

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №4
З предмету «Об'єктно-орієнтоване програмування»

Виконав
Студент групи КН-36а
Рубан Ю.Д.
Перевірили:
Козуля М.М.
Кізілов О.С.

Харків 2017

1 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 $h(x) = f(x + a) + g(x - b)$

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.

1.7 Обчислення π в окремому потоці виконання (додаткове завдання)

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

Обчислення здійснювати в окремому потоці виконання. Під час виконання обчислення надавати користувачеві можливість вводити 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.

2 Хід виконання роботи

Завдання 1:

Створено 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 $h(x) = f(x + a) + g(x - b)$

Код програми:

Файл Controller.java

```
import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;
import javafx.scene.layout.StackPane;

import javax.tools.JavaCompiler;
import javax.tools.ToolProvider;
import java.io.File;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class Controller {
    @FXML
    private TextField a;
    @FXML
    private TextField b;
    @FXML
    private TextField y;
    @FXML
    private TextField g;
    @FXML
    private LineChart<Number,Number> chart;
    @FXML
    private TextField beg;
    @FXML
    private TextField en;
    boolean compile(String sourceFile) {
        JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
        return compiler.run(null, null, null, sourceFile) == 0;
    }
    @FXML
    @SuppressWarnings("resource")
    public void build()
    {
        try {
            chart.getData().clear();
            Double begin = Double.parseDouble(beg.getText());
            Double end = Double.parseDouble(en.getText());
            String first = y.getText();
            String second = g.getText();
            Double param1, param2;
            param1 = Double.parseDouble(a.getText());
```

```

        param2 = Double.parseDouble(b.getText());
        try {
            StringProcessor sp = new StringProcessor();
            sp.genY(first);
            sp.genG(second);
            if(compile(sp.gfile) && compile(sp.yfile)) {
                File yfileClass = new File("out/production/lab4
task1/y.class");
                File gfileClass = new File("out/production/lab4
task1/g.class");
                File yfileJava = new File("out/production/lab4
task1/y.java");
                File gfileJava = new File("out/production/lab4
task1/g.java");
                while (!yfileClass.exists() &&yfileClass.isFile() &&
!gfileClass.exists() && gfileClass.isFile()){
                    while (!yfileJava.exists() &&yfileJava.isFile() &&
!gfileJava.exists() && gfileJava.isFile()){
                        Class<?> c = Class.forName("y");
                        Class<?> c2 = Class.forName("g");
                        Method ym = c.getMethod("f", Double.class, Double.class);
                        Method gm = c2.getMethod("f", Double.class,
Double.class);
                        System.out.println(ym.invoke(null,new
Object[]{100.,0.}));
                        System.out.println(gm.invoke(null,new
Object[]{100.,0.}));
                        yfileClass.delete();
                        gfileClass.delete();
                        yfileJava.delete();
                        gfileJava.delete();
                        XYChart.Series<Number, Number> series = new
XYChart.Series<>();
                        double step = (end - begin) / 1000;
                        for (double i = begin; i <= end; i += step) {
                            series.getData().add(new XYChart.Data<>((Number) i,
(Number) (((Double) ym.invoke(null, new Object[]{i, param1})) + ((Double)
gm.invoke(null, new Object[]{i, param2}))))));
                        }

                        chart.getData().add(series);
                        StackPane stackPane = new StackPane();
                        for (XYChart.Data data :
chart.getData().get(0).getData()) {
                            stackPane.getChildren().add(data.getNode());
                            stackPane.setVisible(false);
                        }
                    }
                } catch (ClassNotFoundException e) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.show();
                } catch (NoSuchMethodException e) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.show();
                } catch (IllegalAccessException e) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.show();
                } catch (InvocationTargetException e) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.show();
                }
            } catch (Exception e)
            {
                Alert alert = new Alert(Alert.AlertType.ERROR);

```

```

        alert.show();
    }
}

```

Файл Interface.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 Interface extends Application {

    private Stage primaryStage;
    private AnchorPane rootLayout;

    @Override
    public void start(Stage primaryStage) {
        this.primaryStage = primaryStage;
        this.primaryStage.setTitle("task1");

        initRootLayout();
    }

    /**
     * Инициализирует корневой макет.
     */
    public void initRootLayout() {
        try {
            // Загружаем корневой макет из fxml файла.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Interface.class.getResource("sample.fxml"));
            rootLayout = (AnchorPane) loader.load();

            // Отображаем сцену, содержащую корневой макет.
            Scene scene = new Scene(rootLayout);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

