Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ярославский государственный технический университет»

Кафедра «Информационные системы и технологии»

Отчет защищен

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель

\_\_\_\_\_\_\_\_\_\_Д.В.Дидковская

«\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022

**Графический интерфейс**

Отчёт о лабораторной работе №4 по курсу “Информационные технологии”

ЯГТУ 09.03.02-024 ЛР

|  |  |
| --- | --- |
|  | Отчет выполнил  студент группы ЭИС-26  \_\_\_\_\_\_\_\_\_\_\_ А.А.Хрящев  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 |

2022

Цель работы:

Освоить графический интерфейс среды NetBeans.

Задание:

Создать приложения по лабораторным работам: №1, №2 (задание I, только свой вариант), №3 с использованием графического интерфейса.

Код программы:

Disposable subscribe = Observable.just(

new Work4\_one\_part().getjPanel(),

new Work4\_two\_part().getJPanel(),

new Work4\_three\_part().getJPanel())

.unsubscribeOn(Schedulers.io())

.subscribe(next -> {

JFrame frame = new JFrame("App");

frame.setContentPane(next);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setSize(600,400);

frame.setVisible(true);

});

subscribe.dispose();

По 1 лабораторной:

package View;  
  
import example.LabWork1;  
  
import javax.swing.\*;

class Work1 {

@Value

@Builder(toBuilder = true)

public static class One {

@NonNull

double a, b, from, to, dx;

PrintIterator printIterator;

public double f(double x) {

if(a-x>b) {

if(x > 0) return f1(x);

if(x < 0) return f3(x);

}

if (a-x<b) return f2(x);

throw new ArithmeticException("Unknown");

}

public double f1(double x) {

// System.out.println("f1");

return 2.805 \* Math.log(Math.pow(x, 4) - (2 \* a));

}

public double f2(double x) {

// System.out.println("f2");

return Math.sqrt(Math.pow(a, 2) \* Math.pow(b, 3) - 4) + x;

}

public double f3(double x) {

// System.out.println("f3");

if(x == 0)

throw new ArithmeticException("Div by zero");

return Math.cos(Math.abs((2 \* a) / (b \* x))) + 3.7;

}

public void execute(double x) {

printIterator.print(" x f(x)\n");

while (x<=to) {

printIterator.print("| " + x + "\t\t| " + f(x) + "\n");

x+=dx;

}

}

}

@Builder

public static class Two {

@NonNull

double eps, Xn, Xk, dX;

@NonNull

PrintIterator printIterator;

public void execute() {

double x=Xn;

double T,s=0;

T= Math.sin(x);

printIterator.print(" x: ");

while(x<=Xk) {

int n = 1;

while (Math.abs(T) > eps) {

s += T;

n += 1;

T = (Math.sin(2 \* n - 1) \* x) / (2 \* n - 1);

}

// printIterator.print(" " + s + ' ');

printIterator.print(" " + x + '\n');

x+=dX;

}

printIterator.print("\nSum = "+s);

}

}

}

class LabWork1 {

private PrintIterator printIterator;

private LabWork1(PrintIterator printStream) {

this.printIterator = printStream;

}

public static LabWork1 printAction(PrintIterator printIterator) {

return new LabWork1(printIterator);

}

public void partOne() {

Work1.One work1 = Work1.One.builder().printIterator(printIterator).a(-2.83).b(2.05)

.from(-2).to(2).dx(0.5).build();

work1.execute(-2);

}

public void partTwo() {

Work1.Two work2 = Work1.Two.builder().printIterator(printIterator).Xn(Math.PI/10).Xk((9 \* Math.PI)/10)

.dX(0.005 \* Math.PI).eps(Math.pow(10, -3)).build();

work2.execute();

