МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Звіт з лабораторної роботи № 4

з предмету «Об'єктно-орієнтоване програмування»

Виконав:

Студент групи КН-36а

Кулик В.В.

Перевірили:

Козуля М.М.

Кізілов О.С.

Харків 2017

### Завдання на лабораторну роботу

### 1.1 Індивідуальне завдання

Створити програму графічного інтерфейсу користувача, яка призначена для побудови графіку довільних функцій. Користувач повинен увести дійсні значення *a* і *b*, функції *f(x)* і *g(x)* у вигляді рядків, які відповідають синтаксису Java. У програмі здійснюється обчислення функції *h(x)* відповідно до індивідуального завдання:

Таблиця 1.1 – Індивідуальні завдання

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номери варіантів | | Функція *h(x)* | Номери варіантів | | Функція *h(x)* |
| 1 | 13 | *a*∙*f(x) – b∙g(x)* | 7 | 19 | *f(a + x) + b∙g(x)* |
| 2 | 14 | *f(x + a) + g(x – b)* | 8 | 20 | *f(a / x) – g(b∙x)* |
| 3 | 15 | *(a-f(x))(b + g(x))* | 9 | 21 | *f(x – a)∙g(x + b)* |
| 4 | 16 | *f(a∙x) – g(b∙x)* | 10 | 22 | *f(a / x) + g(b / x)* |
| 5 | 17 | *f(x / a)∙g(x + b)* | 11 | 23 | *a∙f(x)* + *b∙g(x)* |
| 6 | 18 | *f(a / x) – g(b / x)* | 12 | 24 | *a*∙*f(x)* – *g(b∙x)* |

Після введення необхідних функцій, діапазону відображення графіку і натиснення відповідної кнопки здійснюється побудова графіку. Слід також передбачити функцію очищення рядків уведення й графіку.

Для програмної реалізації обробки введених виразів слід застосувати динамічну компіляцію коду. Для створення застосунку графічного інтерфейсу користувача слід використати засоби JavaFX. Рекомендований підхід - використання компоненту LineChart.

### 1.2 Перегляд всіх полів класу

Створити консольний застосунок, в якому користувач вводить ім'я класу і отримує інформацію про всі поля цього класу (включаючи закриті і захищені).

### 1.3 Створення застосунку графічного інтерфейсу користувача для отримання простих множників чисел

### За допомогою засобів JavaFX розробити застосунок графічного інтерфейсу користувача, в якому користувач вводить діапазон чисел (від і до), а у вікні відображаються числа і їх прості множники. Реалізувати можливість призупинення, відновлення потоку, а також повного припинення і повторного обчислення з новими даними.

### 1.4 Робота з BlockingQueue

### Створити консольну програму, в якій один потік виконання додає цілі числа до черги BlockingQueue,а інший обчислює їх середнє арифметичне.

### 1.5 Виклик функції для обраного класу (додаткове завдання)

Створити класи з однойменними методами. Вибрати клас за ім'ям і викликати його метод.

### 1.6 Інтерпретація математичних виразів (додаткове завдання)

Створити консольний застосунок, який дозволяє вводити математичні вирази, обчислювати і виводити результат. Вираз може складатися з констант, математичних операцій і дужок. Для реалізації використовувати засоби пакету javax.script.

*Примітка*. Синтаксис математичних виразів JavaScript аналогічний Java. Результат можна виводити за допомогою функції print() без створення додаткових змінних.

### 1.7 Обчислення π в окремому потоці виконання (додаткове завдання)

Реалізувати програму обчислення π с точністю до заданого ε як суму послідовності:

http://iwanoff.96.lt/oop_kn/Images/03_pi_1.png

Обчислення здійснювати в окремому потоці виконання. Під час виконання обчислення надавати користувачеві можливість уводити запит про кількість обчислених елементів суми.

### 1.8 Робота з потоками даних (додаткове завдання)

Створити консольну програму, в якій виводяться всі додатні цілі числа, сума цифр яких дорівнює заданому значенню. Використати потоки даних.

