7. Все ФиО должны быть пронумерованы, при чём, если предусмотрено изменение номера ФиО, то необходимо обеспечить уникальность номера каждого ФиО.

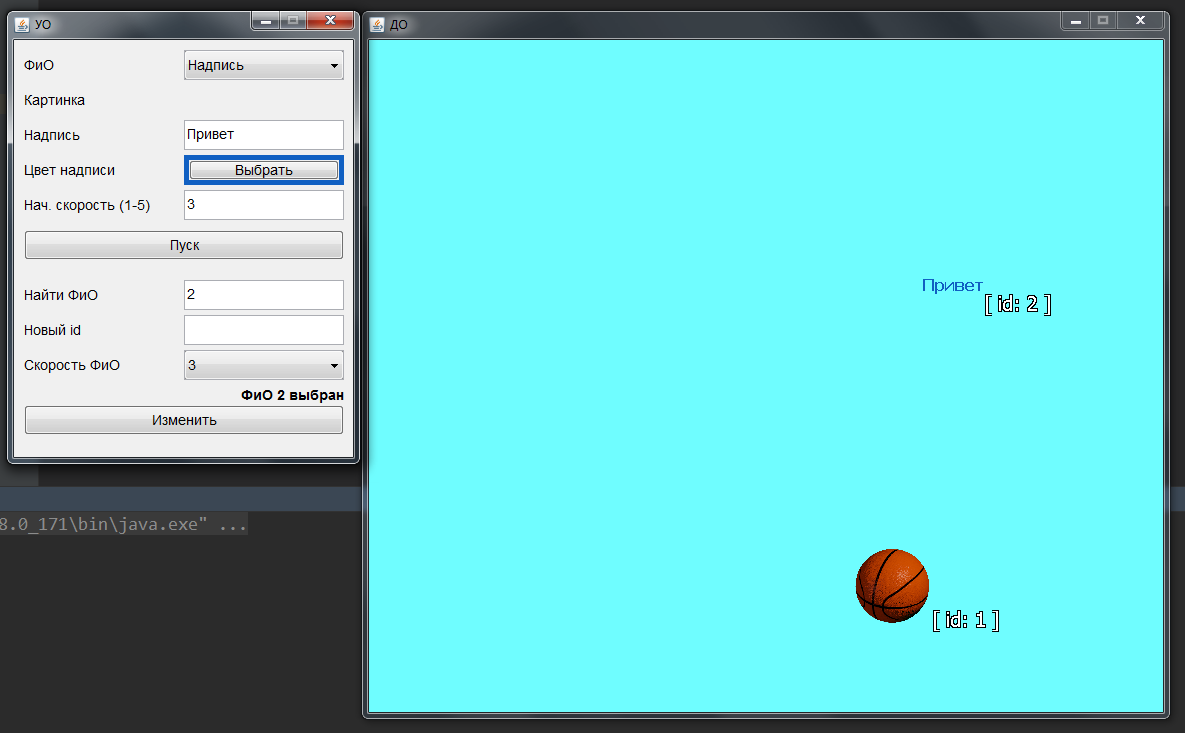
8. Приложение должно закрываться при закрывании любого из окон.