}

}  
  
public class Work4\_one\_part {  
 private JPanel jPanel;  
 private JButton BtnTab;  
 private JButton infinityRowButton;  
 private JTextArea textArea1;  
 private JScrollPane scrollPane;  
 private JTextArea textArea2;  
  
 public Work4\_one\_part() {  
 BtnTab.addActionListener(actionEvent -> LabWork1.*printAction*(text -> textArea1.setText(textArea1.getText() + text)).partOne());  
  
 infinityRowButton.addActionListener(actionEvent -> LabWork1.*printAction*(text -> textArea2.setText(textArea2.getText() + text)).partTwo());  
 }  
  
 public JPanel getjPanel() {  
 return jPanel;  
 }  
}

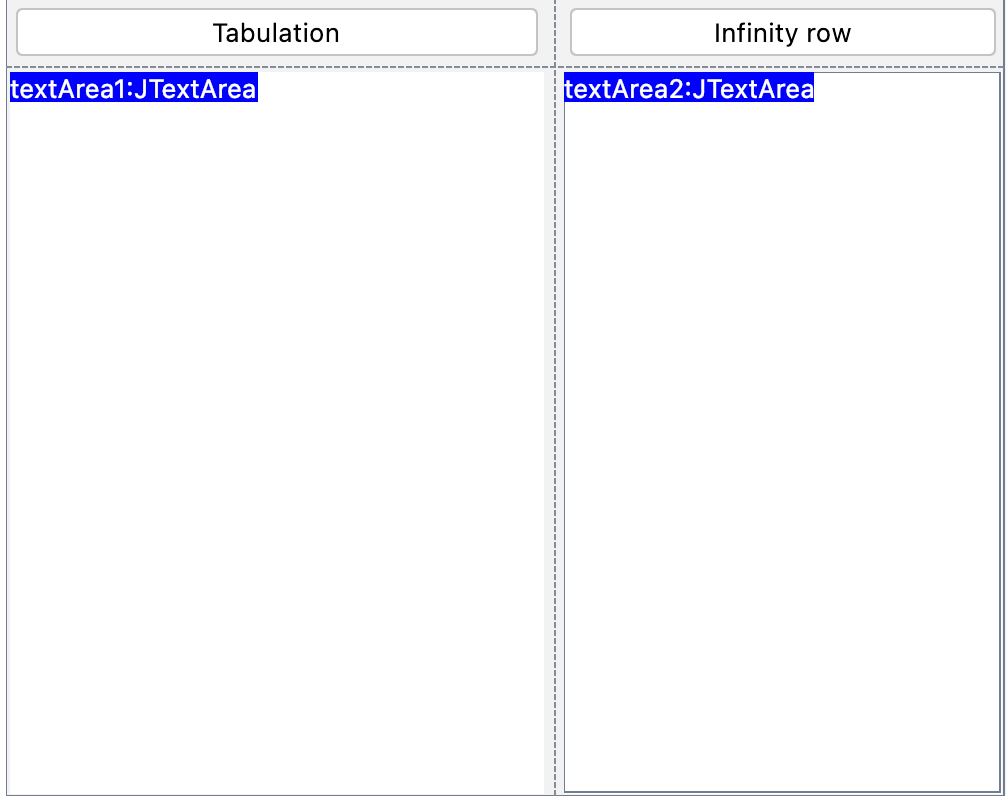


Рисунок 1 – Графический интерфейс программы по 1 лабораторной

По 2 лабораторной:

package View;  
  
import Util.Convert;  
import io.reactivex.rxjava3.core.Observable;  
import io.reactivex.rxjava3.disposables.Disposable;  
import io.reactivex.rxjava3.subjects.BehaviorSubject;  
import works.Work2;  
  
import javax.swing.\*;  
import java.util.\*;

class UtilArray {

public static int barrierElement(int @NonNull[] arr, int value) {

final int size = arr.length;

if (size != 0) {

int last = arr[size - 1];//Сохраним прежний элемент массива

arr[size - 1] = value;//Гарантируем, что value есть в массиве

//Есть гарантия того, что элемент есть в массиве, значит индекс можно не проверять

int i = 0;

for (i = 0; arr[i] != value; ++i) {//Одно условие в цикле

}

arr[size - 1] = last;//Восстанавливаем последний элемент

if (i != (size - 1) || value == last) {//Не уткнулись в барьер или последний элемент был искомым

return i;

}

}

return -1;

}

public static void insertionSort(int @NonNull [] arrayPtr) // сортировка вставками