**Хід роботи**

### 1.1 Індивідуальне завдання

*Код програми 1.1:*

**Main.java**

**import javafx.application.Application;**

**import javafx.fxml.FXMLLoader;**

**import javafx.scene.Scene;**

**import javafx.scene.layout.AnchorPane;**

**import javafx.stage.Stage;**

**import java.io.IOException;**

**public class Main extends Application {**

**private Stage primaryStage;**

**private AnchorPane root;**

**public Stage getPrimaryStage() {**

**return primaryStage;**

**}**

**@Override**

**public void start(Stage primaryStage) throws Exception{**

**this.primaryStage = primaryStage;**

**primaryStage.setTitle("Поиск точки пересечения");**

**initRootLayout();**

**}**

**public void initRootLayout() {**

**try {**

**FXMLLoader loader = new FXMLLoader();**

**loader.setLocation(Main.class.getResource("sample.fxml"));**

**root = (AnchorPane) loader.load();**

**Scene scene = new Scene(root);**

**primaryStage.setScene(scene);**

**Controller controller = loader.getController();**

**controller.setMain(this);**

**primaryStage.show();**

**} catch (IOException e) {**

**System.out.println(e.getMessage());**

**}**

**}**

**public static void main(String[] args) {**

**launch(args);**

**}**

**}**

**Poly.java**

**import org.mariuszgromada.math.mxparser.Argument;**

**import org.mariuszgromada.math.mxparser.Expression;**

**import java.util.function.Function;**

**public class Poly implements Function<Double, Double> {**

**Argument argX = new Argument("x");**

**Expression y;**

**@Override**

**public Double apply(Double x) {**

**argX.setArgumentValue(x);**

**return y.calculate();**

**}**

**public Poly(String fun) {**

**fun = fun.replaceAll(" ", "");**

**y = new Expression(fun, argX);**

**}**

**}**

**Controller.java**

**import javafx.fxml.FXML;**

**import javafx.scene.chart.LineChart;**

**import javafx.scene.chart.XYChart;**

**import javafx.scene.control.TextField;**

**import javafx.scene.layout.StackPane;**

**import java.util.function.Function;**

**public class Controller {**

**Main main;**

**@FXML**

**private TextField af;**

**@FXML**

**private TextField bf;**

**@FXML**

**private TextField ff;**

**@FXML**

**private TextField gf;**

**@FXML**

**private LineChart<Number,Number> chart;**

**public Controller() {**

**}**

**public void setMain(Main main) {**

**this.main = main;**

**}**

**@FXML**

**private void handleDraw() {**

**Function<Double,Double> f = new Poly(ff.getText());**

**Function<Double,Double> g = new Poly(gf.getText());**

**chart.getData().clear();**

**double a = Double.parseDouble(af.getText());**

**double b = Double.parseDouble(bf.getText());**

**XYChart.Series poly = new XYChart.Series<>();**

**Function<Double,Double> h = (Double x) -> {return a\*f.apply(x) - b\*g.apply(x);};**

**for (double j = 0.1; j <= 50; j += 0.1)**

**poly.getData().add(new XYChart.Data<Number, Number>(j, h.apply(j)));**

**chart.getData().add(poly);**

**for (XYChart.Data<Number, Number> data : chart.getData().get(0).getData()) {**

**StackPane stackPane = (StackPane) data.getNode();**

**stackPane.setDisable(true);**

**stackPane.setVisible(false);**

**}**

**}**

**}**

**FXClass.java**

**package laba4;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import java.lang.reflect.Method;**

**import javax.tools.JavaCompiler;**

**import javax.tools.ToolProvider;**

**import