Файл StringProcessor.java

```

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;

public class StringProcessor {

    final String yfile = "out\\production\\lab4 task1\\y.java";
    final String gfile = "out\\production\\lab4 task1\\g.java";

    void genY(String expression) {

```

```

        try (PrintWriter out = new PrintWriter(yfile)) {
            expression=expression.replace("\\", "");
            out.println("public class y {");
            out.println("    public static Double f(Double x, Double arg) {");
            out.println("        x+=arg;");
            out.println("        return " + expression + ";");
            out.println("    }");
            out.println("}");
            out.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    void genG(String expression) {
        try (PrintWriter out = new PrintWriter(gfile)) {
            expression=expression.replace("\\", "");
            out.println("public class g {");
            out.println("    public static Double f(Double x, Double arg)");
            out.println("        {");
            out.println("            x=x-arg;");
            out.println("            return " + expression + ";");
            out.println("        }");
            out.println("}");
            out.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

Файл sample.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.chart.CategoryAxis?>
<?import javafx.scene.chart.LineChart?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/8.0.112" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="Controller">
    <children>
        <TextField fx:id="a" layoutX="14.0" layoutY="69.0" prefHeight="25.0"
prefWidth="42.0" />
        <TextField fx:id="b" layoutX="73.0" layoutY="69.0" prefHeight="25.0"
prefWidth="42.0" />
        <TextField fx:id="y" layoutX="10.0" layoutY="126.0" />
        <TextField fx:id="g" layoutX="10.0" layoutY="177.0" />

        <LineChart fx:id="chart" layoutX="190.0" layoutY="3.0"
prefHeight="400.0" prefWidth="416.0">
            <xAxis>
                <NumberAxis />
            </xAxis>
            <yAxis>
                <NumberAxis />
            </yAxis>
        </LineChart>
    </children>
</AnchorPane>

```

```

</LineChart>
<Label layoutX="14.0" layoutY="41.0" text="a" />
<Label layoutX="71.0" layoutY="41.0" text="b" />
<Label layoutX="14.0" layoutY="109.0" text="y(x)" />
<Label layoutX="10.0" layoutY="160.0" text="g(x)" />
<TextField fx:id="beg" layoutX="10.0" layoutY="263.0" prefHeight="25.0"
prefWidth="63.0" />
<TextField fx:id="en" layoutX="107.0" layoutY="263.0" prefHeight="25.0"
prefWidth="63.0" />
<Label layoutX="10.0" layoutY="246.0" text="begin" />
<Label layoutX="107.0" layoutY="246.0" text="end" />
<Button layoutX="9.0" layoutY="336.0" mnemonicParsing="false"
onAction="#build" text="Button" />
</children>
</AnchorPane>

```

Результат виконання на рисунку 1.