{

int temp;// временная переменная для хранения значения элемента сортируемого массива

// индекс предыдущего элемента

int item;

for (int counter = 1; counter < arrayPtr.length; counter++)

{

temp = arrayPtr[counter]; // инициализируем временную переменную текущим значением элемента массива

item = counter-1; // запоминаем индекс предыдущего элемента массива

while(item >= 0 && arrayPtr[item] > temp) // пока индекс не равен 0 и предыдущий элемент массива больше текущего

{

arrayPtr[item + 1] = arrayPtr[item]; // перестановка элементов массива

arrayPtr[item] = temp;

item--;

}

}

}

}

class Work2 {

private final PrintIterator printIterator;

private Work2(PrintIterator printStream) {

this.printIterator = printStream;

}

public static Work2 printAction(PrintIterator printStream) {

return new Work2(printStream);

}

public One workOne(int[] a, int[] b) {

return new One(a, b);

}

public Two workTwo(String text) {

return new Two(text);

}

public class One {

private final int[] a;

private final int[] b;

public One(int[] a, int[] b) {

this.a = a;

this.b = b;

}

public void execute() {

printIterator.print("Begin: \n" +

"a = " + Arrays.toString(a) + '\n' +

"b = " + Arrays.toString(b) + '\n');

UtilArray.barrierElement(a, 4);

UtilArray.insertionSort(b);

printIterator.print("Sort array: \n" +

"b = " + Arrays.toString(b) + '\n');

final int findItem = Arrays.binarySearch(b, 4);

final Serializable textFindItem = findItem < 0 ? "undenfided" : findItem;

printIterator.print("First: " + UtilArray.barrierElement(a, 4) + '\n');

printIterator.print("Second: " + textFindItem + '\n');

printIterator.print("Unique: " + Arrays.toString(unionListUnique(a, b)) + '\n');

}

private int[] unionListUnique(int[] a, int[] c) {

int[] arr = unionList(a,c);

return Arrays.stream(arr).distinct().toArray();

}

private int[] unionList(int[] a, int[] b) {

int[]c = new int[a.length+b.length];

int count = 0;

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

c[i] = a[i];

count++;

}

for(int j = 0;j<b.length;j++) {

c[count++] = b[j];

}

return c;

}

}

public class Two {

private String word;

public Two(String word) {

this.word = word;

}

public void execute() {

char[] mas = word.toCharArray();

boolean isSorted = false;

char buf;

while(!isSorted) {

isSorted = true;

for (int i = 0; i < word.length()-1; i++) {

if(mas[i] > mas[i+1]){

isSorted = false;

buf = mas[i];

mas[i] = mas[i+1];

mas[i+1] = buf;

}

}

}

printIterator.print(Arrays.toString(mas));

}

}

}  
  
public class Work4\_two\_part {  
 private JPanel panel;  
 private JButton createArrayButton;  
 private JTextArea textArea1;  
 private JButton sortArrayAndMergeButton;  
 private JTextArea textArea2;  
 private JButton bubbleSortButton;  
 private JTextArea textArea3;  
 private JTextArea textArea4;  
 private Disposable mSubscription;  
  
 List<Integer> mas1 = new ArrayList<>();  
 List<Integer> mas2 = new ArrayList<>();  
 public Work4\_two\_part() {  
 createArrayButton.addActionListener(actionEvent -> {  
 mSubscription = Observable.*range*(0, 10).buffer(2).subscribe(next -> {  
 mas1.add(next.get(0));  
 mas2.add(next.get(1));  
 textArea1.setText(textArea1.getText() + (next.get(0) + " "));  
 textArea4.setText(textArea4.getText() + (next.get(1) + " "));  
 });  
 mSubscription.dispose();  
 });  
  
 sortArrayAndMergeButton.addActionListener(actionEvent -> {  
 BehaviorSubject<String> subject = BehaviorSubject.*create*();  
 Disposable disposable = subject.subscribe(textArea2::setText);  
  