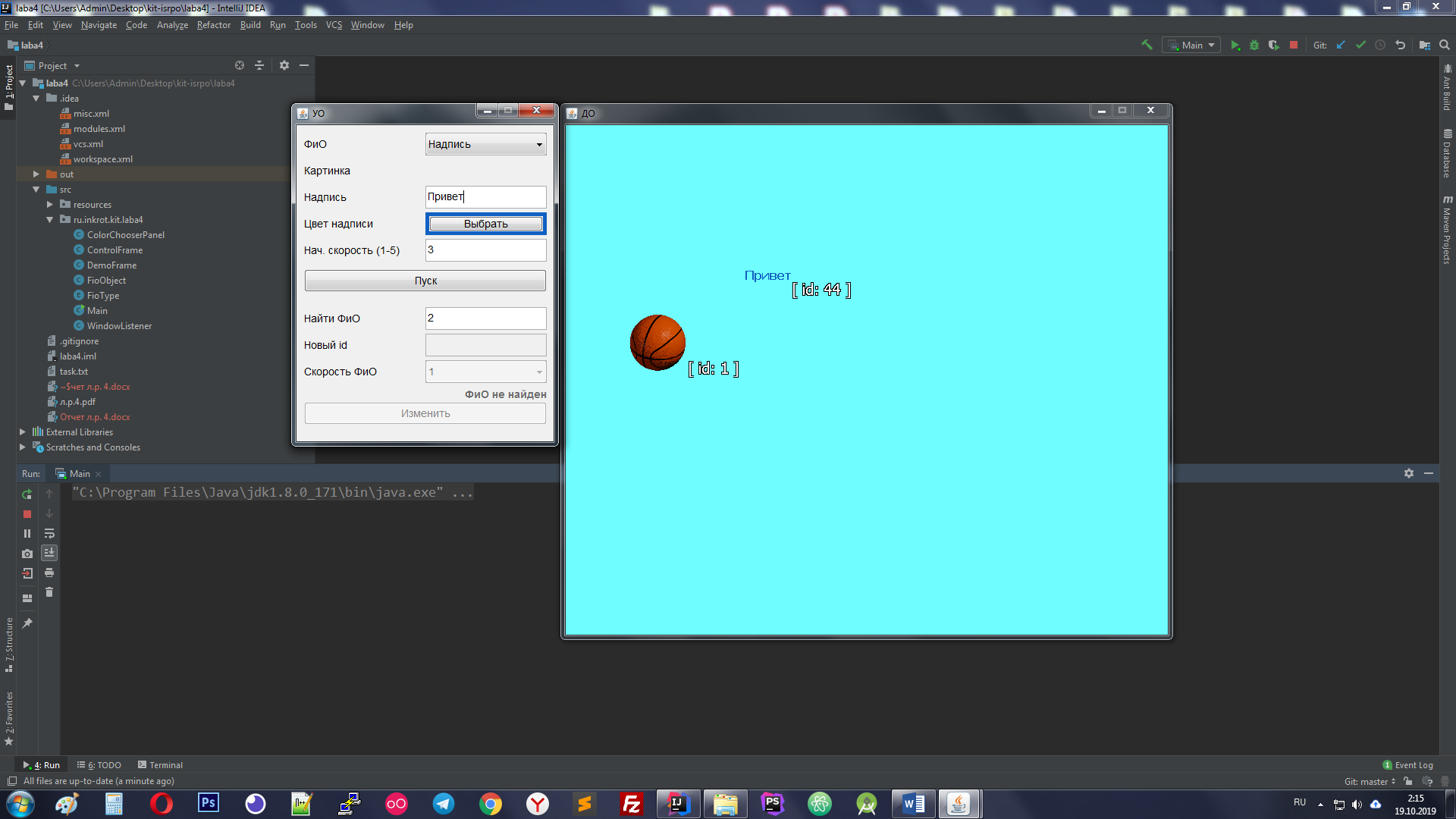
Вариант №2: 918 = 1110010110

|  |  |  |
| --- | --- | --- |
| Число ФиО | 2 | ЗАДАНО В КОДЕ УО |
| ФиО | 2 | ОБЕКТЫ (маленькая простая картинка или надпись, которая задаётся из УО) |
| Задание цвета текста ФиО | 2 | СТАНДАРТНЫЙ ЭЛЕМЕНТ ВЫБОРА ЦВЕТА |
| Выбор запускаемого ФИО | 1 | ВЫПАДАЮЩИЙ СПИСОК НАЗВАНИЙ |
| Задание начальной скорости ФиО | 1 | УКАЗАНИЕМ В ТЕКСТОВОМ ПОЛЕ |
| Способ выбора, запущенного ФиО | 2 | ИЗ ТЕКСТОВОГО ПОЛЯ, ГДЕ ВВОДИТСЯ НОМЕР ФиО |
| Присвоение номера ФиО | 1 | АВТОМАТИЧЕСКИ |
| Возможность смены номера ФиО из УО | 2 | ДА |
| Регулировка скорости перемещения, выбранного ФиО | 2 | ИЗ ВЫПАДАЮЩЕГО СПИСКА |
| Изменения размера окна отображения ФиО | 1 | НЕТ |

**Работа программы:**







Исходные коды:

package ru.inkrot.kit.laba4;  
  
import javax.swing.\*;  
import javax.swing.colorchooser.AbstractColorChooserPanel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.lang.reflect.Array;  
import java.lang.reflect.Field;  
  
class ColorChooserPanel extends JDialog implements ActionListener {  
  
 private final JColorChooser jCC;  
 private final JPanel panel;  
  
