**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** |  |

**Дисциплина:** Коллекции

**Название:** Языки программирования для работы с большими данными

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ6-22М |  | Д.М. Карасёв\_\_ |
|  | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
| Преподаватель |  | П.В. Степанов | |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными

**Вариант 1:**

1. 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.
2. 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.

**Решение:**

package lr6\_v1\_task2;  
  
import java.util.PriorityQueue;  
import java.util.Queue;  
  
public class lr6\_v1\_task2 {  
 public static double findResistance(Queue<Double> currentQueue, Queue<Double> voltageQueue) throws Exception {  
  
 if(currentQueue.size() != voltageQueue.size()){  
 throw new Exception("Структуры данных имеют разный размер.");  
 }  
  
 int n = currentQueue.size();  
 double sumIU = 0.0;  
 double sumI2 = 0.0;  
 double meanI = 0.0;  
 double meanU = 0.0;  
  
  
 while (!currentQueue.isEmpty() && !voltageQueue.isEmpty()) {  
 double current = currentQueue.poll();  
 double voltage = voltageQueue.poll();  
 sumIU += current \* voltage;  
 sumI2 += Math.*pow*(current, 2);  
 meanI += current;  
 meanU += voltage;  
 }  
  
 meanI /= n;  
 meanU /= n;  
  
  
 double R = (sumIU - n \* meanI \* meanU) / (sumI2 - n \* Math.*pow*(meanI, 2));  
  
  
 return R;  
 }  
 public static void main(String[] args) {  
  
 Queue<Double> current = new PriorityQueue<Double>();  
 Queue<Double> voltage = new PriorityQueue<Double>();  
  
 current.add(1.0);  
 current.add(2.0);  
 current.add(3.0);  
  
 voltage.add(2.0);  
 voltage.add(4.0);  
 voltage.add(6.0);  
  
 try {  
 double Resistance = *findResistance*(current,voltage);  
 System.*out*.println("Resistance: " + Resistance);  
 }catch (Exception e){  
 System.*out*.println(e.getMessage());  
 }  
  
 }  
}

На рисунке 1 представлен результат выполнения.



Рисунок 1 – Результат выполнения

package lr6\_v1\_task3;  
  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.LinkedHashSet;  
import java.util.Set;  
  
public class lr6\_v1\_task3 {  
 public static void main(String[] args){  
  
 Set<Integer> set = new LinkedHashSet<Integer>();  
 Set<Integer> newSet = new LinkedHashSet<Integer>();  
 set.add(1);  
 set.add(2);  
 set.add(3);  
 set.add(4);  
 set.add(5);  
  
 int sum;  
 while (set.size() > 1) {  
 newSet.clear();  
 while (set.size() > 1){  
 Iterator<Integer> iterator = set.iterator();  
 int first = iterator.next();  
 iterator.remove();  
 int second = iterator.next();  
 sum = first + second;  
 newSet.add(sum);  
 System.out.println("Сумма " + first + " и " + second + " = " + sum);  
 }  
 set.clear();  
 set.addAll(newSet);  
 System.out.println(set);  
 }  
 System.out.println("Окончательный результат: " + set.iterator().next());  
 }  
}

На рисунке 2 представлен результат выполнения.

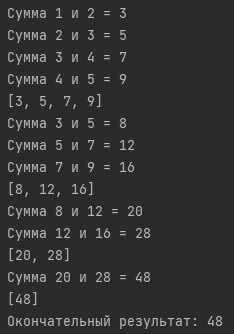


Рисунок 2 – Результат выполнения

**Вариант 2:**

1. На базе коллекций реализовать структуру хранения чисел с поддержкой следующих операций:   
   • добавление/удаление числа;  
   • поиск числа, наиболее близкого к заданному (т.е. модуль разницы минимален).
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.

