



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01/07 Интеллектуальные системы анализа,
обработки и интерпретации больших данных

О Т Ч Е Т

по лабораторной работе № 6

Название: Коллекции

Дисциплина: Языки программирования для работы с большими
данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

Д.В. Авдонин

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2022

Лабораторная работа № 6

Задание:

2. Списки (стеки, очереди) I(1..n) и U(1..n) содержат результаты n измерений тока и напряжения на неизвестном сопротивлении R. Найти приближенное число R методом наименьших квадратов.

Ход работы:

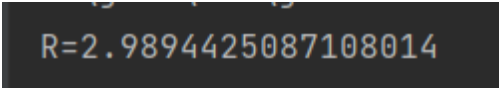
Код программы:

```
public static double mnk(Stack<Double> X, Stack<Double> Y, int n)
{
    double sx2, sxy, buf_u, buf_i;
    sx2 = sxy = 0.0;
    for (int i = 0; i < n; i++)
    {
        buf_u = Y.pop();
        buf_i = X.pop();

        sx2 += buf_i * buf_i;
        sxy += buf_i * buf_u;
    }
    return sxy/sx2;
}

public static void main(String[] args){
    Stack<Double> ii = new Stack<>();
    Stack<Double> uu = new Stack<>();
    for(int i = 1; i <= 20; i++){
        ii.push(((double) i)/10);
    }
    uu.push(0.27);
    uu.push(0.56);
    uu.push(0.9);
    uu.push(1.18);
    uu.push(1.49);
    uu.push(1.79);
    uu.push(2.05);
    uu.push(2.42);
    uu.push(2.68);
    uu.push(3.01);
    uu.push(3.35);
    uu.push(3.56);
    uu.push(3.85);
    uu.push(4.18);
    uu.push(4.48);
    uu.push(4.79);
    uu.push(5.12);
    uu.push(5.45);
    uu.push(5.68);
    uu.push(5.9);

    double q = mnk(ii, uu,20);
    System.out.println("R=" + q);
}
```



R=2.9894425087108014

Рисунок 1 – Результат работы программы

Задание:

3. С использованием множества выполнить попарное суммирование произвольного конечного ряда чисел по следующим правилам: на первом этапе суммируются попарно рядом стоящие числа, на втором этапе суммируются результаты первого этапа и т.д. до тех пор, пока не останется одно число.

Ход работы:

Код программы:

```
public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    System.out.print("Введите количество чисел: ");
    int n, a, c;
    if (scanner.hasNextInt()){
        n = scanner.nextInt();
    } else {
        System.out.println("error");
        n = - 1;
    }
    TreeSet<Integer> integerTreeSet = new TreeSet<>();
    for (int i = 0; i < n; i++){
        System.out.print("Введите число " + i + ": ");
        if (scanner.hasNextInt()){
            integerTreeSet.add(scanner.nextInt());
        } else {
            System.out.println("error");
        }
    }
    System.out.println("Исходное множество: " + integerTreeSet);
    Iterator<Integer> iterator = integerTreeSet.iterator();
    for(int i = 0; i < n-1; i++){
        TreeSet<Integer> buf = new TreeSet<>();
        if(iterator.hasNext()){
            a = iterator.next();
        } else {
            System.out.println("error");
            a = -1;
        }
        while(iterator.hasNext()){
            c = a;
            a = iterator.next();
            buf.add(c + a);
        }
        System.out.println("Промежуточное множество: " + buf);
        iterator = buf.iterator();
    }
}
```

```

Введите количество чисел: 5
Введите число 0: 1
Введите число 1: 2
Введите число 2: 3
Введите число 3: 4
Введите число 4: 5
Исходное множество: [1, 2, 3, 4, 5]
Промужеточное множество: [3, 5, 7, 9]
Промужеточное множество: [8, 12, 16]
Промужеточное множество: [20, 28]
Промужеточное множество: [48]

```

Рисунок 2 – Результат работы программы

Задание:

2. Реализовать класс, моделирующий работу N-местной автостоянки. Машина подъезжает к определенному месту и едет вправо, пока не встретится свободное место. Класс должен поддерживать методы, обслуживающие приезд и отъезд машины.