 ActionListener onColorChosen;  
  
 public ColorChooserPanel(final String title, ActionListener onColorChosen) {  
 this.setTitle(title);  
 this.onColorChosen = onColorChosen;  
 this.jCC = new JColorChooser();  
 try {  
 this.modifyJColorChooser();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 this.panel = new JPanel();  
 this.panel.add(this.jCC);  
 this.panel.setOpaque(false);  
 this.jCC.setOpaque(false);  
 this.jCC.setPreviewPanel(new JPanel());  
 this.jCC.setColor(120, 20, 57);  
 this.add(this.panel, BorderLayout.CENTER);  
 JButton button = new JButton("Ок");  
 button.addActionListener(this);  
 this.add(button, BorderLayout.SOUTH);  
  
 this.setDefaultCloseOperation(WindowConstants.HIDE\_ON\_CLOSE);  
 this.pack();  
 final Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();  
 this.setLocation((screenSize.width - this.getWidth()) / 2,  
 (screenSize.height - this.getHeight()) / 2);  
 this.setResizable(false);  
 }  
  
 @Override  
 public void actionPerformed(final ActionEvent e) {  
 e.setSource(jCC.getColor());  
 onColorChosen.actionPerformed(e);  
 }  
  
 private void modifyJColorChooser() throws Exception {  
 final AbstractColorChooserPanel[] panels = this.jCC.getChooserPanels();  
 for (final AbstractColorChooserPanel accp : panels) {  
 if (!accp.getDisplayName().equals("HSL")) {  
 this.jCC.removeChooserPanel(accp);  
 }  
 }  
 final AbstractColorChooserPanel[] colorPanels = this.jCC.getChooserPanels();  
 final AbstractColorChooserPanel cp = colorPanels[0];  
 Field f;  
 f = cp.getClass().getDeclaredField("panel");  
 f.setAccessible(true);  
 Object colorPanel;  
 colorPanel = f.get(cp);  
 Field f2;  
 f2 = colorPanel.getClass().getDeclaredField("spinners");  
 f2.setAccessible(true);  
 Object rows;  
 rows = f2.get(colorPanel);  
 final Object transpSlispinner = Array.get(rows, 3);  
 Field f3;  
 f3 = transpSlispinner.getClass().getDeclaredField("slider");  
 f3.setAccessible(true);  
 JSlider slider;  
 slider = (JSlider) f3.get(transpSlispinner);  
 slider.setVisible(false);  
 Field f4;  
 f4 = transpSlispinner.getClass().getDeclaredField("spinner");  
 f4.setAccessible(true);  
 JSpinner spinner;  
 spinner = (JSpinner) f4.get(transpSlispinner);  
 spinner.setVisible(false);  
 Field f5;  
 f5 = transpSlispinner.getClass().getDeclaredField("label");  
 f5.setAccessible(true);  
 JLabel label;  
 label = (JLabel) f5.get(transpSlispinner);  
 label.setVisible(false);  
 }  
}

package ru.inkrot.kit.laba4;  
  
import javax.imageio.ImageIO;  
import javax.imageio.ImageReader;  
import javax.swing.JPanel;  
import javax.swing.BorderFactory;  
import java.awt.\*;  
import java.awt.event.ItemEvent;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.text.DecimalFormat;  
import java.text.NumberFormat;  
import java.util.Collections;  
import java.util.Locale;  
import java.util.Set;  
import javax.swing.JLabel;  
import javax.swing.JComboBox;  
import javax.swing.JButton;  
import javax.swing.JTextField;  
import javax.swing.\*;  
import javax.swing.filechooser.FileNameExtensionFilter;  
  
public class ControlFrame extends JFrame {  
  
 // constants  
 private Font font = new Font("Arial", 0, 14);  
 public static int *WIDTH* = 345;  
 public static int *HEIGHT* = 445;  
  
 // ui  
 private JComboBox typeCombo;  
 private JButton chooseImageButton;  
 private JTextField textValueField;  
 private JButton textColorButton;  
 private JTextField startSpeedField;  
 private JDialog colorChooseDialog;  
 private JButton runButton;  
 private JTextField findIdField;  
 private JLabel foundObjectLabel;  
 private JTextField newIdField;  
 private JComboBox speedCombo;  
 private JButton editButton;  
 private KeyListener numbersKeyListener;  
  