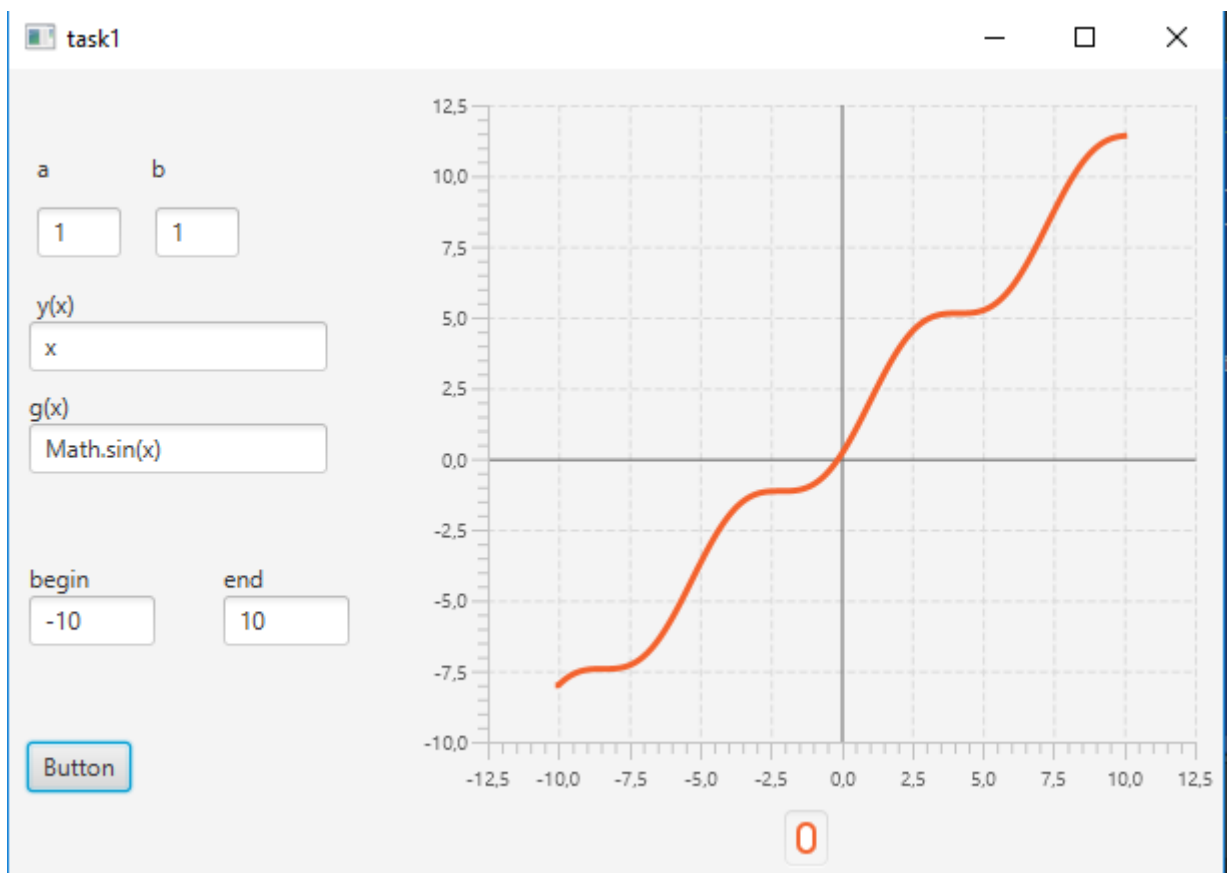


Рис 2.1 – Результат програми завдання 1

Завдання 2:

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

Код програми:

```
import java.lang.reflect.Field;
```



```

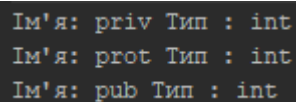
public class ShowAllMethods {

    @SuppressWarnings("resource")
    public static void main(String[] args) {
        try {
            Class<?> c = Class.forName("Test");
            for (Field m : c.getDeclaredFields()) {
                System.out.printf("Ім'я: %s Тип : %s\n", m.getName(),
m.getType());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class Test {
    private int priv;
    protected int prot;
    public int pub;
}

```

Результат виконання на рисунку 2.



```

Ім'я: priv Тип : int
Ім'я: prot Тип : int
Ім'я: pub Тип : int

```

Рисунок 2.2 – Результат виконання програми завдання 2

Завдання 3:

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

Код програми:

Файл Counter.java

```

import java.util.*;

public class Counter {
    private java.util.List<Decompose> numbers;
    private int begin;
    private int end;
    public Counter(int begin,int end)
    {
        this.begin=begin;
        this.end=end;
        numbers = new LinkedList<>();
        if(begin>end)throw new IllegalArgumentException();
        if(begin<0 || end<0)throw new IllegalArgumentException();
    }

    public int getBegin() {

```

```

        return begin;
    }

    public int getEnd() {
        return end;
    }

    public synchronized List<Decompose> getNumbers() {
        return numbers;
    }

    public synchronized Integer getLastAdded()
    {
        if(numbers.size()>1)
            return numbers.get(numbers.size()-1).getNumber().getValue();
        else return -1;
    }
}

```

Файл Decompose.java

```

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;

public class Decompose {
    private IntegerProperty number;
    private StringProperty decompose;
    public Decompose(int number,String decompose)
    {
        this.number = new SimpleIntegerProperty(number);
        this.decompose = new SimpleStringProperty(decompose);
    }

    public IntegerProperty getNumber() {
        return number;
    }

    public StringProperty getDecompose() {
        return decompose;
    }

    public void setNumber(IntegerProperty number) {
        this.number = number;
    }

    public void setDecompose(StringProperty decompose) {
        this.decompose = decompose;
    }
}

```

Файл Simplimizer.java

```

public class Simplimizer implements Runnable
{
    private int b,e;
    private Counter c;
    private Thread primeThread;
    public Simplimizer(Counter c)
    {
        this.b=c.getBegin();
        this.e=c.getEnd();
        this.c=c;
    }
}