Ход работы:

Код программы:

```

public class Parking {
    static HashMap<Integer, String> parkingHashMap;

    public Parking(){
        parkingHashMap = new HashMap<>();
        for (int i = 0; i < 10; i++){
            parkingHashMap.put(i + 1, "Vacant");
        }
    }

    public static class Car{
        String name;
        int place_number;

        public Car(String name){
            this.name = name;
            this.place_number = -1;
        }

        public String get_status() {
            if (this.place_number == -1){
                return "not parked";
            } else {
                return "parked";
            }
        }
    }

    public void parking(){
        boolean flag = false;
        List<Integer> indexes = new ArrayList<Integer>(parkingHashMap.keySet());
        for (int index : indexes){
            if (Objects.equals(parkingHashMap.get(index), "Vacant")){

```

```

        parkingHashMap.replace(index, "Engaged");
        this.place_number = index;
        flag = true;
        break;
    }
}
if (flag) {
    System.out.println("Машина припаркована на место " + this.place_number);
} else {
    System.out.println("Все места заняты");
}
}

public void leaving(){
    if (Objects.equals(this.get_status(), "parked")){
        parkingHashMap.replace(this.place_number, "Vacant");
        System.out.println("Машина уехала с места " + this.place_number);
        this.place_number = -1;
    } else {
        System.out.println("Машина не припаркована");
    }
}
}

@Override
public String toString() {
    return "Parking{" +
        "parkingHashMap=" + parkingHashMap +
        '}';
}

public static void main(String[] args){
    Parking parking = new Parking();
    Parking.Car car = new Parking.Car("Моя машина");
    Parking.Car car_1 = new Parking.Car("Моя машина 1");
    Parking.Car car_2 = new Parking.Car("Моя машина 2");
    Parking.Car car_3 = new Parking.Car("Моя машина 3");
    Parking.Car car_4 = new Parking.Car("Моя машина 4");
    Parking.Car car_5 = new Parking.Car("Моя машина 5");
    car.parking();
    car_1.parking();
    car_2.parking();
    car_3.parking();
    car_4.parking();
    car_5.parking();

    car_3.leaving();

    System.out.println(parking);
}

```

```

Машина припаркована на место 1
Машина припаркована на место 2
Машина припаркована на место 3
Машина припаркована на место 4
Машина припаркована на место 5
Машина припаркована на место 6
Машина уехала с места 4
Parking{parkingHashMap={1=Engaged, 2=Engaged, 3=Engaged, 4=Vacant, 5=Engaged, 6=Engaged, 7=Vacant, 8=Vacant, 9=Vacant, 10=Vacant}}

```

Рисунок 3 – Результат работы программы

Задание:

3. Во входном файле хранятся две разреженные матрицы А и В. Построить циклически связанные списки СА и СВ, содержащие ненулевые элементы соответственно матриц А и В. Просматривая списки, вычислить: а) сумму $S = A + B$; б) произведение $P = A * B$.

Ход работы:

Код программы:

```
public class Number {
    private int i;
    private int j;
    private int n;

    public Number(int i, int j, int n){
        this.i = i;
        this.j = j;
        this.n = n;
    }

    public int getN() {
        return n;
    }

    public int getI() {
        return i;
    }

    public int getJ() {
        return j;
    }

    @Override
    public String toString() {
        return "Number{" +
            "i=" + i +
            ", j=" + j +
            ", n=" + n +
            '}';
    }
}

public static void main(String[] args) throws IOException {
    Path in_path = Paths.get("./matrix.txt");
    LinkedList<Number> CA = new LinkedList<>();
    LinkedList<Number> CB = new LinkedList<>();
    LinkedList<Number> S = new LinkedList<>();
    LinkedList<Number> P = new LinkedList<>();

    Scanner scanner = new Scanner(in_path);
    scanner.useDelimiter(System.getProperty("line.separator"));
    String line;
    String[] split_line;
    int ii = 0, jj = 0, n = 0;
    while (scanner.hasNext()){
```

```

    jj = 0;
    line = scanner.next();
    if (line.isEmpty()){
        break;
    }
    split_line = line.split(" ");
    n = split_line.length;
    for (int j = 0; j < n; j++){
        if (!Integer.parseInt(split_line[j]) == 0) {
            CA.add(new Number(ii, jj, Integer.parseInt(split_line[j])));
        }
        jj++;
    }
    ii++;
}
ii = jj = 0;
while (scanner.hasNext()){
    jj = 0;
    line = scanner.next();
    split_line = line.split(" ");
    for (String sym : split_line){
        if (!Integer.parseInt(sym) == 0) {
            CB.add(new Number(ii, jj, Integer.parseInt(sym)));
        }
        jj++;
    }
    ii++;
}

int iter_A = 0, iter_B = 0;
boolean flag_A = true, flag_B = true;
for (int i = 0; i < n; i++){
    for (int j = 0; j < n; j++){
        try{
            CA.get(iter_A);
        } catch (Exception e){
            flag_A = false;
        }
        try {
            CB.get(iter_B);
        } catch (Exception e){
            flag_B = false;
        }
        if ((flag_A && flag_B) && CA.get(iter_A).getI() == i && CA.get(iter_A).getJ() == j &&
            CB.get(iter_B).getI() == i && CB.get(iter_B).getJ() == j){
            S.add(new Number(i, j, CA.get(iter_A).getN() + CB.get(iter_B).getN()));
            iter_A++;
            iter_B++;
        } else if ((flag_A && CA.get(iter_A).getI() == i && CA.get(iter_A).getJ() == j){
            S.add(new Number(i, j, CA.get(iter_A).getN()));
            iter_A++;
        } else if (flag_B && CB.get(iter_B).getI() == i && CB.get(iter_B).getJ() == j){
            S.add(new Number(i, j, CB.get(iter_B).getN()));
            iter_B++;
        } else {
            S.add(new Number(i, j, 0));
        }
    }
}
}

```

```

int multi_buf = 0;
for (int i = 0; i < n; i++){
    for (int j = 0; j < n; j++){

        for (Number value : CA) {
            for (Number number : CB) {
                if (value.getI() == i && value.getJ() == number.getI() &&
                    number.getJ() == j) {
                    multi_buf += value.getN() * number.getN();
                }
            }
        }
        P.add(new Number(i, j, multi_buf));
        multi_buf = 0;
    }
}

System.out.println(CA);
System.out.println(CB);
System.out.println(S);
System.out.println(P);
}

```

```

[Number{i=0, j=0, n=1}, Number{i=1, j=1, n=1}, Number{i=2, j=0, n=1}]
[Number{i=0, j=0, n=1}, Number{i=2, j=0, n=3}]
[Number{i=0, j=0, n=2}, Number{i=0, j=1, n=0}, Number{i=0, j=2, n=0},
 Number{i=1, j=0, n=0}, Number{i=1, j=1, n=1}, Number{i=1, j=2, n=0},
 Number{i=2, j=0, n=4}, Number{i=2, j=1, n=0}, Number{i=2, j=2, n=0}]
[Number{i=0, j=0, n=1}, Number{i=0, j=1, n=0}, Number{i=0, j=2, n=0},
 Number{i=1, j=0, n=0}, Number{i=1, j=1, n=0}, Number{i=1, j=2, n=0},|
 Number{i=2, j=0, n=1}, Number{i=2, j=1, n=0}, Number{i=2, j=2, n=0}]

```

Рисунок 4 – Результат работы программы

Вывод: лабораторная работа выполнена в соответствии с заданием и вариантом.