 // control  
 private FioType selectedType = FioType.*PICTURE*;  
 private File selectedFile = null;  
 private Color textColor = Color.*BLACK*;  
 private FioObject selectedFio = null;  
  
 public ControlFrame() {  
 setTitle("УО");  
 int x = (Main.*SCREEN\_WIDTH* / 2) - (*WIDTH* + DemoFrame.*WIDTH*) / 2;  
 int y = (Main.*SCREEN\_HEIGHT* / 2) - DemoFrame.*WIDTH* / 2;  
 setLocation(x, y);  
 setSize(*WIDTH*, *HEIGHT*);  
 addWindowListener(new WindowListener());  
 *setDefaultLookAndFeelDecorated*(false);  
 setLayout(null);  
 setResizable(false);  
 initListeners();  
 createUI();  
  
 colorChooseDialog = new ColorChooserPanel("Выберите цвет", e -> {  
 textColor = (Color) e.getSource();  
 textColorButton.setBorder(BorderFactory.*createMatteBorder*(5, 5, 5, 5, textColor));  
 colorChooseDialog.setVisible(false);  
 });  
 colorChooseDialog.setSize(570,290);  
 colorChooseDialog.setLocationRelativeTo(null);  
 colorChooseDialog.setVisible(false);  
  
 new DemoFrame(this);  
 setVisible(true);  
 repaint();  
 }  
  
 private void initListeners() {  
 numbersKeyListener = new KeyListener() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 JTextField field = (JTextField) e.getSource();  
 char c = e.getKeyChar();  
 if (! isNumber(c)) e.consume();  
 if (field.equals(startSpeedField) && (! (c >= '1' && c <= '5') || field.getText().length() > 0)) e.consume();  
 }  
 @Override  
 public void keyPressed(KeyEvent e) {  
  
 }  
 @Override  
 public void keyReleased(KeyEvent e) {  
 JTextField field = (JTextField) e.getSource();  
 if (field.equals(findIdField)) findId();  
 }  
 };  
 }  
  
 private JLabel newLabel(String title) {  
 JLabel label = new JLabel(title);  
 label.setFont(font);  
 return label;  
 }  
  
 private JButton newButton(String title) {  
 JButton button = new JButton(title);  
 button.setFont(font);  
 button.setFocusable(false);  
 return button;  
 }  
  
 private JComboBox newCombo(String [] data) {  
 JComboBox comboBox = new JComboBox(data);  
 comboBox.setFont(font);  
 comboBox.setFocusable(false);  
 return comboBox;  
 }  
  
 private JTextField newTextField(String text) {  
 JTextField textField = new JTextField(text);  
 textField.setFont(font);  
 return textField;  
 }  
  
 private void createUI() {  
 int w = WIDTH - 25;  
 JPanel runPanel = new JPanel();  
 runPanel.setBounds(10, 10, w, 170);  
 runPanel.setLayout(new GridLayout(5, 2, 0, 5));  
 add(runPanel);  
  
 runPanel.add(newLabel("ФиО"));  
 runPanel.add(typeCombo = newCombo(new String[]{"Картинка", "Надпись"}));  
 typeCombo.addItemListener(e -> {  
 if (e.getStateChange() == ItemEvent.SELECTED) setFioType(e.getItem().toString());  
 });  
  
 runPanel.add(newLabel("Картинка"));  
 runPanel.add(chooseImageButton = newButton("Выбрать"));  
 chooseImageButton.addActionListener(e -> chooseImage());  
  
 runPanel.add(newLabel("Надпись"));  
 runPanel.add(textValueField = newTextField(""));  
  
 runPanel.add(newLabel("Цвет надписи"));  
 runPanel.add(textColorButton = newButton("Выбрать"));  
 textColorButton.setBorder(BorderFactory.createMatteBorder(5, 5, 5, 5, textColor));  
 textColorButton.addActionListener(e -> colorChooseDialog.setVisible(true));  
  
 runPanel.add(newLabel("Нач. скорость (1-5)"));  
 runPanel.add(startSpeedField = newTextField("1"));  
 startSpeedField.setTransferHandler(null);  
 startSpeedField.addKeyListener(numbersKeyListener);  
  
 add(runButton = newButton("Пуск"));  
 runButton.setBounds(10, runPanel.getY() + runPanel.getHeight() + 10, w, 30);  
 runButton.addActionListener(e -> runObject());  
  