javafx.application.Application;**

**import javafx.beans.value.ChangeListener;**

**import javafx.beans.value.ObservableValue;**

**import javafx.event.Event;**

**import javafx.fxml.FXML;**

**import javafx.scene.Scene;**

**import javafx.scene.chart.CategoryAxis;**

**import javafx.scene.chart.LineChart;**

**import javafx.scene.chart.NumberAxis;**

**import javafx.scene.chart.XYChart;**

**import javafx.scene.chart.XYChart.Data;**

**import javafx.scene.control.Alert;**

**import javafx.scene.control.Button;**

**import javafx.scene.control.Label;**

**import javafx.scene.control.TextField;**

**import javafx.scene.control.Alert.AlertType;**

**import javafx.scene.layout.AnchorPane;**

**import javafx.scene.layout.BorderPane;**

**import javafx.scene.layout.Pane;**

**import javafx.stage.Stage;**

**public class FXClass extends Application {**

**AnchorPane root = new AnchorPane();**

**TextField aValue = new TextField();**

**TextField bValue = new TextField();**

**TextField fFunc = new TextField();**

**TextField gFunc = new TextField();**

**TextField from = new TextField("from");**

**TextField to = new TextField("to");**

**Label a = new Label("A");**

**Label b = new Label("B");**

**Label f = new Label("f(x)");**

**Label g = new Label("g(x)");**

**Button clear = new Button("clear");**

**Button build = new Button("build");**

**NumberAxis xAxis = new NumberAxis();**

**NumberAxis yAxis = new NumberAxis();**

**private LineChart<Number, Number> lineChart = new LineChart(xAxis,yAxis);**

**private final String PATH = "C://Users//lena//workspace//laba4//bin//FGFunc.java";**

**private final String BUILD = "build";**

**@Override**

**public void start(Stage primaryStage) throws Exception {**

**AnchorPane.setLeftAnchor(aValue,54.0);**

**AnchorPane.setTopAnchor(aValue,32.0);**

**aValue.setPrefHeight(25.0);**

**aValue.setPrefWidth(48.0);**

**AnchorPane.setLeftAnchor(bValue,144.0);**

**AnchorPane.setTopAnchor(bValue,32.0);**

**bValue.setPrefHeight(25.0);**

**bValue.setPrefWidth(48.0);**

**AnchorPane.setLeftAnchor(fFunc,51.0);**

**AnchorPane.setTopAnchor(fFunc,76.0);**

**fFunc.setMinHeight(25.0);**

**fFunc.setMinWidth(141.0);**

**AnchorPane.setLeftAnchor(gFunc,51.0);**

**AnchorPane.setTopAnchor(gFunc,113.0);**

**gFunc.setMinHeight(25.0);**

**gFunc.setMinWidth(141.0);**

**AnchorPane.setLeftAnchor(a,31.0);**

**AnchorPane.setTopAnchor(a,36.0);**

**AnchorPane.setLeftAnchor(b,118.0);**

**AnchorPane.setTopAnchor(b,36.0);**

**AnchorPane.setLeftAnchor(f,17.0);**

**AnchorPane.setTopAnchor(f,80.0);**

**AnchorPane.setLeftAnchor(g,17.0);**

**AnchorPane.setTopAnchor(g,117.0);**

**AnchorPane.setLeftAnchor(lineChart,194.0);**

**AnchorPane.setLeftAnchor(clear,18.0);**

**AnchorPane.setTopAnchor(clear,182.0);**

**clear.setMnemonicParsing(false);**

**clear.setOnAction(this::onClear);**

**clear.setPrefWidth(176.0);**

**clear.setPrefHeight(25);**

**AnchorPane.setLeftAnchor(build,18.0);**

**AnchorPane.setTopAnchor(build,148.0);**

**build.setMnemonicParsing(false);**

**build.setOnAction(this::onBuild);**

**build.setPrefWidth(176.0);**

**build.setPrefHeight(25);**

**AnchorPane.setLeftAnchor(from,27.0);**

**AnchorPane.setTopAnchor(from,220.0);**

**from.setPrefWidth(76.0);**

**from.setPrefHeight(25);**

**AnchorPane.setLeftAnchor(to ,120.0);**

**AnchorPane.setTopAnchor(to,220.0);**

**to.setPrefWidth(76.0);**