 Work2 work2 = Work2.*printAction*(subject::onNext);  
  
 int[] emit1 = Convert.*covertListToArrayInteger*(mas1);  
 int[] emit2 = Convert.*covertListToArrayInteger*(mas2);  
 Work2.One one = work2.workOne(emit1, emit2);  
 one.execute();  
 disposable.dispose();  
 });  
  
 bubbleSortButton.addActionListener(actionEvent -> {  
 BehaviorSubject<String> subject = BehaviorSubject.*create*();  
 Disposable disposable = subject.subscribe(textArea3::setText);  
  
 Work2 work2 = Work2.*printAction*(subject::onNext);  
  
 Work2.Two two = work2.workTwo("REST14342");  
 two.execute();  
 disposable.dispose();  
 });  
 }  
  
 public JPanel getJPanel() {  
 return panel;  
 }  
}

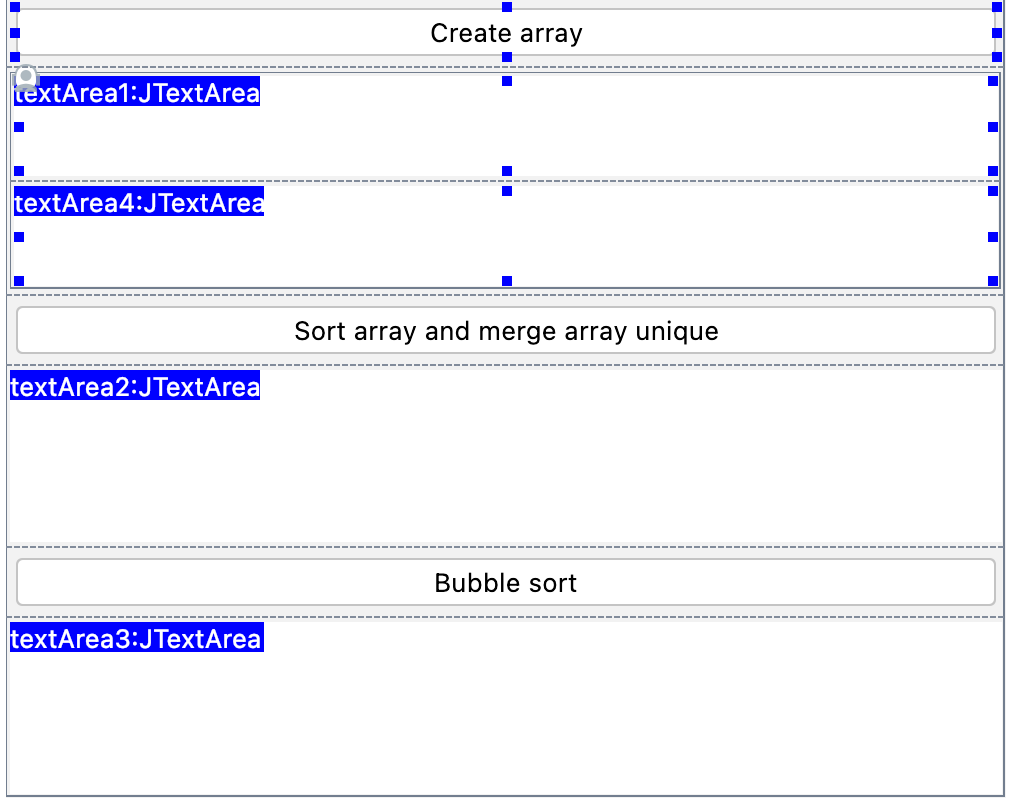


Рисунок 2 – Графический интерфейс программы по 2 лабораторной

По 3 лабораторной:

package View;  
  
import Util.Randomize;  
import io.reactivex.rxjava3.core.Observable;  
import io.reactivex.rxjava3.disposables.Disposable;  
import io.reactivex.rxjava3.subjects.BehaviorSubject;  
import works.Work3;  
  
import javax.swing.\*;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.concurrent.TimeUnit;  
import java.util.stream.Collectors;  
  
public class Work4\_three\_part {  
 private JPanel panel;  
 private JButton createMatrixButton;  
 private JTextArea textArea1;  
  