 JPanel findPanel = new JPanel();  
 findPanel.setBounds(10, runPanel.getY() + runPanel.getHeight() + 60, w, 100);  
 findPanel.setLayout(new GridLayout(3, 2, 0, 5));  
 add(findPanel);  
  
 findPanel.add(newLabel("Найти ФиО"));  
  
 findPanel.add(findIdField = newTextField(""));  
 findIdField.setTransferHandler(null);  
 findIdField.addKeyListener(numbersKeyListener);  
  
 findPanel.add(newLabel("Новый id"));  
 findPanel.add(newIdField = newTextField(""));  
 newIdField.setTransferHandler(null);  
 newIdField.addKeyListener(numbersKeyListener);  
  
 findPanel.add(newLabel("Скорость ФиО"));  
 findPanel.add(speedCombo = newCombo(new String[]{"1", "2", "3", "4", "5"}));  
  
 add(foundObjectLabel = new JLabel("ФиО не найден", SwingConstants.RIGHT));  
 foundObjectLabel.setFont(new Font("Arial", 1, 14));  
 foundObjectLabel.setBounds(10, findPanel.getY() + findPanel.getHeight(), w, 30);  
  
 add(editButton = newButton("Изменить"));  
 editButton.setBounds(10, findPanel.getY() + findPanel.getHeight() + 25, w, 30);  
 editButton.addActionListener(e -> editFio());  
  
 setFioType("Картинка");  
 searchFailed();  
 }  
  
 private void editFio() {  
 if (selectedFio == null) return;  
 String newIdStr = newIdField.getText();  
 String newSpeedStr = speedCombo.getSelectedItem().toString();  
 try {  
 int newSpeed = Integer.parseInt(newSpeedStr);  
 selectedFio.setSpeed(newSpeed);  
 if (newIdStr.length() > 0) {  
 int newId = Integer.parseInt(newIdStr);  
 int result = selectedFio.setId(newId);  
 if (result == 1)  
 showMessageBox("Ошибка", "Идентификатор занят", JOptionPane.*ERROR\_MESSAGE*);  
 else if (result == 2)  
 showMessageBox("Ошибка", "Идентификатор должен быть в дипазоне от 1 до " + FioObject.*MAX\_NUMBER\_OF\_OBJECTS*, JOptionPane.*ERROR\_MESSAGE*);  
 }  
 findId();  
 } catch (Exception e) {}  
 }  
  
 private void showMessageBox(String title, String message, int type) {  
 JOptionPane.*showMessageDialog*(null, message, title, type);  
 }  
  
 private boolean isNumber(char ch){  
 return ch >= '0' && ch <= '9';  
 }  
  
 private void findId() {  
 selectedFio = null;  
 if (findIdField.getText().length() < 1) {  
 searchFailed();  
 return;  
 }  
 String idStr = findIdField.getText();  
 try {  
 int id = Integer.valueOf(idStr);  
 Set<FioObject> objects = Collections.synchronizedSet(FioObject.getAllObjects());  
 for (FioObject obj : objects) {  
 if (obj.getId() == id) {  
 selectedFio = obj;  
 searchSuccessful();  
 return;  
 }  
 }  
 searchFailed();  
 } catch (Exception exception) {  
 findIdField.setText("");  
 }  
 }  
  
 private void searchSuccessful() {  
 foundObjectLabel.setText("ФиО " + selectedFio.getId() + " выбран");  
 speedCombo.setSelectedItem(String.valueOf(selectedFio.getSpeed()));  
 newIdField.setEnabled(true);  
 speedCombo.setEnabled(true);  
 foundObjectLabel.setEnabled(true);  
 editButton.setEnabled(true);  
 }  
  
 private void searchFailed() {  
 foundObjectLabel.setText("ФиО не найден");  
 speedCombo.setSelectedIndex(0);  
 newIdField.setText("");  
 newIdField.setEnabled(false);  
 speedCombo.setEnabled(false);  
 foundObjectLabel.setEnabled(false);  
 editButton.setEnabled(false);  
 }  
  