**to.setPrefHeight(25);**

**lineChart.setTitle("f(a / x) – g(b∙x)");**

**root.getChildren().addAll(aValue,bValue,fFunc, gFunc, from, to, a,b,f,g,clear,build,lineChart);**

**Scene scene = new Scene(root, 800, 400);**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**}**

**public static void main(String[] args) {**

**launch(args);**

**}**

**private void onClear(Event event) {**

**aValue.clear();**

**bValue.clear();**

**fFunc.clear();**

**gFunc.clear();**

**lineChart.getData().clear();**

**}**

**private void onBuild(Event event) {**

**lineChart.getData().clear();**

**if (checkFields().equals(BUILD)) {**

**genSource();**

**XYChart.Series<Number, Number> series = solve();**

**lineChart.getData().add(series);**

**} else {**

**showError(checkFields());**

**}**

**}**

**private XYChart.Series<Number, Number> solve() {**

**double f = Double.parseDouble(aValue.getText());**

**double u = Double.parseDouble(bValue.getText());**

**Double from1=Double.parseDouble(from.getText());**

**Double to1=Double.parseDouble(to.getText());**

**XYChart.Series<Number, Number> series = new XYChart.Series<>();**

**try {**

**if (compile()) {**

**MyLoader test = new MyLoader();**

**Class<?> fClass = test.loadFG();**

**Method fMethod = fClass.getMethod("getFY", double.class);**

**Method gMethod = fClass.getMethod("getGY", double.class);**

**for (double i = from1; i <= to1; i += 0.1) {**

**double fy = (double) fMethod.invoke(null, new Object[] { (f / i) });**

**double gy = (double) gMethod.invoke(null, new Object[] { (u \* i) });**

**double y = fy - gy;**

**if (Math.abs(y) > 50 && series.getData().size() > 0) {**

**y = (double) series.getData().get(series.getData().size() - 1).getYValue() +5;**

**}**

**series.getData().add(new Data<>(i, y));**

**}**

**}else {**

**throw new Exception();**

**}**

**} catch (Exception e) {**

**showError("invalid function");**

**e.printStackTrace();**

**}**

**return series;**

**}**

**private boolean compile() {**

**JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();**

**return compiler.run(null, null, null, PATH) == 0;**

**}**

**private void genSource() {**

**try (PrintWriter out = new PrintWriter(PATH)) {**

**out.println("");**

**out.println("import java.lang.Math;\n");**

**out.println("public class FGFunc {");**

**out.println(" public static double getFY(double x) {");**

**out.println(" return " + fFunc.getText().toString() + ";");**

**out.println(" }");**

**out.println(" public static double getGY(double x) {");**

**out.println(" return " + gFunc.getText().toString() + ";");**

**out.println(" }");**

**out.println("}");**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**private String checkFields() {**

**if (aValue.getText().length() == 0 || bValue.getText().length() == 0 || fFunc.getText().length() == 0**

**|| gFunc.getText().length() == 0) {**

**return "Fill all fields";**

**}**

**Double i1 = Double.parseDouble(from.getText());**

**Double i2 = Double.parseDouble(to.getText());**

**return i1 < i2 ? BUILD : "Check interval";**

**}**

**private void showError(String message) {**

**Alert alert = new Alert(AlertType.ERROR);**

**alert.titleProperty().set("ERROR");**

**alert.setHeaderText(message);**

**alert.show();**

**}**

**@FXML**

**private void initialize() {**

**setListeners();**

**lineChart.setTitle("f(a / x) – g(b∙x)");**

**}**

