

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"МИРЭА - Российский технологический университет"РТУ МИРЭА |

**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**ПРАКТИЧЕСКАЯ РАБОТА №1**

**Метод Парето**

**по дисциплине**

**«Теория принятия решений»**

Студент группы: ИКБО-04-19 Сюртуков З.А *(Фамилия студента)*

Руководитель работы Железняк Л.М.\_

*(Фамилия преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2021

**Выбор планеты для проживания по методу Парето**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Вариант | Температура  (-) | Плотность  (+) | Вода  (+) | Вулканы  (-) | Атмосфера  (+) |
| 1 | Меркурий | 427 | 5.43 | 0 | 423 | 0 |
| 2 | Венера | 600 | 5.42 | 0 | 1023 | 90 |
| 3 | Земля | 20 | 5.5 | 5000 | 56 | 1 |
| 4 | Марс | -40 | 5.4 | 5 | 100 | 2 |
| 5 | Юпитер | -108 | 0.6 | 300 | 0 | 3000 |
| 6 | Сатурн | -109 | 0.7 | 300 | 0 | 3000 |
| 7 | Уран | -100 | 0.8 | 100 | 0 | 2000 |
| 8 | Нептун | -150 | 0.8 | 200 | 0 | 2500 |
| 9 | Плутон | -150 | 5.1 | 0 | 40 | 0.5 |

**Метод Парето**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | X | X | X | X | X | X | X | X | X |
| 2 | H | X | X | X | X | X | X | X | X |
| 3 | A3 | H | X | X | X | X | X | X | X |
| 4 | H | H | H | X | X | X | X | X | X |
| 5 | H | H | H | H | X | X | X | X | X |
| 6 | H | H | H | H | A6 | X | X | X | X |
| 7 | H | H | H | H | H | H | X | X | X |
| 8 | H | H | H | H | H | H | A8 | X | X |
| 9 | H | H | H | H | H | H | H | H | X |

**Программный код на языке Java для реализации метода Парето**

Solution.java

import First.Alternative;  
import First.Pareto;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
public class Solution {  
 public static void main(String[] args) throws IOException {  
 List<Alternative> alternatives = new ArrayList<>();  
 List<String> names = new ArrayList<>();  
 List<Boolean> heights = new ArrayList<>();  
 names.add("Температура");  
 names.add("Плотность");  
 names.add("Вода");  
 names.add("Вулканы");  
 names.add("Атмосфера");  
 heights.add(false);  
 heights.add(true);  
 heights.add(true);  
 heights.add(false);  
 heights.add(true);  
 List<Double> counts = new ArrayList<>();  
 counts.add(427.);  
 counts.add(5.43);  
 counts.add(0.);  
 counts.add(423.);  
 counts.add(0.);  
 alternatives.add(new Alternative("Меркурий", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(600.);  
 counts.add(5.42);  
 counts.add(0.);  
 counts.add(1023.);  
 counts.add(90.);  
 alternatives.add(new Alternative("Венера", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(20.);  
 counts.add(5.5);  
 counts.add(5000.);  
 counts.add(56.);  
 counts.add(1.);  
 alternatives.add(new Alternative("Земля", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(-40.);  
 counts.add(5.4);  
 counts.add(5.);  
 counts.add(100.);  
 counts.add(2.);  
 alternatives.add(new Alternative("Марс", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(-108.);  
 counts.add(0.6);  
 counts.add(300.);  
 counts.add(0.);  
 counts.add(3000.);  
 alternatives.add(new Alternative("Юпитер", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(-109.);  
 counts.add(0.7);  
 counts.add(300.);  
 counts.add(0.);  
 counts.add(3000.);  
 alternatives.add(new Alternative("Сатурн", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(-100.);  
 counts.add(0.8);  
 counts.add(100.);  
 counts.add(0.);  
 counts.add(2000.);  
 alternatives.add(new Alternative("Уран", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(-150.);  
 counts.add(0.8);  
 counts.add(200.);  
 counts.add(0.);  
 counts.add(2500.);  
 alternatives.add(new Alternative("Нептун", names, heights, counts));  
 counts = new ArrayList<>();  
 counts.add(-150.);  
 counts.add(5.1);  
 counts.add(0.);  
 counts.add(40.);  
 counts.add(0.5);  
 alternatives.add(new Alternative("Плутон", names, heights, counts));  
 Pareto.*out*(alternatives);  
 System.*out*.println();  
 Pareto.*paretoMethod*(alternatives);  
 System.*out*.println();  
 Pareto.*out*(alternatives);  
 }  
}

