Самченко ІТІНФ-20-1 В-21

У первом потоці треба виконувати обчислення інтегралу від деякої функції, а в іншому - інтегралу від іншої функції. Функції обчислення інтегралів повинні бути реалізовані у динамічній бібліотеці, яка підключається до основного додатку. Ці функції і будуть викликатися в потоках. Також в потоці неоходімо реалізувати і побудувати графіки обох функцій, за якими обчислювалися інтеграли. Результати обчислення обох інтегралів необхідно передати по локальній мережі по протоколу TCP / IP в окремий додаток і в ньому вивести на екран ці результати. Малювання графіків необхідно реалізувати з використанням функцій GDI +. Передачу даних можна організувати і на локальному комп'ютері через локальний IP адрес за будь-якого порту з використанням сокетів. У програмі необхідно передбачити можливість вибору кольору, товщини і текстури лінії, якої малюється графік. Ці параметри повинні записуватися в реєстр Windows, а при запуску програми читатися і застосовуватися при малюванні. При першому запуску програма повинна перевірити наявність запису в реєстрі, і якщо її немає, то видати повідомлення й потім записати значення за замовчуванням. Після першого запуску програма повинна поміщати іконку в Windows Tray і при натисканні комбінації клавіш Ctrl + S активувати головний додаток, а при натисненні Ctrl + B активувати другий додаток.

CalculatIntegral.java

package com.example.laba4;  
  
public class CalculatIntegral {  
 public static double integralKvadret(double a, double b, Function function) {  
 double area = 0;  
 double h = 0.01;  
 for(int i = 0; i < (b-a)/h; i++) {  
 area += h \* function.func(a+i\*h);  
 }  
 return area;  
 }  
 interface Function {  
 public double func (double x);  
 }  
}

Chart.java

package com.example.laba4;  
  
import lombok.Getter;  
import lombok.Setter;  
import java.awt.\*;  
import java.io.Serializable;  
  
@Setter  
@Getter  
  
public class Chart implements Serializable {  
 private String function;  
 private String texture;  
 private String thickness;  
 private Color color;  
 private double A;  
 private double B;  
  
 Chart (String function, String texture, String thickness, Color color, double A, double B) {  
 this.function = function;  
 this.texture = texture;  
 this.thickness = thickness;  
 this.color = color;  
 this.A = A;  
 this.B = B;  
 }  
}

ClientApplication.java

package com.example.laba4;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class ClientApplication extends Application {  
 @Override  
 public void start(Stage stage) throws Exception {  
 Parent root = FXMLLoader.*load*(getClass().getResource("client.fxml"));  
 Scene scene = new Scene(root,500, 600);  
 stage.setTitle("Параметры графика");  
 stage.setScene(scene);  
 stage.show();  
 }  
   
 public static void main(String[] args){ *launch*(); }  
}

ClientControler.java

package com.example.laba4;  
  
import java.io.\*;  
import java.net.InetAddress;  
import java.net.Socket;  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.ResourceBundle;  
import javafx.event.ActionEvent;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
import javafx.scene.paint.Color;  
  
public class ClientController {  
  
 private ArrayList<Chart> data = new ArrayList<Chart>();  
  
 private final static int *SERVER\_PORT* = 8071;  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private Button addButon;  
  
 @FXML  
 private ColorPicker chooseColor;  
  
 @FXML  
 private ComboBox<String> chooseFunction;  
  
 @FXML  
 private ComboBox<String> chooseTexture;  
  
 @FXML  
 private ComboBox<String> chooseThickness;  
  
 @FXML  
 private Button operButon;  
  
 @FXML  
 private TextField limitA;  
  
 @FXML  
 private TextField limitB;  
  
 @FXML  
 void paintGraphics(ActionEvent event) throws IOException {  
 if(!data.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Придуприждение");  
 alert.setHeaderText(null);  
 alert.setContentText("При выборе пуктира рационально\n" +  
 "использовать шрифт, который не\n" +  
 "привешает или равен 5, иначе пунктира \n" +  
 "не будет заметно. После отрисовки все\n" +  
 "внесённые данные будут очищенны.");  
 alert.showAndWait();  
  
 sendKeysInfo();  
  
 data.clear();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка");  
 alert.setHeaderText(null);  
 alert.setContentText("Добавте графики, которые нужно отрисовать.");  
 alert.showAndWait();  
 }  
 }  
  
 @FXML  
 void addGraphics(ActionEvent event) {  
 try {  
 if(data.size()<5) {  
 double a = Double.*parseDouble*(limitA.getText());  
 double b = Double.*parseDouble*(limitB.getText());  
  
 Color c = chooseColor.getValue();  
 java.awt.Color awtColor = new java.awt.Color((float) c.getRed(),  
 (float) c.getGreen(),  
 (float) c.getBlue(),  
 (float) c.getOpacity());  
  
 data.add(new Chart(chooseFunction.getSelectionModel().getSelectedItem().toString(),  
 chooseTexture.getSelectionModel().getSelectedItem().toString(),  
 chooseThickness.getSelectionModel().getSelectedItem().toString(),  
 awtColor, a, b));  
  