**private void setListeners() {**

**aValue.textProperty().addListener(new FieldListener(aValue));**

**bValue.textProperty().addListener(new FieldListener(bValue));**

**from.textProperty().addListener(new FieldListener(from));**

**to.textProperty().addListener(new FieldListener(to));**

**}**

**private boolean isNumber(String str) {**

**if (str == null || str.isEmpty())**

**return false;**

**if (str.contains(",")) {**

**return false;**

**}**

**int count = 0;**

**for (int i = 0; i < str.length(); i++) {**

**if (Character.compare(str.charAt(i), '.') == 0) {**

**count++;**

**}**

**}**

**if (count > 1) {**

**return false;**

**}**

**for (int i = 0; i < str.length(); i++) {**

**if (!Character.isDigit(str.charAt(i)) && !(i == 0 && str.charAt(i) == '-') && str.charAt(i) != '.')**

**return false;**

**}**

**return true;**

**}**

**class FieldListener implements ChangeListener<String> {**

**private TextField field;**

**public FieldListener(TextField field) {**

**this.field = field;**

**}**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**if (!isNumber(newValue) || newValue.isEmpty()) {**

**field.setText(newValue.length() > 1 ? newValue.substring(0, newValue.length() - 1) : "");**

**}**

**}**

**}**

**}**

**MyLoader.java**

**package laba4;**

**import java.io.ByteArrayOutputStream;**

**import java.io.FileInputStream;**

**import java.io.InputStream;**

**public class MyLoader extends ClassLoader {**

**public MyLoader() {**

**}**

**public Class loadFG() {**

**try {**

**InputStream input = new FileInputStream("C://Users//lena//workspace//laba4//bin//FGFunc.class");**

**ByteArrayOutputStream buffer = new ByteArrayOutputStream();**

**int data = input.read();**

**while (data != -1) {**

**buffer.write(data);**

**data = input.read();**

**}**

**input.close();**

**byte[] classData = buffer.toByteArray();**

**return defineClass("FGFunc", classData, 0, classData.length);**

**} catch (Exception e) {**

**System.out.println("error");**

**e.printStackTrace();**

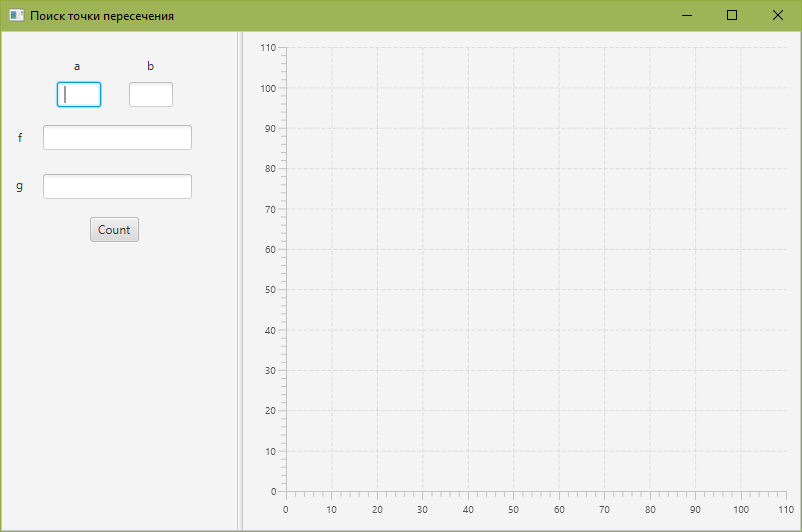
**}**

**return null;**

**}**

**}**

*Після запуску програми відкривається наступне вікно:*



### 1.2 Перегляд всіх полів класу

*Код програми 1.2:*

**ShowAllMethods.java**

**import java.util.Scanner;**

**public class ShowAllMethods {**

**@SuppressWarnings("resource")**

**public static void main(String[] args) {**

**System.out.println("Please, enter a name of class: ");**

**String nameOfClass = new Scanner(System.in).next();**

**try{**

**Class<?> c = Class.forName(nameOfClass);**

**for(int i=0;i<c.getDeclaredFields().length;i++) {**

**c.getDeclaredFields()[i].setAccessible(true);**

**System.out.println("Name: "+c.getDeclaredFields()[1].getName()+" Type: "+c.getDeclaredFields()[i].getType());**

**if(c.getDeclaredFields()[i].getModifiers()==1)**

**System.out.println("Modifiers: public");**

**if(c.getDeclaredFields()[i].getModifiers()==2)**

**System.out.println("Modifiers: private");**

**if(c.getDeclaredFields()[i].getModifiers()==3)**

**System.out.println("Modifiers: protected");**

**}**

**}**

**catch(ClassNotFoundException e){**

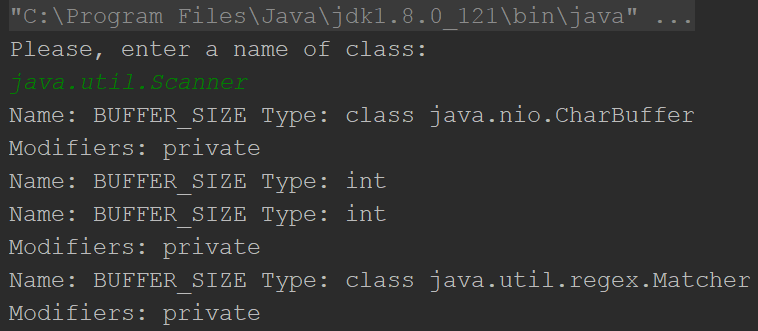
**System.err.println("Wrong the name of class");**

**}**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.3 Створення застосунку графічного інтерфейсу користувача для отримання простих множників чисел