```

```

@Override
public void run()
{
    for(int i = b;i<=e;i++)
    {
        String temp = "";
        int k=i;
        int l=2;
        while(l<=k)
        {
            if(k%l==0)
            {
                temp+=l;
                k=k/l;
                if(k>1)
                {
                    temp+="*";
                }
            }
            else l++;
        }
        c.getNumbers().add(new Decompose(i,temp));
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e1)
        {
        }
    }
}

public void start()
{
    primeThread = new Thread(this);
    primeThread.start();
}

public void stop()
{
    primeThread.stop();
}

public void suspend()
{
    primeThread.suspend();
}

public void resume()
{
    primeThread.resume();
}

public void restart(Counter counter)
{
    c = counter;
    b=c.getBegin();
    e=c.getEnd();
    if(primeThread!=null)
    {
        primeThread.stop();
        start();
    }
}
}

```

}
 Файл Interface.java

```

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.StringProperty;
import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;

public class Controller {
    private Counter c;
    private Simplimizer simplimizer;
    private Thread update;
    @FXML
    private TextField begin;
    @FXML
    private TextField end;
    private TableView<Decompose> tableView;
    @FXML
    private TableColumn<Decompose, Integer> number;
    @FXML
    private TableColumn<Decompose, String> decompose;
    private void initialize() {
        number.setCellValueFactory(cellData ->
cellData.getValue().getNumber().asObject());
        decompose.setCellValueFactory(cellData ->
cellData.getValue().getDecompose());
    }

    @FXML
    public void go_button()
    {
        try {
            clear();
            c = new Counter(Integer.parseInt(begin.getText()),
Integer.parseInt(end.getText()));
            simplimizer = new Simplimizer(c);
            update = new Thread(new Update(tableView, c));
            simplimizer.start();
            update.start();
        } catch (Exception e)
        {
            Alert alert = new Alert(Alert.AlertType.ERROR, "Ошибка");
            alert.show();
        }
    }
    @FXML
    public void stop_button()
    {
        simplimizer.suspend();
    }
    @FXML
    public void continue_button()
    {
        simplimizer.resume();
    }
    private void clear()
    {
        if(update!=null)
            update.stop();
        if(simplimizer!=null)
            simplimizer.stop();
    }

```

```

        tableView.refresh();
    }
}
class Update implements Runnable
{
    TableView<Decompose>tableView;
    Counter list;
    Update(TableView<Decompose>d, Counter list)
    {
        tableView = d;
        this.list= list;
    }
    @Override
    public void run() {
        Integer temp = new Integer(-1);
        while (true) {
            if(temp==list.getLastAdded()){continue;}
            temp=list.getLastAdded();
        }
    }
}
tableView.setItems(FXCollections.observableList(list.getNumbers()));
}
}

```

Файл sample.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/8.0.112" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="Controler">
    <children>
        <TextField fx:id="begin" layoutX="266.0" layoutY="123.0" />
        <TextField fx:id="end" layoutX="437.0" layoutY="123.0" />
        <TableView fx:id="tableView" layoutX="14.0" layoutY="14.0"
prefHeight="376.0" prefWidth="209.0">
            <columns>
                <TableColumn fx:id="number" prefWidth="78.0" text="Число" />
                <TableColumn fx:id="decompose" prefWidth="130.0" text="Множители"
/>
            </columns>
        </TableView>
        <Label layoutX="266.0" layoutY="95.0" text="от" />
        <Label layoutX="437.0" layoutY="95.0" text="до" />
        <Button onAction="#go_button" layoutX="266.0" layoutY="200.0"
mnemonicParsing="false" text="Вперед" />
        <Button onAction="#stop_button" layoutX="348.0" layoutY="306.0"
mnemonicParsing="false" text="Стоп" />
        <Button onAction="#continue_button" layoutX="451.0" layoutY="306.0"
mnemonicParsing="false" text="Продолжить" />
    </children>
</AnchorPane>

```

Результат виконання програми на рисунку 3.