 public Work4\_three\_part() {  
 createMatrixButton.addActionListener(actionEvent -> {  
 BehaviorSubject<String> behaviorSubject = BehaviorSubject.*create*();  
 Work3 work3 = Work3.*printAction*(behaviorSubject::onNext);  
 Disposable disposable = behaviorSubject.subscribe(next -> {  
 textArea1.setText(textArea1.getText() + next + ' ');  
 });  
  
 for (int i = 0; i < 3; i++) {  
 List<Integer> one = Arrays.*stream*(Randomize.*getRandom*(5, 0, 10)).boxed().collect(Collectors.*toList*());  
 work3.setData(one);  
 }  
  
 work3.average();  
 disposable.dispose();  
 });  
 }  
  
 public JPanel getJPanel() {  
 return panel;  
 }  
}

import Interface.PrintIterator;

import java.util.ArrayList;

import java.util.List;

import java.util.function.Predicate;

import java.util.function.Supplier;

import java.util.stream.Stream;

class ItemList {

private int id;

private int value;

ItemList(int id, int value) {

this.id = id;

this.value = value;

}

public int getValue() {

return value;

}

public int getId() {

return id;

}

}

class Work3 {

private final PrintIterator printIterator;

private Work3(PrintIterator printStream) {

this.printIterator = printStream;

}

public static Work3 printAction(PrintIterator printStream) {

return new Work3(printStream);

}

List<List<ItemList>> data = new ArrayList<>();

private int num = 0;

private double avg = 0.0;

private int count = 0;

public void setData(List<Integer> list) {

printIterator.print("Set data: ");

list.forEach(result -> printIterator.print(result + " "));

printIterator.print("\n");

this.data.add(this.convertList(list));

}

private List<ItemList> convertList(List<Integer> list) {

List<ItemList> itemLists = new ArrayList<>();

list.forEach(item -> {

itemLists.add(new ItemList(num++, item));

});

return itemLists;

}

public void average() {

Supplier<Stream<ItemList>> supplier = () -> data.stream().flatMap(List::stream).filter(item -> item.getValue() % 10 == 3);

printIterator.print("In even rows: ");

filterAndShowStream(supplier, list -> list.getId() / 2 == 0);

printIterator.print("\n");

printIterator.print("In odd rows: ");

filterAndShowStream(supplier, list -> list.getId() / 2 != 0);

}

private void filterAndShowStream(Supplier<Stream<ItemList>> supplier, Predicate<ItemList> var1) {

supplier.get().filter(var1).forEach(result -> {

avg += result.getValue();

count++;

printIterator.print("{ " + result.getId() + ": " + result.getValue() + " }, ");

});

double result = avg/count;

if(Double.isNaN(result))

result = 0;

printIterator.print("\nAvg: " + result + '\n');

avg = 0.0;

count = 0;