**Решение:**

package lr6\_v2\_task1;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.NoSuchElementException;  
  
public class NumberStorage {  
 private List<Double> values;  
 public NumberStorage(){  
 values = new ArrayList<Double>();  
 }  
  
 public void add(double number){  
 values.add(number);  
 }  
  
 public double get(int index){  
 return values.get(index);  
 }  
  
 public void remove(int index){  
 values.remove(index);  
 }  
  
 public void remove(double number){  
 values.remove(number);  
 }  
  
 public void setValues(List<Double> values) {  
 this.values = values;  
 }  
  
 public List<Double> getValues() {  
 return values;  
 }  
  
 public double findClosestNumber(double number){  
  
 if (values.isEmpty()) {  
 throw new NoSuchElementException("Список пуст");  
 }  
  
 if (values.contains(number)) {  
 return number;  
 }  
  
 double closestNumber = values.get(0);  
  
 for (Double value : values ){  
 if (Math.*abs*(number - value) < (number - closestNumber)){  
 closestNumber = value;  
 }  
 }  
 return closestNumber;  
 }  
}

package lr6\_v2\_task1;  
  
import java.lang.reflect.Array;  
  
public class lr6\_v2\_task1 {  
 public static void main(String[] args) {  
 NumberStorage numberStorage = new NumberStorage();  
  
 numberStorage.add(10.0);  
 numberStorage.add(20.0);  
 numberStorage.add(30.0);  
 numberStorage.add(40.0);  
 numberStorage.add(50.0);  
 numberStorage.add(60.0);  
  
 System.*out*.println(numberStorage.getValues());  
 System.*out*.println("Ближайшее число к 33:");  
 System.*out*.println(numberStorage.findClosestNumber(33.0));  
 }  
}

На рисунке 3 представлен результат выполнения.

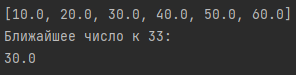


Рисунок 3 – Результат выполнения

package lr6\_v2\_task2;  
  
public class Car {  
 private String carNumber;  
 private int spotNumber;  
 public Car(String carNumber){  
 this.carNumber = carNumber;  
 }  
  
 public int getSpotNumber() {  
 return spotNumber;  
 }  
  
 public void setSpotNumber(int spotNumner) {  
 this.spotNumber = spotNumner;  
 }  
  
 public String getCarNumber() {  
 return carNumber;  
 }  
  
 public void setCarNumber(String carNumber) {  
 this.carNumber = carNumber;  
 }  
}

package lr6\_v2\_task2;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ParkingLot {  
 private int capacity;  
 private List<Car> cars;  
 private List<Integer> freeSpaces;  
  
 public ParkingLot(int capacity) {  
 this.capacity = capacity;  
 this.cars = new ArrayList<>();  
 this.freeSpaces = new ArrayList<>();  
 for (int i = 0; i < capacity; i++) {  
 freeSpaces.add(i);  
 }  
 }  
  
 public int park(Car car) {  
 if (freeSpaces.isEmpty()) {  
 throw new IllegalStateException("No available parking spaces.");  
 }  
 int space = freeSpaces.get(0);  
 freeSpaces.remove(0);  
 car.setSpotNumber(space);  
 cars.add(car);  
 return space;  
 }  
  
 public void unpark(int space) {  
 Car car = null;  
 for (Car c : cars) {  
 if (c.getSpotNumber() == space) {  
 car = c;  
 break;  
 }  
 }  
 if (car == null) {  
 throw new IllegalArgumentException("No car found in space " + space);  
 }  
 freeSpaces.add(space);  
 cars.remove(car);  
 car.setSpotNumber(-1);  
 }  
  
}

package lr6\_v2\_task2;  
  
public class lr6\_v2\_task2 {  
 public static void main(String[] args) {  
 ParkingLot parkingLot = new ParkingLot(1);  
 Car car1 = new Car("ABC123");  
 Car car2 = new Car("DEF456");  
 parkingLot.park(car1);  
 parkingLot.unpark(0);  
 parkingLot.park(car2);  
 }  
}

На рисунке 4 представлен результат выполнения.

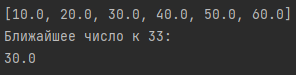


Рисунок 4 – Результат выполнения

**Вывод:** были разработаны методы и классы согласно варианту.