|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 6** |

**Название:**

Коллекции

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | Т.А. Малкина |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

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 com.company;  import java.lang.\*; import java.io.\*; import java.util.\*;  public class Main {   public static double [] mnk(double [] X, double [] Y, int n)  {  double sx2,sx,sy,sxy;  double a,b,d,da,db;  int i;  sx2=sx=sy=sxy=0.0;  for (i=0; i<n; i++)  {  sx+=X[i];  sy+=Y[i];  sx2+=X[i]\*X[i];  sxy+=X[i]\*Y[i];  }   d=((double) n)\*sx2-sx\*sx;  da=((double) n)\*sxy-sx\*sy;  db=sx2\*sy-sx\*sxy;  a=da/d;  b=db/d;  double [] res=new double[2];  res[0]=a;  res[1]=b;  return res;  }    public static void main(String[] args) {  /\*  Вариант 1 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. 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.  \*/  System.out.println("------------------------------");  System.out.println("№ 2");  double U[]={1.5,3,6,9};  double I[]={0.2,0.4,0.8,1.2};  double res[]=mnk(U,I,4);  System.out.println("R="+1.0/res[0]);   System.out.println("------------------------------");  System.out.println("№ 3");  HashSet<Integer> r1 = new HashSet<>();  int k;   for (int i = 0; i<5; i++){  r1.add((int) (Math.random() \* 50 + 5));  }  System.out.println(r1);  Integer[] arr1 = r1.toArray(new Integer[r1.size()]);  ArrayList<Integer> summ\_old = new ArrayList<>();  for (int i = 0; i<4; i++){  summ\_old.add(arr1[i]+arr1[i+1]);  }  System.out.println(summ\_old);   ArrayList<Integer> summ\_new = new ArrayList<>();  k = summ\_old.size();  while (k!=1){  for (int i = 0; i<(k-1); i++){  summ\_new.add(summ\_old.get(i)+summ\_old.get(i+1));  }  summ\_old.clear();  summ\_old.addAll(summ\_new);  System.out.println(summ\_old);  k = summ\_new.size();  summ\_new.clear();  }  System.out.println("Results - "+summ\_old);  } } |

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 com.company;  import java.util.\*;  public class Main {   public static void main(String[] args) {  /\*  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.  \*/  System.out.println("------------------------------");  System.out.println("№ 1");  ArrayList<Integer> number = new ArrayList<>();  int kolvo = 10;  for (int i = 0; i<kolvo; i++){  number.add((int) (Math.random() \* 100));  }  System.out.println("Числа - "+number);  System.out.println("Если хотите добавить число введите - 1. Если хотите удалить число введите - 2. Если хотите найти близкое число введите - 3.");  Scanner in = new Scanner(System.in);  int choo = in.nextInt();  switch (choo){  case 1:  System.out.println("Введите число, которое хотите добавить в коллекцию: ");  int num\_add = in.nextInt();  number.add(num\_add);  System.out.println("Число введено.");  System.out.println("Числа после операции - "+number);  break;  case 2:  System.out.println("Введите число, которое хотите удалить из коллекции: ");  int num\_del = in.nextInt();  number.removeIf(p -> p.equals(num\_del));  System.out.println("Число удалено.");  System.out.println("Числа после операции - "+number);  break;  case 3:  System.out.println("Введите число, которое хотите найти в коллекции: ");  int num\_find = in.nextInt();  Collections.sort(number);  int max =0;  int min =0;  for (int i=0;i<number.size();i++){  if (num\_find >= number.get(i)){  max = i;  }  else {  min = i;  }  }  if(Math.abs(num\_find-number.get(max)) > Math.abs(num\_find-number.get(min))){  System.out.println("Близкое число - " + number.get(min));  } else {  System.out.println("Близкое число - " + number.get(max));  }  break;  default:  System.out.println("Неправильный ввод");  }  System.out.println("------------------------------");  System.out.println("№ 2");  HashMap<Integer,Boolean> place = new HashMap<>();  for (int i = 0; i<kolvo; i++){  place.put(i,false);  }  System.out.println(place);  Parking my\_parking = new Parking(place);  my\_parking.in\_car();  my\_parking.in\_car();  my\_parking.car\_out(1);  my\_parking.in\_car();  System.out.println(my\_parking);  } }  package com.company;   import java.util.\*;  public class Parking {  HashMap<Integer,Boolean> place = new HashMap<>();  public Parking(HashMap<Integer, Boolean> place) {  this.place = place;  }  void in\_car(){  int i = 0;  do {  if (this.place.get(i)){  i++;  } else {  this.place.replace(i,true);  break;  }  }while (!this.place.isEmpty());  System.out.println("Место "+i+" занято");  //System.out.println(this.place);  }  void car\_out (int num){  if (num > this.place.size()){  System.out.println("Такого места нет");  }  else {  this.place.replace(num,false);  System.out.println("Место "+num+" свободно");  //System.out.println(this.place);  }   }    @Override  public String toString() {  return "Parking - "  + place ;  } } |

Ссылка на репозиторий:

https://github.com/Carpediem2026/BigData