|  |  |
| --- | --- |
|  | **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 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **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 | ИУ6-22М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | П.В. Степанов |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2024

**Цель работы:** освоить базовые принципы работы с коллекциями на языке Java.

**Вариант: 1.**

**Задание 1:** Определить множество на основе множества целых чисел. Создать методы для определения пересечения и объединения множеств.

Код решения приведен в листинге 1.

Листинг 1 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Set work example ===\n\n");  MyIntSet set1 = new MyIntSet();  set1.add(1);  set1.add(2);  set1.add(3);  set1.add(4);  MyIntSet set2 = new MyIntSet();  set2.add(3);  set2.add(4);  set2.add(5);  set2.add(6);  System.out.printf("set-1: %s \n", set1);  System.out.printf("set-2: %s \n", set2);  System.out.printf("union: %s \n", set1.union(set2));  System.out.printf("cross: %s \n", set1.cross(set2));  }  }  class MyIntSet extends HashSet<Integer> {  MyIntSet() {  super();  }  public MyIntSet union(MyIntSet set1) {  MyIntSet result = new MyIntSet();  result.addAll(this);  result.addAll(set1);  return result;  }  public MyIntSet cross(MyIntSet set1) {  MyIntSet result = new MyIntSet();  for(Integer elem : this){  if(set1.contains(elem)) {  result.add(elem);  }  }  return result;  }  } |

Результат исполнения программы приведен в листинге 2.

Листинг 2 — результат исполнения программы

|  |
| --- |
| === Set work example ===  set-1: [1, 2, 3, 4]  set-2: [3, 4, 5, 6]  union: [1, 2, 3, 4, 5, 6]  cross: [3, 4] |

**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.

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== I, U, R, least-square work example ===\n\n");  List<Double> i = new ArrayList<>(List.of(1.0, 2.1, 2.9, 4.05, 4.98, 6.03));  List<Double> u = new ArrayList<>(List.of(3.03, 6.1, 8.9, 12.05, 14.98, 18.03));  System.out.printf("U: %s \n", u);  System.out.printf("I: %s \n", i);  Double r = calculateSlope(i, u);  System.out.printf("R: %.5f \n", r);  System.out.printf("diffs: %s\n", calculateDiffs(i, u, r));  }  private static List<Double> calculateDiffs(List<Double> x, List<Double> y, double k){  return IntStream.range(0, x.size())  .mapToObj(i -> Math.abs(x.get(i) \* k - y.get(i)))  .toList();  }  private static Double calculateSlope(List<Double> x, List<Double> y) {  int n = x.size();  Double sx = x.stream().mapToDouble(d -> d).sum();  Double sy = y.stream().mapToDouble(d -> d).sum();  double sxsy =  IntStream.range(0, x.size())  .mapToObj(i -> x.get(i) \* y.get(i))  .toList()  .stream().mapToDouble(d -> d).sum();  double sx2 = x.stream().mapToDouble(d -> d \* d).sum();;  return (double) (n \* sxsy - sx \* sy) / (n \* sx2 - sx \* sx);  }  } |

Результат исполнения программы приведен в листинге 4.

Листинг 4 — результат исполнения программы

|  |
| --- |
| === I, U, R, least-square work example ===  U: [3.03, 6.1, 8.9, 12.05, 14.98, 18.03]  I: [1.0, 2.1, 2.9, 4.05, 4.98, 6.03]  R: 2,99647  diffs: [0.03353314398277352, 0.19258039763617596, 0.21024611755004408, 0.08569076686976551, 0.05759505703421297, 0.0386951417838759] |

**Задание 3:** 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).

Код решения приведен в листинге 5.

Листинг 5 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Custom collection work example ===\n\n");  MyCollection myCollection = new MyCollection();  myCollection.add(1);  myCollection.add(2);  myCollection.add(10);  myCollection.add(11);  System.out.printf("Collection: %s \n", myCollection);  System.out.printf("Closest to 8: %d \n", myCollection.findClosest(8));  }  }  class MyCollection extends HashSet<Integer> {  public Integer findClosest(Integer r){  return this.stream().min(Comparator.comparingInt(a -> Math.abs(r - a))).orElseThrow();  }  } |

Результат исполнения программы приведен в листинге 6.

Листинг 6 — результат исполнения программы

|  |
| --- |
| === Custom collection work example ===  Collection: [1, 2, 10, 11]  Closest to 8: 10 |

**Задание 4:** 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.

Код решения приведен в листинге 7.

Листинг 7 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Parking collection work example ===\n\n");  Parking parking = new Parking(10);  System.out.printf("init: %s\n", Arrays.toString(parking.getData()));  parking.Park();  System.out.printf("=> %s\n", Arrays.toString(parking.getData()));  parking.Park();  System.out.printf("=> %s\n", Arrays.toString(parking.getData()));  parking.Park();  System.out.printf("=> %s\n", Arrays.toString(parking.getData()));  parking.Park();  System.out.printf("=> %s\n", Arrays.toString(parking.getData()));  parking.DriveAway(1);  System.out.printf("<=[1] %s\n", Arrays.toString(parking.getData()));  parking.Park();  System.out.printf("=> %s\n", Arrays.toString(parking.getData()));  }  }  class Parking {  private boolean[] data;  Parking(int n) {  data = new boolean[n];  Arrays.fill(data, false);  }  public boolean[] getData() {  return data;  }  public boolean Park(){  for (int i = 0; i < data.length; i++) {  if(!data[i]){  data[i] = true;  return true;  }  }  return false;  }  public void DriveAway(int position){  data[position] = false;  }  } |

Результат исполнения программы приведен в листинге 8.

Листинг 8 — результат исполнения программы

|  |
| --- |
| === Parking collection work example ===  init: [false, false, false, false, false, false, false, false, false, false]  => [true, false, false, false, false, false, false, false, false, false]  => [true, true, false, false, false, false, false, false, false, false]  => [true, true, true, false, false, false, false, false, false, false]  => [true, true, true, true, false, false, false, false, false, false]  <=[1] [true, false, true, true, false, false, false, false, false, false]  => [true, true, true, true, false, false, false, false, false, false] |

**Вывод:** в ходе лабораторной работы были освоены базовые принципы работы с коллекциями на языке Java.