**Тема**: Використання рефлексії та метапрограмування. Робота з потоками виконання

**Завдання**:

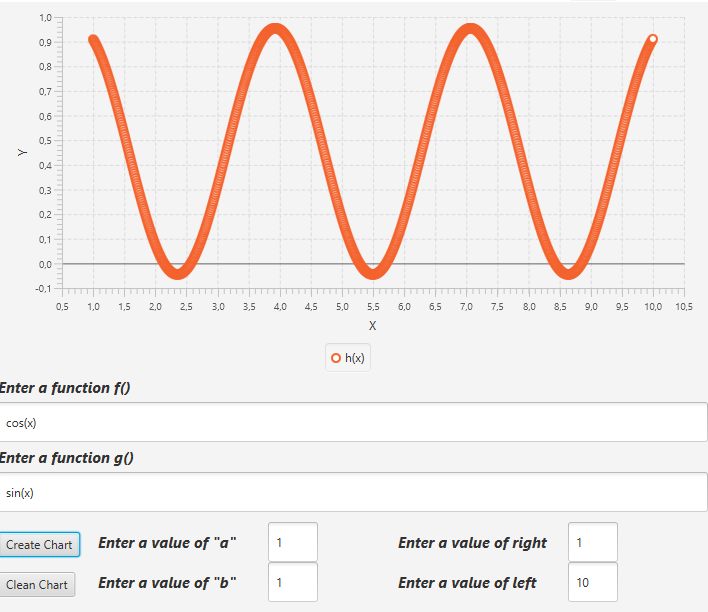
1.1 Індивідуальне завдання

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: f(x – a)∙g(x + b).

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

Розв’язання:

package lab3\_1;  
  
public class Main extends Application {  
 TextField text1,text2,text3,text4,text5,text6;  
 Label label1,label2,label3,label4,label5,label6;  
 LineChart linechart;  
 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"};  
 ScriptEngineManager factory;  
 ScriptEngine engine;  
 Pane root;  
 NumberAxis xAxis;  
 NumberAxis yAxis;  
 public static void main(String[] args) {  
 launch(args);  
 }  
  