 private void runObject() {  
 int speed = 1;  
 try {  
 speed = Integer.parseInt(startSpeedField.getText());  
 } catch (Exception e) {}  
 switch (selectedType) {  
 case PICTURE:  
 try {  
 if (selectedFile != null)  
 FioObject.addObject(ImageIO.read(selectedFile), speed);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 break;  
 case TEXT:  
 String text = textValueField.getText();  
 if (text.length() > 0)  
 FioObject.addObject(text, textColor, speed);  
 break;  
 }  
 findId();  
 }  
  
 private void setFioType(String option) {  
 if (option.equals("Картинка")) {  
 textValueField.setVisible(false);  
 textColorButton.setVisible(false);  
 chooseImageButton.setVisible(true);  
 selectedType = FioType.PICTURE;  
 } else if (option.equals("Надпись")) {  
 chooseImageButton.setVisible(false);  
 textValueField.setVisible(true);  
 textColorButton.setVisible(true);  
 selectedType = FioType.TEXT;  
 }  
 }  
  
 private void chooseImage() {  
 JFileChooser fc = new JFileChooser(System.getProperty("user.dir") + "\\src\\resources");  
 fc.setFileSelectionMode( JFileChooser.FILES\_ONLY);  
 fc.setFileFilter(new FileNameExtensionFilter("Картинки", "png", "jpg", "jpeg", "gif"));  
 if (fc.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION )  
 {  
 selectedFile = fc.getSelectedFile();  
 String name = selectedFile.getName();  
 chooseImageButton.setText(name);  
 }  
 }  
}

package ru.inkrot.kit.laba4;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.font.TextLayout;  
import java.util.Collections;  
import java.util.Set;  
  
public class DemoFrame extends JFrame {  
  
 // constants  
 public static int WIDTH = 800;  
 public static int HEIGHT = 700;  
 private int FPS = 60;  
  
 private DemoPanel demoPanel;  
  
 private Thread repaintThread;  
 private Thread movingThread;  
  
 public DemoFrame(ControlFrame controlFrame) {  
 setTitle("ДО");  
 setLocation(controlFrame.getX() + controlFrame.getWidth() + 10, controlFrame.getY());  
 //setUndecorated(true);  
 setSize(WIDTH, HEIGHT);  
 addWindowListener(new WindowListener());  
 setDefaultLookAndFeelDecorated(false);  
 setLayout(null);  
 setResizable(false);  
  
 repaintThread = new Thread(() -> {  
 while(true) {  
 demoPanel.repaint();  
 try {  
 Thread.sleep(1000 / FPS);  
 } catch (InterruptedException e) {}  
 }  
 });  
  
 movingThread = new Thread(() -> {  
 while(true) {  
 moveAll();  
 try {  
 Thread.sleep(1);  
 } catch (InterruptedException e) {}  
 }  
 });  
  
 demoPanel = new DemoPanel();  
 int mx = 0;  
 int my = 0;  
 demoPanel.setBounds(mx, my, getWidth() - mx, getHeight() - my);  
 add(demoPanel);  
  
 repaintThread.start();  
 movingThread.start();  
  
 setVisible(true);  
 }  
  
 private void moveAll() {  
 Set<FioObject> objects = FioObject.*getAllObjects*();  
 for (FioObject obj : objects) {  
 obj.nextStep(demoPanel.getWidth() - 6, demoPanel.getHeight() - 28);  
 }  
 }  
  
 class DemoPanel extends JPanel {  
  
 private Color backgroundColor = new Color(111, 253, 255);  
  
 private Font idTextFont = new Font("Tahoma", 0, 20);  
  
 public void paint(Graphics g) {  
 Graphics2D g2d = (Graphics2D) g;  
  
 // background  
 g2d.setColor(backgroundColor);  
 g2d.fillRect(0 , 0, getWidth(), getHeight());  
  
 // draw objects  
 Set<FioObject> objects = Collections.synchronizedSet(FioObject.getAllObjects());  
 for (FioObject obj : objects) {  
 int x = obj.getX(), y = obj.getY();  
 if (obj.getType() == FioType.PICTURE) {  
 int size = FioObject.IMAGE\_SIZE;  
 g2d.drawImage(obj.getImage(), x, y, size, size, null);  
 } else if (obj.getType() == FioType.TEXT) {  
 g2d.setFont(FioObject.TEXT\_FONT);  
 g2d.setColor(obj.getTextColor());  
 g2d.drawString(obj.getText(), x, y);  
 }  
 Dimension d = obj.getDimension();  
 String text = "[ id: " + obj.getId() + " ]";  
  