*Код програми 1.3:*

**Main.java**

**package fourthlab.thirdex;**

**import java.lang.Thread.State;**

**import java.util.Observable;**

**import com.sun.media.jfxmedia.events.NewFrameEvent;**

**import javafx.application.Application;**

**import javafx.beans.value.ChangeListener;**

**import javafx.beans.value.ObservableValue;**

**import javafx.geometry.Insets;**

**import javafx.scene.Scene;**

**import javafx.scene.control.Button;**

**import javafx.scene.control.Label;**

**import javafx.scene.control.TextArea;**

**import javafx.scene.control.TextField;**

**import javafx.scene.layout.BorderPane;**

**import javafx.scene.layout.HBox;**

**import javafx.scene.layout.VBox;**

**import javafx.stage.Stage;**

**public class Main extends Application {**

**private TextField from;**

**private TextField until;**

**private TextArea result;**

**private BorderPane root;**

**private Button clear;**

**private Button stop;**

**private Button pause;**

**private Button resume;**

**private SimpleThread thread;**

**public static void main(String[] args) {**

**launch(args);**

**}**

**@Override**

**public void start(Stage primaryStage) throws Exception {**

**initFields();**

**setFieldsListeners();**

**setButtonsListeners();**

**thread = new SimpleThread();**

**thread.setDaemon(true);**

**Label l=new Label("Интервал ");**

**Label fromm=new Label("от ");**

**Label to=new Label(" до ");Label l1=new Label(" ");Label l2=new Label(" ");Label l3=new Label(" ");**

**HBox box = new HBox(l,fromm,from,to,until);**

**HBox box2 = new HBox(clear,l1, stop,l2, pause,l3, resume);**

**HBox box3=new HBox(result);**

**box.setPadding(new Insets(7, 7, 7, 7));**

**box2.setPadding(new Insets(7, 7, 7, 7));**

**box2.setPadding(new Insets(10, 10, 10, 10));**

**root.setCenter(box2);**

**root.setTop(box);**

**root.setBottom(box3);**

**Scene scene = new Scene(root, 450, 250);**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**}**

**private void setFieldsListeners() {**

**class Listen implements ChangeListener<String> {**

**private TextField field;**

**public Listen(TextField field) {**

**this.field = field;**

**}**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**thread.interrupt();**

**if (!isNumber(newValue) || newValue.isEmpty() || newValue.contains(",") || newValue.contains(".")) {**

**field.setText(newValue.length() > 1 ? newValue.substring(0, newValue.length() - 1) : "");**

**} else {**

**if (checkInterval()) {**

**thread = new SimpleThread();**

**thread.setDaemon(true);**

**thread.setF(Integer.parseInt(from.getText()));**

**thread.setU(Integer.parseInt(until.getText()));**

**thread.start();**

**} else {**

**result.setText("invalid interval");**

**}**

**}**

**}**

**}**

**from.textProperty().addListener(new Listen(from));**

**until.textProperty().addListener(new Listen(until));**

**}**

**private void setButtonsListeners() {**

**pause.setOnAction(t -> thread.pauseT());**

**stop.setOnAction(t -> thread.interrupt());**

**resume.setOnAction(t -> thread.resumeT());**

**clear.setOnAction(t -> {**

**from.clear();**

**until.clear();**

**result.clear();**

**thread.interrupt();**

**});**

**}**

**private boolean checkInterval() {**

**if (from.getText().length() == 0 || until.getText().length() == 0) {**

**return false;**

**}**

**Integer i1 = Integer.parseInt(from.getText());**

**if (i1 == 0) {**

**return false;**

**}**

**Integer i2 = Integer.parseInt(until.getText());**

**switch (i1.compareTo(i2)) {**

**case 1:**

**case 0:**

**return false;**

**case -1:**

**return true;**

**}**

**return false;**

**}**

**private void initFields() {**

**root = new BorderPane();**

**from = new TextField();**

**until = new TextField();**

**result = new TextArea();**

**clear = new Button("Очистить");**

**stop = new Button("Остановить");**

**resume = new Button("Возобновить");**

**pause = new Button("Пауза");**

**}**

**private boolean isNumber(String str) {**

**if (str == null || str.isEmpty())**

**return false;**

**if (str.charAt(0) == ',' || str.charAt(0) == '.' || str.contains("-") || str.contains(".") || str.contains(","))**

**return false;**

**for (int i = 0; i < str.length(); i++) {**

**if (!Character.isDigit(str.charAt(i)) & str.charAt(i) != '.' & str.charAt(i) != '-' & str.charAt(i) != ',')**

**return false;**

**}**

**return true;**

**}**

**class SimpleThread extends Thread {**

**private int f;**

**private int u;**

**private StringBuffer sBuffer;**

**private Boolean controller = false;**

**public SimpleThread() {**

**sBuffer = new StringBuffer();**

**}**

**public SimpleThread(int f, int u) {**

**this.f = f;**

**this.u = u;**

**sBuffer = new StringBuffer();**

**}**

**public void setF(int f) {**

**this.f = f;**

**clearSbuffer();**

**}**

**public void setU(int u) {**

**this.u = u;**

**clearSbuffer();**

**}**

**private void clearSbuffer() {**

**if (sBuffer.toString().length() > 0) {**

**sBuffer.delete(0, sBuffer.length() - 1);**

**}**

**}**

**public synchronized void resumeT() {**

**controller = false;**

**notify();**

**}**

**public void pauseT() {**

**controller = true;**

**}**

**@Override**

**public void run() {**

**int value = 0;**

**int temp = 0;**

**for (; f <= u; f++) {**

**temp = f;**

**String string = "";**

**while (temp != 1 || string.isEmpty()) {**

**synchronized (this) {**

**while (controller) {**

**try {**

**wait();**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**if (temp % 2 == 0 && temp / 2 != 1) {**

**value = 2;**

**} else if (temp % 3 == 0 && temp / 3 != 1) {**

**value = 3;**

**} else {**

**if (!string.isEmpty()) {**

**string += temp + " ";**

**} else**

**string += "Простой множитель ";**

**break;**

**}**

**temp /= value;**

**string += value + " \* ";**

**}**

**try {**

**Thread.sleep(500);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**f = u + 1;**

**break;**

**}**

**string = string.substring(0, string.length() - 2);**

**sBuffer.append(f + " = " + string + "\n");**

**result.setText(sBuffer.toString());**

**result.setScrollTop(Double.MAX\_VALUE);**

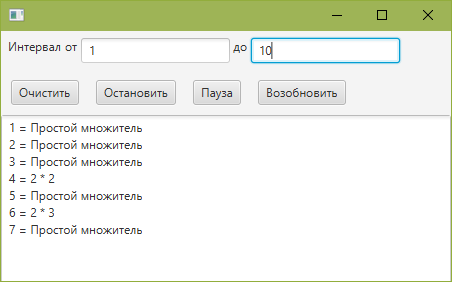
**}**

**}**

**}**

**}**

*Після запуску програми відкривається наступне вікно:*



### 1.4  Робота з BlockingQueue

*Код програми 1.4:*

**BlockingQueueDemo.java**

**import java.util.Scanner;**

**import java.util.concurrent.ArrayBlockingQueue;**

**import java.util.concurrent.BlockingQueue;**

**class Producer implements Runnable {**

**private BlockingQueue<Integer> queue;**

**int countToAdd;**

**public Producer(BlockingQueue<Integer> queue, int countToAdd) {**

**this.queue = queue;**

**this.countToAdd = countToAdd;**

**}**

**public void run() {**

**// Добавление чисел:**

**try {**

**for (int i = 1; i <= countToAdd; i++) {**

**queue.put(i);**

**System.out.printf("Added: %d%n", i);**

**Thread.sleep(100);**

**}**

**}**

**catch (InterruptedException e) {**

**System.out.println("Producer interrupted");**

**}**

**}**

**}**

**class Consumer implements Runnable {**

**private BlockingQueue<Integer> queue;**

**int countToTake;**

**public Consumer(BlockingQueue<Integer> queue, int countToTake) {**

**this.queue = queue;**

**this.countToTake = countToTake;**

**}**

**public void run() {**

**// Вычитание чисел:**

**double j=0;**

**try {**

**for (double i = 1; i <= countToTake; i++) {**

**j+=(double)queue.take();**

**double d = j/i;**

**System.out.println("Average:"+d);**

**}**

**}**

**catch (InterruptedException e) {**

**System.out.println("Consumer interrupted");**

**}**

**}**

**}**

**public class BlockingQueueDemo {**

**public static void main(String[] args) throws InterruptedException {**

**System.out.println("Enter the value elements of the line");**

**Scanner in = new Scanner(System.in);**

**int count = in.nextInt();**

**BlockingQueue<Integer> queue = new ArrayBlockingQueue<>(count);**

**// Создаем два потока - для записи и чтения:**

**Thread producerThread = new Thread(new Producer(queue, count));**

**Thread consumerThread = new Thread(new Consumer(queue, count));**

**producerThread.start();**

**consumerThread.start();**

**// Ждем 10 секунд и прерываем 1 поток:**

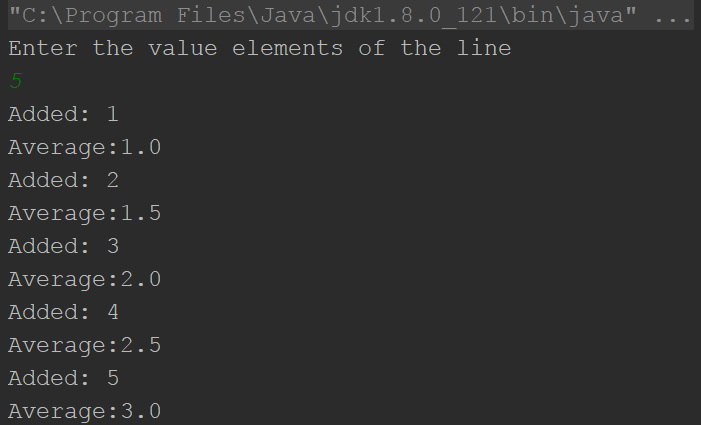
**Thread.sleep(10000);**

**producerThread.interrupt();**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.5 Виклик функції для обраного класу (додаткове завдання)

*Код програми 1.5:*

**CallOnSomeFunction.java**

**package lab3\_3;**

**import java.util.Scanner;**

**class A{**

**public void a(){System.out.println("It is class \"A\" ");}**

**}**

**class B{**

**public void b(){System.out.println("It is class \"B\" ");}**

**}**

**class C{**

**public void c(){System.out.println("It is class \"C\" ");}**

**}**

**public class CallOnSomeFunction {**

**public static void main(String[] args) {**

**Object o;**

**System.out.println("Enter a number of some class: ");**

**int i = new Scanner(System.in).nextInt();**

**switch (i){**

**case 1 :**

**o = new A();**

**break;**

**case 2 :**

**o = new B();**

**break;**

**case 3 :**

**o = new C();**

**break;**

**default:**

**System.out.println("This number of class is not correct ");**

**return;**

**}**

**if(o instanceof A)**

**((A) o).a();**

**if(o instanceof B)**

**((B) o).b();**

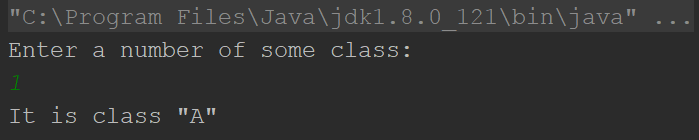
**if(o instanceof C)**

**((C) o).c();**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.6 Інтерпретація математичних виразів (додаткове завдання)

*Код програми 1.6:*

**MathSolution.java**

**package lab3\_4;**

**import javax.script.ScriptEngine;**

**import javax.script.ScriptEngineManager;**

**import java.util.Scanner;**

**public class MathSolution {**

**private static String[] func1Names = new String[] {"abs", "acos", "asin", "atan", "ceil", "cos", "exp", "floor", "log", "random",**