```

    }
    private int add;
    private BlockingQueue<Integer>bq;
    @Override
    public void run()
    {
        try {
            for (int i = 0;i<add ; i++) {
                bq.put(i);

                System.out.println("added = "+i);
            }
        }catch (Exception e){}
    }
}

```

Файл avger.java

```

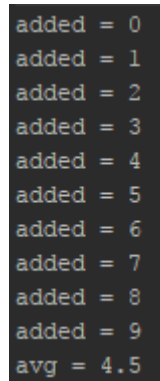
import java.util.concurrent.BlockingQueue;

public class avgder implements Runnable {
    private BlockingQueue<Integer> bq;
    public avgder(BlockingQueue<Integer>e)
    {
        bq=e;
    }

    @Override
    public void run()
    {
        try {
            double avg=0;
            int size = bq.size();
            for (int i = 0;i<size; i++) {
                avg+=bq.take();
            }
            avg/=size;
            System.out.println("avg = "+avg);
        }catch (Exception e){}
    }
}

```

Результат виконання на рисунку 4



```

added = 0
added = 1
added = 2
added = 3
added = 4
added = 5
added = 6
added = 7
added = 8
added = 9
avg = 4.5

```

Рис. 2.4 – Результат програми завдання 4

Завдання 5:

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

Код програми:

```
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class Main {
    public static void main(String[] args) {
        try {
            Method m;
            Class<?>c = Class.forName("A");
            m= c.getMethod("meth");
            m.invoke(c.newInstance());
            c = Class.forName("B");
            m=c.getMethod("meth");
            m.invoke(c.newInstance());
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        }
    }
}

class A
{
    public void meth()
    {
        System.out.println("A");
    }
}

class B
{
    public void meth()
    {
        System.out.println("B");
    }
}
```

Результат виконання на рисунку 5



Рис 2.5 – Результат програми завдання 5

Завдання 6:

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

Код програми:

```
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;
```

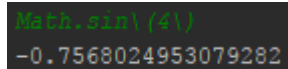


```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ScriptException {
        ScriptEngineManager factory = new ScriptEngineManager();
        ScriptEngine engine = factory.getEngineByName("JavaScript");
        String expr = "c = ";
        Scanner scanner = new Scanner(System.in);
        String exp1 = scanner.nextLine();
        expr+=exp1;

        Object res = engine.eval(expr.replace("\\\\", "\\"));
        System.out.println(res);
    }
}
```

Результат виконання на рисунку 6.



```
Math.js>
-0.7568024953079282
```

Рис 2.6 – Результат виконання програми завдання 6.

Завдання 7:

Реалізовано програму обчислення π . Обчислення здійснюється в окремому потоці виконання. Під час виконання обчислення користувачеві надається можливість уводити запит про кількість обчислених елементів суми.

Код програми:

```
public class Main {

    public static void main(String[] args) {
        PiCounter pi = new PiCounter();
        Thread counter = new Thread(pi);
        counter.start();
        try {
            Thread.sleep(250);
            System.out.println(pi.getIter());
            System.out.println(pi.getPi());
            Thread.sleep(250);
            System.out.println(pi.getIter());
            System.out.println(pi.getPi());
            Thread.sleep(250);
            counter.interrupt();
            System.out.println(pi.getIter());
            System.out.println(pi.getPi());

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class PiCounter implements Runnable
{
    private double pi=0;
    private int iter=0;
    public synchronized double getPi() {
```

```

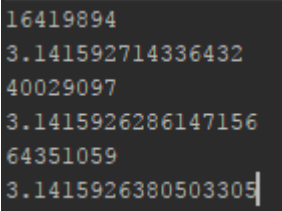
        return pi;
    }

    public synchronized int getIter() {
        return iter;
    }

    @Override
    public void run()
    {
        double i=1;
        int k=1;
        while(true)
        {
            pi+=(4./i)*k;
            i+=2;
            k*=-1;
            iter++;
        }
    }
}

```

Результат виконання програми на рисунку 7.



```

16419894
3.141592714336432
40029097
3.1415926286147156
64351059
3.1415926380503305

```

Рис. 2.7 – Результат виконання програми завдання 7.

Завдання 8:

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

Код програми:

```

public class Main {
    public static void main(String[] args) throws InterruptedException {
        additions ad = new additions(11);
        Thread thread = new Thread(ad);
        thread.start();
        thread.join();
    }
}

class additions implements Runnable
{
    Integer number;
    additions(Integer number)
    {
        this.number=number;
    }
    public Integer getNumber() {
        return number;
    }

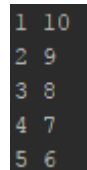
    public void setNumber(Integer number) {
        this.number = number;
    }
}

```

```
}

@Override
public void run()
{
    for(Integer i=1;i<=number/2;i++)
    {
        System.out.println(i+" "+(number-i));
    }
}
}
```

Результат виконання програми на рисунку 8.



1	10
2	9
3	8
4	7
5	6

Рис. 2.8 – Результат програми завдання 8

Висновки:

У даній лабораторній роботі було засвоєне використання рефлексії та метапрограмування, а також використання потоків виконання.