 TextLayout textLayout = new TextLayout(text, idTextFont, g2d.getFontRenderContext());  
 g2d.setPaint(Color.BLACK);  
 textLayout.draw(g2d, x + (int)d.getWidth() - 1, y + (int)d.getHeight() - 1);  
 textLayout.draw(g2d, x + (int)d.getWidth(), y + (int)d.getHeight() - 1);  
 textLayout.draw(g2d, x + (int)d.getWidth() + 1, y + (int)d.getHeight() + 1);  
 textLayout.draw(g2d, x + (int)d.getWidth() - 1, y + (int)d.getHeight() + 1);  
 textLayout.draw(g2d, x + (int)d.getWidth() + 1, y + (int)d.getHeight() - 1);  
  
 g2d.setPaint(Color.WHITE);  
 textLayout.draw(g2d, x + (int)d.getWidth(), y + (int)d.getHeight());  
 }  
 }  
 }  
}

package ru.inkrot.kit.laba4;  
  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
import java.util.concurrent.CopyOnWriteArraySet;  
import java.util.concurrent.ThreadLocalRandom;  
  
public class FioObject {  
  
 public static int *MAX\_NUMBER\_OF\_OBJECTS* = 200;  
 public static int *IMAGE\_SIZE* = 80;  
 public static Font *TEXT\_FONT* = new Font("Tahoma", 1, 16);  
  
 private static CopyOnWriteArraySet<FioObject> *allObjects* = new CopyOnWriteArraySet<>();  
  
 private static double[] *speedMap* = { 0.2, 0.3, 0.5, 0.7, 1.0 };  
  
 public static CopyOnWriteArraySet<FioObject> getAllObjects() {  
 return *allObjects*;  
 }  
  
 private static void addObject(FioObject fioObject) {  
 if (*allObjects*.size() < *MAX\_NUMBER\_OF\_OBJECTS*)  
 *allObjects*.add(fioObject);  
 }  
  
 public static void addObject(String text, Color textColor, int speed) {  
 *addObject*(new FioObject(text, textColor, speed));  
 }  
  
 public static void addObject(BufferedImage image, int speed) {  
 *addObject*(new FioObject(image, speed));  
 }  
  
 public static boolean isFreeId(int id) {  
 for (FioObject obj : *allObjects*) {  
 if (obj.getId() == id) return false;  
 }  
 return true;  
 }  
  
 private FioType type;  
  
 private int id;  
  
 private int speed = 1;  
  
 private double x = 0, y = 0;  
 private double dx = 0, dy = 0;  
  
 private BufferedImage image;  
  
 private String text;  
 private Color textColor;  
  
 private FioObject(BufferedImage image, int speed) {  
 if (! isCorrectSpeed(speed)) return;  
 getNextId();  
 this.type = FioType.*PICTURE*;  
 this.image = image;  
 this.speed = speed;  
 setLocation(0, 0);  
 setDirection(randomAngle());  
 }  
  
 private FioObject(String text, Color textColor, int speed) {  
 if (! isCorrectSpeed(speed)) return;  
 getNextId();  
 this.type = FioType.*TEXT*;  
 this.text = text;  
 this.textColor = textColor;  
 this.speed = speed;  
 setLocation(1, 13);  
 setDirection(randomAngle());  
 }  
  
 private int randomAngle() {  
 return ThreadLocalRandom.*current*().nextInt(0, 90 + 1);  
 }  
  
 private boolean isCorrectSpeed(int speed) {  
 if (speed >= 1 && speed <= 5) return true;  
 return false;  
 }  
  
 private void getNextId() {  
 int firstFreeId = 1;  
 while (! *isFreeId*(firstFreeId)) {  
 firstFreeId++;  
 }  
 id = firstFreeId;  
 }  
  
 public void nextStep(int width, int height) {  
 Dimension d = getDimension();  
 if (y >= height - d.getHeight()) dy = -dy;  
 else if (x >= width - d.getWidth()) dx = -dx;  
 else if (y <= -1) dy = -dy;  
 else if (x <= -1) dx = -dx;  
 double s = *speedMap*[speed - 1];  
 x += dx \* s;  
 y += dy \* s;  
 }  
  