 System.*out*.println("Добавлено");  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка");  
 alert.setHeaderText(null);  
 alert.setContentText("В одновременную обработку можно отправить от 1 до 5 графиков.");  
 alert.showAndWait();  
 }  
 }  
 catch (NumberFormatException e) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка");  
 alert.setHeaderText(null);  
 alert.setContentText("Неверно введёные грницы вычисления интеграла.\n" +  
 "Граници надо задавать числами, а дробные значения через точку.");  
 alert.showAndWait();  
 }  
 }  
  
 private void sendKeysInfo() throws IOException {  
 Socket s = new Socket(InetAddress.*getLocalHost*(), *SERVER\_PORT*);  
 ObjectOutputStream out = new ObjectOutputStream(s.getOutputStream());  
 out.writeObject(data);  
 out.flush();  
 s.close();  
 }  
  
 @FXML  
 void initialize() {  
 ObservableList<String> list1 = FXCollections.*observableArrayList*("y=x^2", "y=sin(x)");  
 chooseFunction.setItems(list1);  
  
 ObservableList<String> list2 = FXCollections.*observableArrayList*("сплошная", "пунктир");  
 chooseTexture.setItems(list2);  
  
 ObservableList<String> list3 = FXCollections.*observableArrayList*("1", "2", "3", "4",  
 "5", "6", "7", "8", "9", "10", "11", "12", "14", "15", "16", "17", "18", "19", "20");  
 chooseThickness.setItems(list3);  
 }  
}

ServerApplication.java

package com.example.laba4;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
  
public class ServerApplication {  
 private static final int *SERVER\_PORT* = 8071;  
  
 ServerApplication() throws IOException {  
 System.*out*.println("Server is started");  
  
 ServerSocket serv = new ServerSocket(*SERVER\_PORT*);  
 while (true) {  
 Socket sock = serv.accept();  
 processRequest(sock);  
 sock.close();  
 }  
 }  
  
 private void processRequest(Socket sock) throws IOException {  
 ObjectInputStream in = new ObjectInputStream(sock.getInputStream());  
  
 ArrayList<Chart> inData= new ArrayList<Chart>();  
 try {  
 inData=((ArrayList<Chart>)in.readObject());  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 sock.close();  
  
 for(int i = 0; i < inData.size(); i++)  
 new ServerFrame(inData.get(i));  
  
 System.*out*.println("Сработало");  
 }  
  
 public static void main(String[] args) throws IOException { new ServerApplication(); }  
}

ServerFrame.java

package com.example.laba4;  
  
import org.jfree.chart.ChartFactory;  
import org.jfree.chart.ChartPanel;  
import org.jfree.chart.JFreeChart;  
import org.jfree.chart.plot.PlotOrientation;  
import org.jfree.chart.plot.XYPlot;  
import org.jfree.data.xy.XYDataset;  
import org.jfree.data.xy.XYSeries;  
import org.jfree.data.xy.XYSeriesCollection;  
import javax.swing.\*;  
import java.awt.\*;  
  
public class ServerFrame extends JFrame {  
 private Chart ch;  
  
 ServerFrame(Chart ch) {  
 this.ch = ch;  
  
 Thread th = new Thread(new ChartBuilder());  
 th.start();  
 }  
  
 private class ChartBuilder implements Runnable {  
 @Override  
 public void run() {  
 XYSeries series = new XYSeries(ch.getFunction());  
  
 for (double i = -20; i < 20; i += 0.1) {  
 if(ch.getFunction().equals("y=sin(x)"))  
 series.add(i, Math.*sin*(i));  
  
 if (ch.getFunction().equals("y=x^2"))  
 series.add(i, Math.*pow*(i,2));  
 }  
  
 String res = " ";  
 if(ch.getFunction().equals("y=sin(x)")) {  
 res = Double.*toString*(CalculatIntegral.*integralKvadret*(ch.getA(), ch.getB(), x -> (Math.*sin*(x))));  
 }  
 if(ch.getFunction().equals("y=x^2")) {  
 res = Double.*toString*(CalculatIntegral.*integralKvadret*(ch.getA(), ch.getB(), x -> (Math.*pow*(x, 2))));  
 }  
  
 XYDataset xyDataset = new XYSeriesCollection(series);  
 JFreeChart chart = ChartFactory  
 .*createXYLineChart*(/\*ch.getFunction()\*/"Значение интеграла в диапазоне от А = " + ch.getA() +  
 " до В = " + ch.getB() + " равна: " + res,  
 "x", "y",  
 xyDataset,  
 PlotOrientation.*VERTICAL*,  
 true, true, true);  
  
 XYPlot plot = (XYPlot) chart.getPlot();  
 plot.getRenderer().setSeriesPaint(0, ch.getColor());  
  
 float f = Float.*parseFloat*(ch.getThickness());  
 plot.getRenderer().setSeriesStroke( 0 , new BasicStroke(f));  
  
 if(ch.getTexture().equals("пунктир")) {  
 plot.getRenderer().setSeriesStroke(  
 0,  
 new BasicStroke(  
 f, BasicStroke.*CAP\_ROUND*, BasicStroke.*CAP\_BUTT*,  
 1.0f, new float[]{5.0f, 5.0f}, 0.0f  
 ));  
  
 }  
  
 JFrame frame =  
 new JFrame("График");  
 frame.getContentPane()  
 .add(new ChartPanel(chart));  
 frame.setSize(400, 300);  
 frame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);  
 frame.show();  
 }  
 }  
}

Результат:

Изображение выглядит как текст, снимок экрана, монитор, электроника

Автоматически созданное описание