 @Override  
 public void init() throws ScriptException {  
 factory = new ScriptEngineManager();  
 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));  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 primaryStage.setTitle("Lab4\_1");  
 root = new Pane();  
 Scene scene = new Scene(root,700,600);  
 text1 = new TextField ();  
 text1.setPrefSize(710,40);  
 text1.setLayoutY(scene.getWidth()-300);  
 text2 = new TextField ();  
 text2.setPrefSize(710,40);  
 text2.setLayoutY(scene.getWidth()-230);  
 text3 = new TextField ();  
 text3.setPrefSize(50,40);  
 text3.setLayoutY(scene.getWidth()-180);  
 text3.setLayoutX(270);  
 text4 = new TextField ();  
 text4.setPrefSize(50,40);  
 text4.setLayoutY(scene.getWidth()-140);  
 text4.setLayoutX(270);  
 text5 = new TextField ();  
 text5.setPrefSize(50,40);  
 text5.setLayoutY(scene.getWidth()-180);  
 text5.setLayoutX(570);  
 text6 = new TextField ();  
 text6.setPrefSize(50,40);  
 text6.setLayoutY(scene.getWidth()-140);  
 text6.setLayoutX(570);  
 label1=new Label("Enter a function f()");  
 label1.setPrefSize(500,20);  
 label1.setFont(Font.font("TimesNewRoman", FontWeight.BOLD, FontPosture.ITALIC, 15));  
 label1.setLayoutY(scene.getWidth()-325);  
 label2=new Label("Enter a function g()");  
 label2.setPrefSize(500,20);  
 label2.setFont(Font.font("TimesNewRoman", FontWeight.BOLD, FontPosture.ITALIC, 15));  
 label2.setLayoutY(scene.getWidth()-255);  
 label3=new Label("Enter a value of \"a\"");  
 label3.setPrefSize(150,20);  
 label3.setFont(Font.font("TimesNewRoman", FontWeight.BOLD, FontPosture.ITALIC, 15));  
 label3.setLayoutY(scene.getWidth()-170);  
 label3.setLayoutX(100);  
 label4=new Label("Enter a value of \"b\"");  
 label4.setPrefSize(150,20);  
 label4.setFont(Font.font("TimesNewRoman", FontWeight.BOLD, FontPosture.ITALIC, 15));  
 label4.setLayoutY(scene.getWidth()-130);  
 label4.setLayoutX(100);  
 label5=new Label("Enter a value of right");  
 label5.setPrefSize(150,20);  
 label5.setFont(Font.font("TimesNewRoman", FontWeight.BOLD, FontPosture.ITALIC, 15));  
 label5.setLayoutY(scene.getWidth()-170);  
 label5.setLayoutX(400);  
 label6=new Label("Enter a value of left");  
 label6.setPrefSize(150,20);  
 label6.setFont(Font.font("TimesNewRoman", FontWeight.BOLD, FontPosture.ITALIC, 15));  
 label6.setLayoutY(scene.getWidth()-130);  
 label6.setLayoutX(400);  
 Button btn1 = new Button("Create Chart");  
 btn1.setLayoutY(scene.getWidth()-170);  
 btn1.setOnAction(this::AddChart);  
 Button btn2 = new Button("Clean Chart");  
 btn2.setLayoutY(scene.getWidth()-130);  
 btn2.setOnAction((event)->{  
 linechart.getData().clear();  
 xAxis.setTickLabelsVisible(false);  
 yAxis.setTickLabelsVisible(false);  
 });  
 xAxis = new NumberAxis() ;  
 xAxis.setLabel("X") ;  
 xAxis.setForceZeroInRange(false);  
 xAxis.setTickLabelsVisible(false);  
 yAxis = new NumberAxis() ;  
 yAxis.setLabel("Y");  
 yAxis.setForceZeroInRange(false);  
 yAxis.setTickLabelsVisible(false);  
 linechart = new LineChart(xAxis,yAxis) ;  
 linechart.setLayoutX(0);  
 linechart.setPrefHeight(375);  
 linechart.setPrefWidth(700);  
 linechart.setLayoutX(0);  
 linechart.setLayoutY(0);  
 root.getChildren().addAll(linechart);  
 root.getChildren().addAll(text1,text2,label1,label2,label3,text3,label4,text4,label5,label6,text5,text6,btn1,btn2);  
 primaryStage.setResizable(false);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private void AddChart(ActionEvent event) {  
 try {  
 if(Double.parseDouble(text5.getText())>=Double.parseDouble(text6.getText())){  
 showMessage("THE RIGHT SIDE IS BIG THEN THE SECOND");  
 return;  
 }  
 XYChart.Series series = new XYChart.Series() ;  
 double t1;  
 double t2;  
 for (double d = Double.parseDouble(text5.getText()); d <= Double.parseDouble(text6.getText()); d += 10) {  
 try {  
 engine.put("x", d - Double.parseDouble(text3.getText()));  
 t1 = (double) engine.eval(text1.getText());  
 engine.put("x", d + Double.parseDouble(text4.getText()));  
 t2 = (double) engine.eval(text2.getText());  
 series.getData().add(new XYChart.Data(d, t1 \* t2));  
 } catch (ScriptException e) {  
 e.printStackTrace();  
 }  
 }  
 series.setName("h(x)");  
 xAxis.setTickLabelsVisible(true);  
 yAxis.setTickLabelsVisible(true);  
 linechart.getData().addAll(series);  
 }catch (NumberFormatException e){  
 showMessage("PLEASE, FULL IN ALL FIELDS CORRECT");  
 }  
  
 }  
  
 public static void showMessage(String message) {  
 Alert alert = new Alert(Alert.AlertType.INFORMATION);  
 alert.setTitle("Error");  
 alert.setHeaderText(message);  
 alert.showAndWait();  
 }  
  