**"round", "sin", "sqrt", "tan"};**

**private static String[] func2Names = new String[] {"atan2", "max", "min", "pow"};**

**public static void main(String[] args) throws Exception {**

**ScriptEngineManager factory = new ScriptEngineManager();**

**ScriptEngine engine = factory.getEngineByName("JavaScript");**

**engine.put("pi", Math.PI);**

**engine.put("e", Math.E);**

**for( String func : func1Names )**

**engine.eval(String.format("function %1$s(x){return Math.%1$s(x);}", func));**

**for( String func : func2Names )**

**engine.eval(String.format("function %1$s(x,y){return Math.%1$s(x,y);}", func));**

**System.out.println("Enter a mathematical expression: ");**

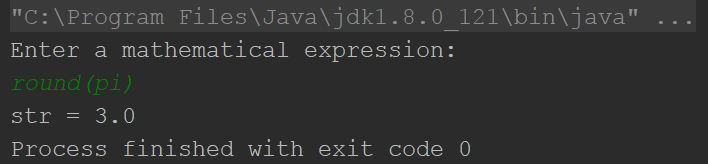
**String str = new Scanner(System.in).next();**

**System.out.print("str = "+engine.eval(str));**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.7 Обчислення π в окремому потоці виконання (додаткове завдання)

*Код програми 1.7:*

**Test.java**

**package lab5\_pi;**

**import java.util.Scanner;**

**public class Test implements Runnable{**

**@Override**

**public void run() {**

**double eps;**

**Scanner in = new Scanner(System.in);**

**System.out.println("Enter epsilon: ");**

**eps = in.nextDouble();**

**double p=0;**

**double a=4;**

**double b=4/3;**

**int check;**

**int count=2;**

**for (double i=5;a-b>eps;i+=2){**

**p+=a-b;**

**a=4/i;**

**i+=2;**

**b=4/i;**

**count+=2;**

**System.out.println("Вы хотите узнать о шаге? Да - 1");**

**check = in.nextInt();**

**if(check == 1){**

**System.out.println(count);**

**}**

**}**

**System.out.println(p);**

**}**

**public static void main(String[] args) {**

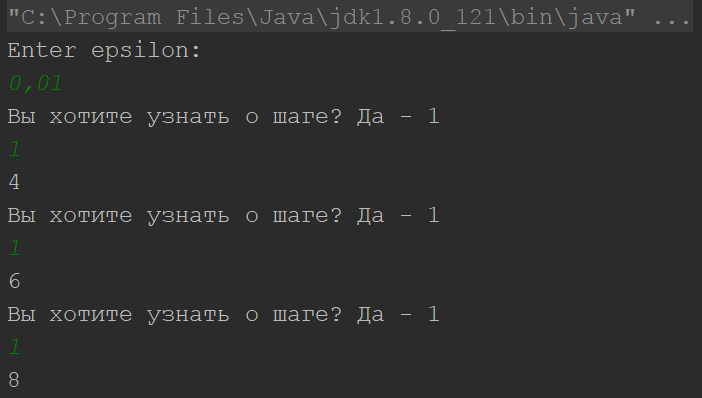
**Thread a = new Thread(new Test());**

**a.start();**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.8 Робота з потоками даних (додаткове завдання)

*Код програми 1.8:*

**Test.java**

**package lab5\_4;**

**import java.util.Scanner;**

**public class Test implements Runnable {**

**@Override**

**public void run() {**

**int i ;**

**System.out.println("Enter the value: ");**

**Scanner in = new Scanner(System.in);**

**try {**

**i = in.nextInt();**

**for (int j = 2; i / j != 1; j++) {**

**for (; ; ) {**

**if (i > 0 && i % j == 0) {**

**i -= j;**

**System.out.println(j + " ");**

**} else {**

**break;**

**}**

**}**

**if(j==0)**

**break;**

**}**

**}catch(NumberFormatException e){**

**System.out.println("Incorrect data");**

**}**

**}**

**public static void main(String[] args) {**

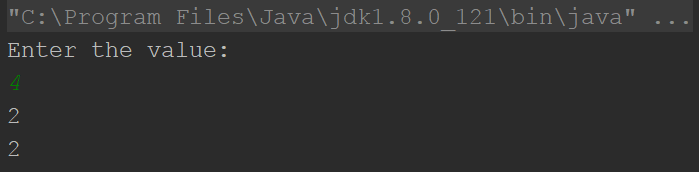
**Thread t = new Thread(new Test());**

**t.start();**

**}**

**}**

*Виведення даних у консольне вікно:*



**Висновок:**

У даній роботі за допомогою мови програмування Java та використання попереднього досвіду було cтворено програми графічного інтерфейсу користувача, під час розробки було використано рефлексії та метапрограмування. Також була опрацьована робота з потоками виконання.