}

}

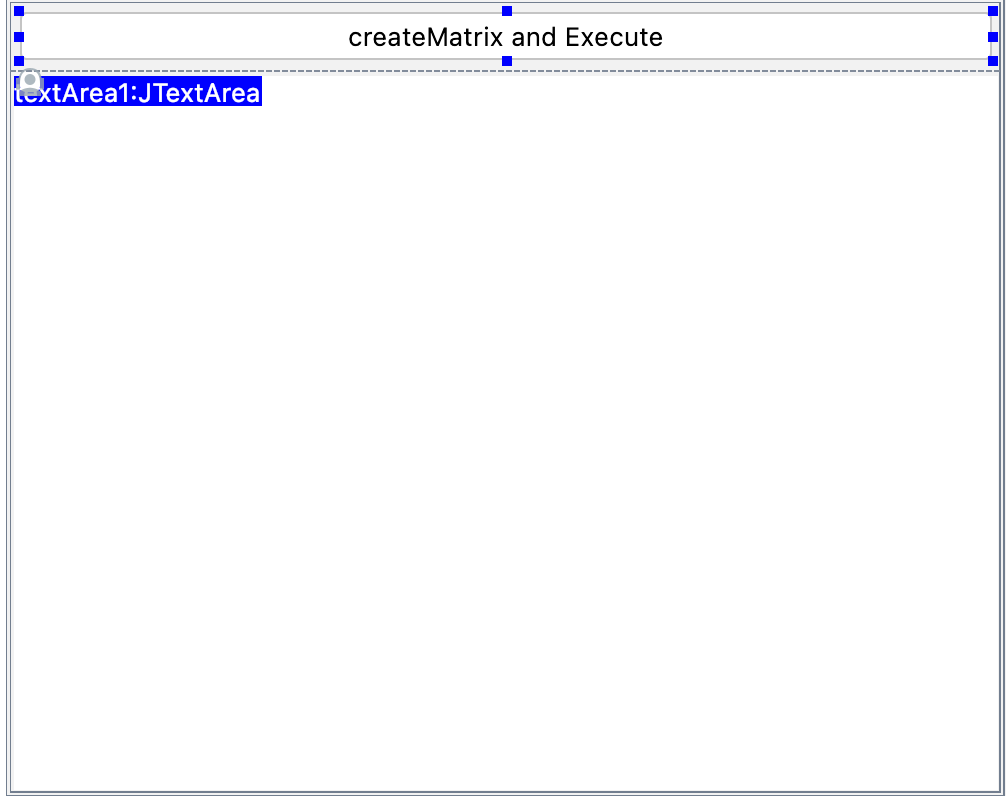


Рисунок 3 – Графический интерфейс программы по 3 лабораторной

Скриншоты выполнения:

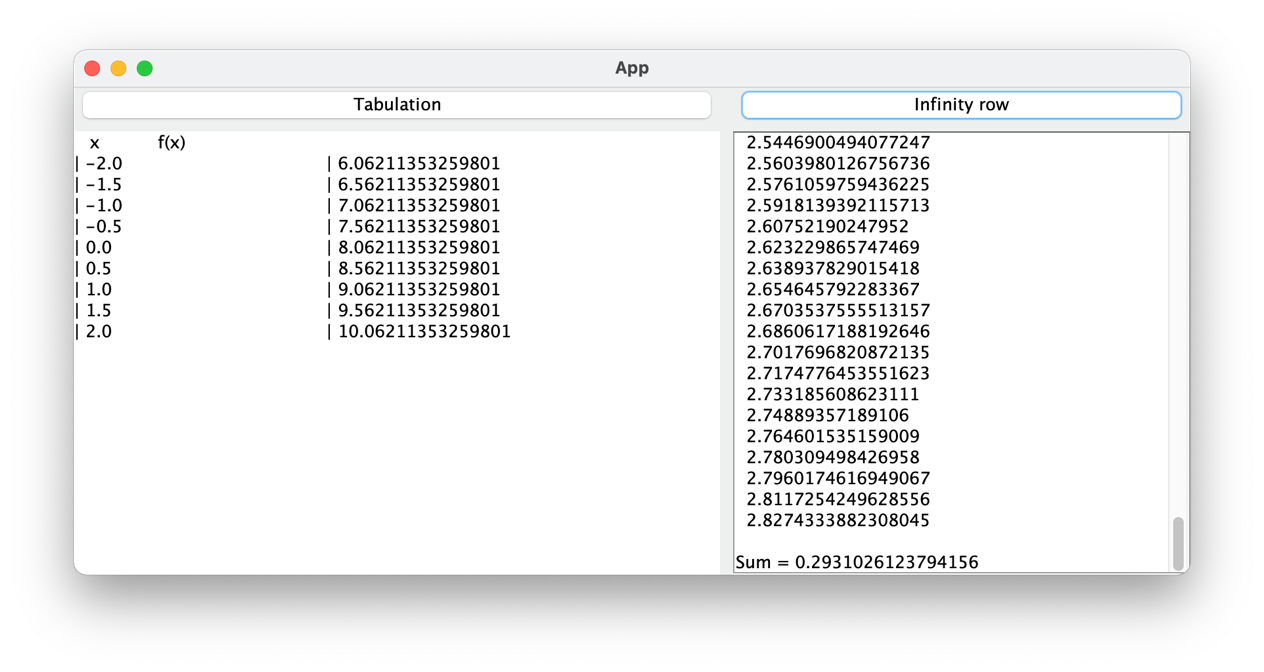


Рисунок 4 – Выполнение программы по 1 лабораторной

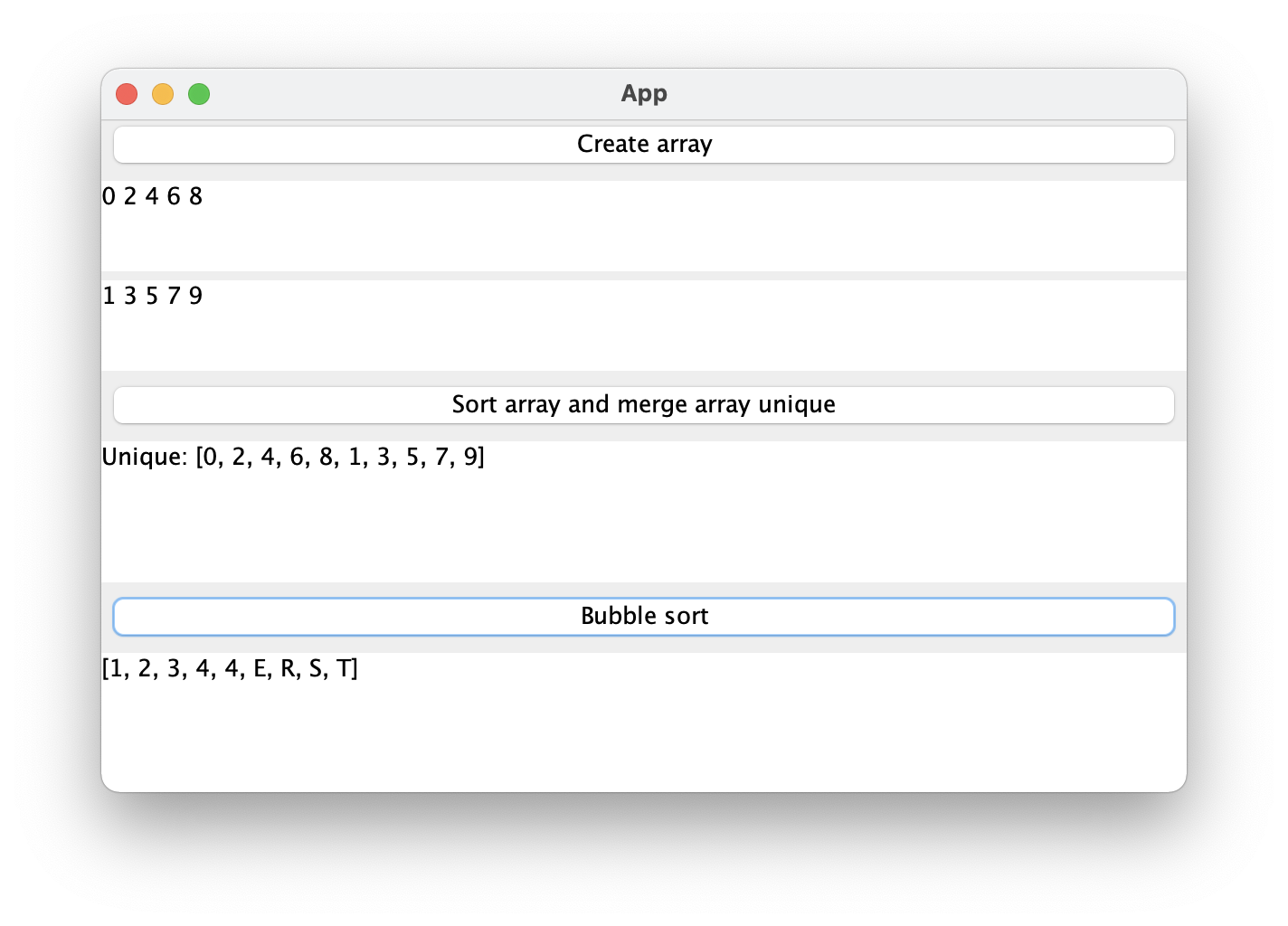


Рисунок 5 – Выполнение программы по 2 лабораторной

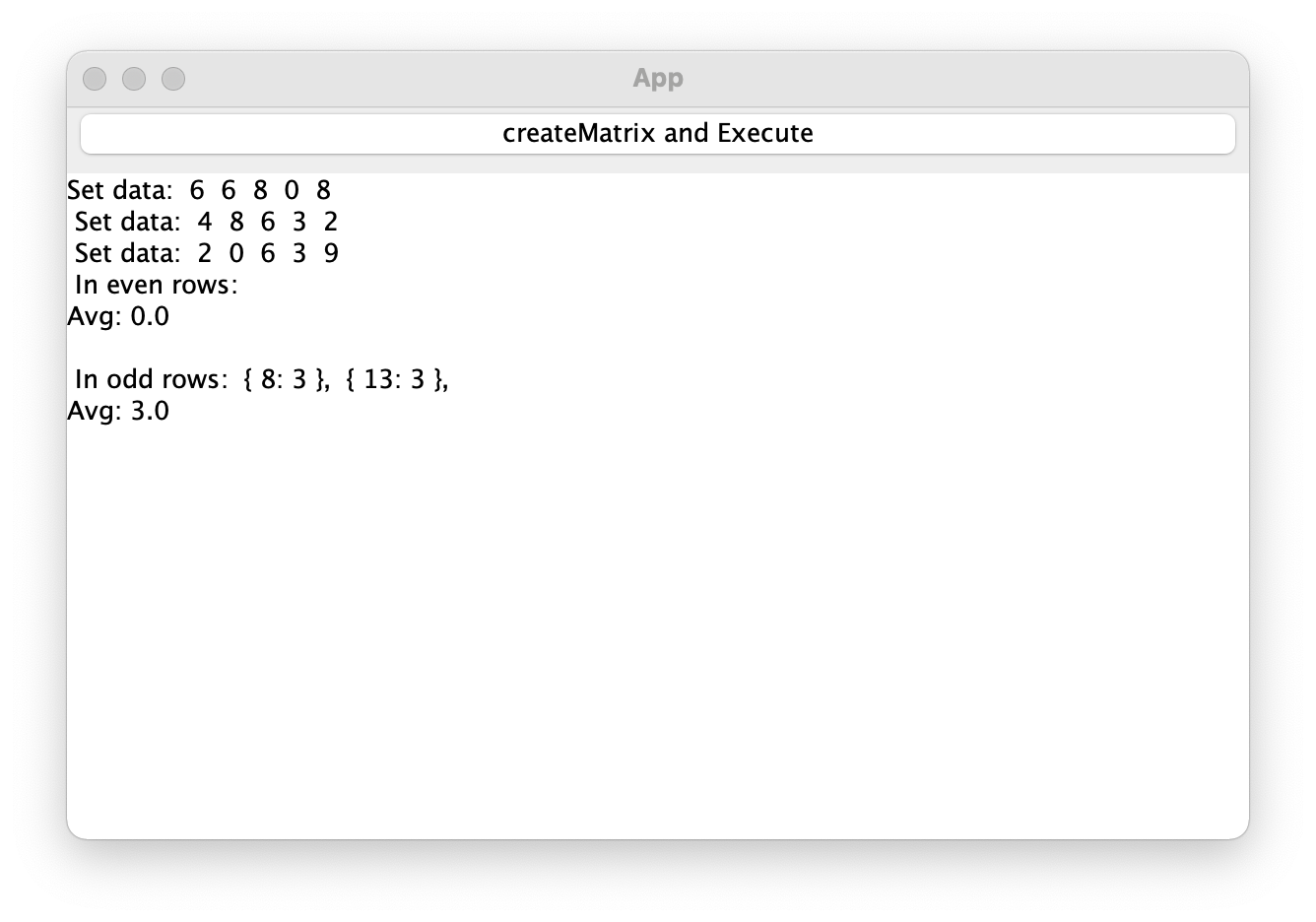


Рисунок 6 – Выполнение программы по 3 лабораторной

Вывод:

Я познакомился с графическим интерфейсом для языка Java, разработал 3 программы с его использованием. Тем самым выполнил 4 лабораторную работу.