}

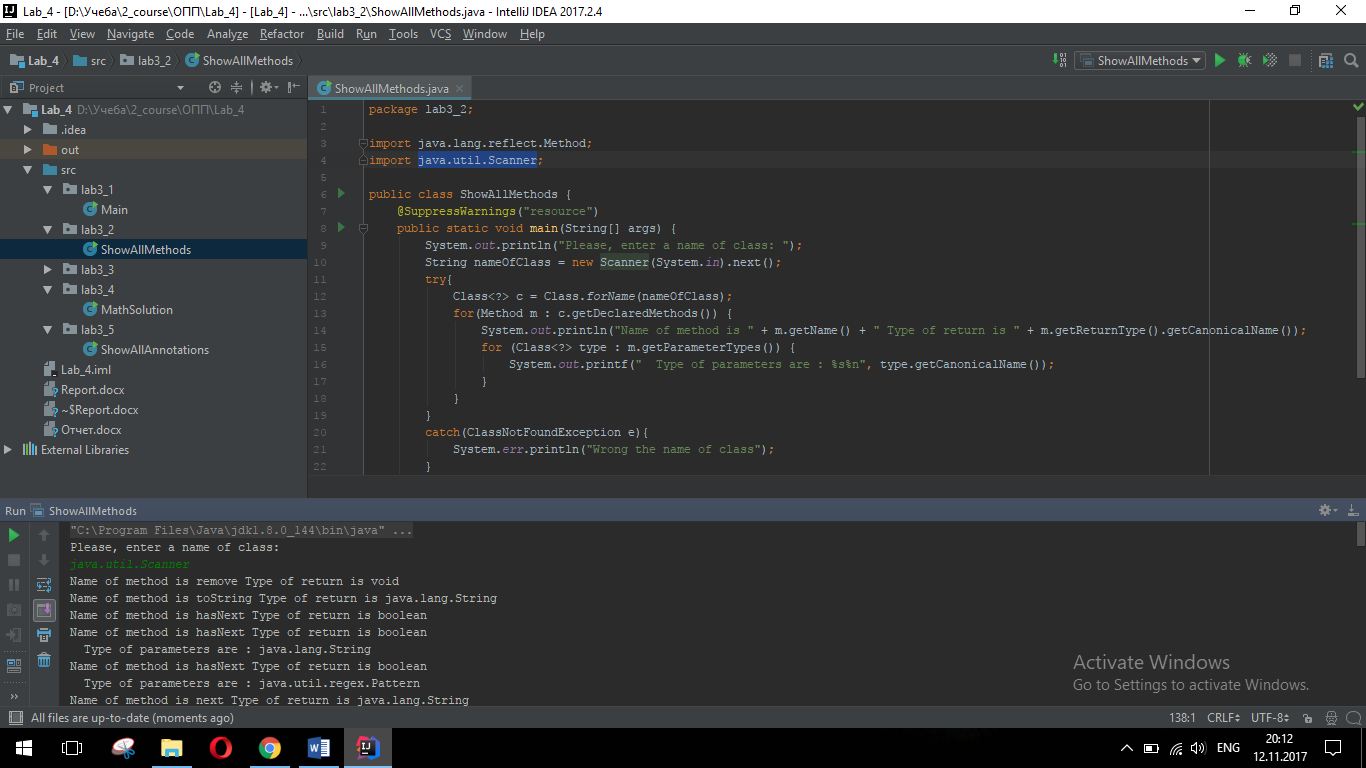


**1.2 Перегляд всіх полів класу**

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

Розв’язання:

package lab3\_2;  
  
import java.lang.reflect.Method;  
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(Method m : c.getDeclaredMethods()) {  
 System.*out*.println("Name of method is " + m.getName() + " Type of return is " + m.getReturnType().getCanonicalName());  
 for (Class<?> type : m.getParameterTypes()) {  
 System.*out*.printf(" Type of parameters are : %s%n", type.getCanonicalName());  
 }  
 }  
 }  
 catch(ClassNotFoundException e){  
 System.*err*.println("Wrong the name of class");  
 }  
 }  
}



***1.3 Робота з BlockingQueue***

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

package Laba4\_4;

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() {

// Намагаємося додавати числа:

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() {

// Вилучаємо числа:

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);

// Створюємо і запускаємо два потоки - для запису і читання:

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

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

producerThread.start();

consumerThread.start();

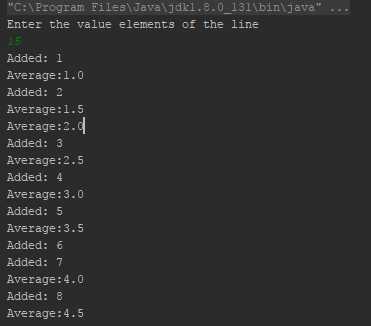
// Чекаємо 10 секунд і перериваємо перший потік:

Thread.sleep(10000);

producerThread.interrupt();

}

}



**1.4 Виклик функції для обраного класу (додаткове завдання)**

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

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();

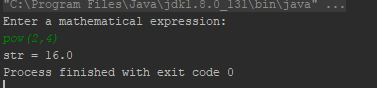
}

}

**1.5 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

package lab3\_4;  
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));  
 }  
}

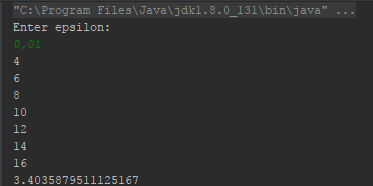


***1.6 Обчислення π в окремому потоці виконання (додаткове завдання)***

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

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

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("Do you want to know about step? If Yes 0 ");  
 check = in.nextInt();  
 if(check == 0){  
 System.*out*.println(count);  
 }  
 }  
 System.*out*.println(p);  
 }  
  
 public static void main(String[] args) {  
 Thread a = new Thread(new Test());  
 a.start();  
 }  
}



**1.7 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.

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;

}

}

}

}catch(NumberFormatException e){

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

}

}

public static void main(String[] args) {

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

t.start();

}

}

**Висновок**: на цій лабораторній роботі було вивчено рефлексію та метапрограмування.