Alternative.java

package First;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Alternative implements Comparable<Alternative> {  
 private final String name;  
 private final List<Criteria> criterias = new ArrayList<>();  
  
 public Alternative(String name, List<String> nCriterias, List<Boolean> bCriterias, List<Double> iCriterias) {  
 this.name = name;  
 for (int i = 0; i < nCriterias.size(); i++)  
 criterias.add(new Criteria(nCriterias.get(i), iCriterias.get(i), bCriterias.get(i)));  
 }  
  
 public int compareTo(Alternative o) {  
 int res = 0;  
 for (int i = 0; i < criterias.size(); i++) {  
 if (criterias.get(i).height && criterias.get(i).count > o.criterias.get(i).count || !criterias.get(i).height && criterias.get(i).count < o.criterias.get(i).count) {  
 if (res < 0)  
 return 0;  
 res++;  
 } else if (criterias.get(i).height && criterias.get(i).count < o.criterias.get(i).count || !criterias.get(i).height && criterias.get(i).count > o.criterias.get(i).count)  
 if (res > 0)  
 return 0;  
 res--;  
 }  
 return res;  
 }  
  
 public String toString() {  
 StringBuilder builder = new StringBuilder("Вариант ").append(name).append(":");  
 for (Criteria criteria : criterias)  
 builder.append(", ").append(criteria.criteriaName).append(" = ").append(criteria.count);  
 return builder.toString();  
 }  
  
  
 private static class Criteria {  
 private final String criteriaName;  
 private final double count;  
 private final boolean height;  
  
 public Criteria(String criteriaName, double count, boolean height) {  
 this.criteriaName = criteriaName;  
 this.count = count;  
 this.height = height;  
 }  
 }  
}

Pareto.java

package First;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.List;  
  
public class Pareto {  
 public static List<Alternative> in() throws IOException {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 List<Alternative> alternatives = new ArrayList<>();  
 System.*out*.println("Введите количество критериев");  
 int n = Integer.*parseInt*(reader.readLine());  
 List<String> names = new ArrayList<>();  
 List<Boolean> heights = new ArrayList<>();  
 for (int i = 0; i < n; i++) {  
 System.*out*.println("Введите имя для критерия");  
 names.add(reader.readLine());  
 System.*out*.println("Критерий положительный или отрицательный(true/false)?");  
 heights.add(Boolean.*parseBoolean*(reader.readLine()));  
 }  
 System.*out*.println("Введите количество альтернатив");  
 n = Integer.*parseInt*(reader.readLine());  
 for (int i = 0; i < n; i++) {  
 System.*out*.println("Введите название варианта");  
 String name = reader.readLine();  
 List<Double> counts = new ArrayList<>();  
 for (String na : names) {  
 System.*out*.println("Введите значение для критерия " + na);  
 counts.add(Double.*parseDouble*(reader.readLine()));  
 }  
 alternatives.add(new Alternative(name, names, heights, counts));  
 }  
 return alternatives;  
 }  
  
 public static void paretoMethod(List<Alternative> alternatives) {  
 label:  
 for (int i = 0; i < alternatives.size();) {  
 for (int j = i + 1; j < alternatives.size();) {  
 int res = alternatives.get(i).compareTo(alternatives.get(j));  
 if (res < 0) {  
 System.*out*.println(alternatives.get(j) + " доминирует над " + alternatives.get(i));  
 alternatives.remove(i);  
 continue label;  
 }  
 else if (res > 0) {  
 System.*out*.println(alternatives.get(i) + " доминирует над " + alternatives.get(j));  
 alternatives.remove(j);  
 continue;  
 }  
 j++;  
 }  
 i++;  
 }  
 }  
  
 public static void out(List<Alternative> alternatives) {  
 for (Alternative alternative : alternatives)  
 System.*out*.println(alternative);  
 }  
}

**Скриншот результата работы программы**

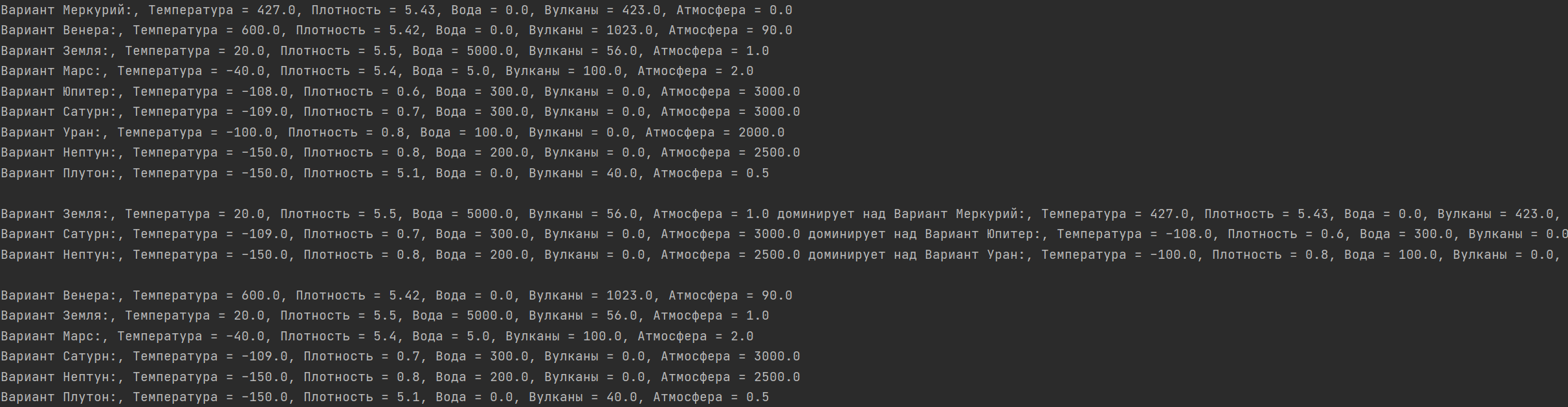
****

Рисунок 1. Вывод программы “метод Парето”