 */\*\*  
 \** ***@param*** *angle of direction in degrees  
 \*/* public void setDirection(double angle) {  
 double rad = angle \* Math.*PI* / 180;  
 dx = Math.*cos*(rad);  
 dy = Math.*sin*(rad);  
 }  
  
 public void setLocation(int x, int y) {  
 this.x = (double) x;  
 this.y = (double) y;  
 }  
  
 public Dimension getTextSize() {  
 BufferedImage b = new BufferedImage(1, 1, BufferedImage.*TYPE\_INT\_ARGB*);  
 Graphics g = b.getGraphics();  
 FontMetrics metrics = g.getFontMetrics(*TEXT\_FONT*);  
 int hgt = metrics.getHeight();  
 int adv = metrics.stringWidth(text);  
 return new Dimension(adv, hgt);  
 }  
  
 public Dimension getDimension() {  
 return type == FioType.*TEXT* ? getTextSize() : new Dimension(*IMAGE\_SIZE*, *IMAGE\_SIZE*);  
 }  
  
 public int getX() {  
 return (int) x;  
 }  
  
 public void setX(int x) {  
 this.x = (double) x;  
 }  
  
 public int getY() {  
 return (int) y;  
 }  
  
 public void setY(int y) {  
 this.y = (double) y;  
 }  
  
 public double getDx() {  
 return dx;  
 }  
  
 public void setDx(double dx) {  
 this.dx = dx;  
 }  
  
 public double getDy() {  
 return dy;  
 }  
  
 public void setDy(double dy) {  
 this.dy = dy;  
 }  
  
 public int getSpeed() {  
 return speed;  
 }  
  
 public void setSpeed(int speed) {  
 this.speed = speed;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *id new identifier of object  
 \** ***@return*** *error code or 0 - success,  
 \* codes: 1 - id is bussy;  
 \* 2 - id within wrong rangle (allowed: 1 - 5)  
 \*/* public int setId(int id) {  
 if (! *isFreeId*(id)) return 1;  
 if (! (id >= 1 && id <= *MAX\_NUMBER\_OF\_OBJECTS*)) return 2;  
 this.id = id;  
 return 0;  
 }  
  
 public FioType getType() {  
 return type;  
 }  
  
 public void setType(FioType type) {  
 this.type = type;  
 }  
  
 public BufferedImage getImage() {  
 return image;  
 }  
  
 public void setImage(BufferedImage image) {  
 this.image = image;  
 }  
  
 public String getText() {  
 return text;  
 }  
  
 public void setText(String text) {  
 this.text = text;  
 }  
  
 public Color getTextColor() {  
 return textColor;  
 }  
  
 public void setTextColor(Color textColor) {  
 this.textColor = textColor;  
 }  
}

package ru.inkrot.kit.laba4;  
  
enum FioType {  
  
 *PICTURE*("Картинка"),  
 *TEXT*("Надпись");  
  
 String type;  
  
 FioType(String type) {  
 this.type = type;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public static FioType fromString(String type) {  
 for (FioType t : FioType.*values*()) {  
 if (t.getType().equalsIgnoreCase(type)) {  
 return t;  
 }  
 }  
 throw new IllegalArgumentException("No constant with type " + type + " found");  
 }  
}

package ru.inkrot.kit.laba4;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Main {  
  
 private static Dimension *screenSize*;  
 public static int *SCREEN\_WIDTH*;  
 public static int *SCREEN\_HEIGHT*;  
  
 public static void main(String[] args) throws Exception {  
 UIManager.*setLookAndFeel*(UIManager.*getSystemLookAndFeelClassName*());  
 *screenSize* = Toolkit.*getDefaultToolkit*().getScreenSize();  
 *SCREEN\_WIDTH* = (int) *screenSize*.getWidth();  
 *SCREEN\_HEIGHT* = (int) *screenSize*.getHeight();  
 new ControlFrame();  
 }  
}

package ru.inkrot.kit.laba4;  
  
import java.awt.event.WindowEvent;  
  
public class WindowListener implements java.awt.event.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) {  
  
 }  
}

