|  |  |
| --- | --- |
| 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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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 №** | 3 |

**Название:**

Классы. Наследование. Полиморфизм

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | А.А. Морозова |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

**Цель работы** — ознакомление механизмами создания классов, наследования и полиморфизма в языке Java.

1. Определить класс Вектор в R3. Реализовать методы для проверки векторов на ортогональность, проверки пересечения не ортогональных векторов, сравнения векторов. Создать массив из m объектов. Определить, какие из векторов компланарны.

Часть кода задания приведена в листинге 1, результат выполнения – на рисунке 1.

Листинг 1 – Метод, проверяющий, компланарны ли векторы

|  |
| --- |
| public static boolean areCoplanar(Vector[] vectors) {  if (vectors.length < 3) {  return true;  }  Vector v1 = vectors[0];  Vector v2 = vectors[1];  Vector normal = v1.crossProduct(v2).normalize();  for (int i = 2; i < vectors.length; i++) {  Vector v = vectors[i];  if (normal.dotProduct(v.crossProduct(v1)) != 0) {  return false;  }  }  return true;  } |

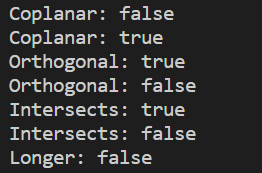


Рисунок 1 – Результат выполнения операций над различными векторами

1. Определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы

Часть кода задания приведена в листинге 2, результат выполнения – на рисунке 2.

Листинг 2 – Умножение матрицПервое слово с минимальным числом различных символов

|  |
| --- |
| public Matrix multiplication(Matrix m) {  int[][] arr = new int[this.n][this.n];  for (int i = 0; i < this.n; i++) {  for (int j = 0; j < this.n; j++) {  for (int k = 0; k < this.n; k++) {  arr[i][j] += this.arr[i][k] \* m.arr[k][j];  }  }  }  return new Matrix(arr);  } |

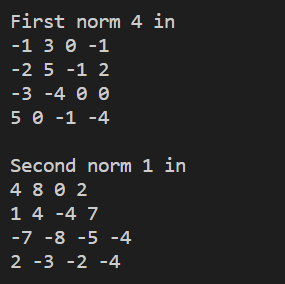


Рисунок 2 – Нахождение норм матриц

1. Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.
   1. Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести:
      1. список пациентов, имеющих данный диагноз;
      2. список пациентов, номер медицинской карты у которых находится в заданном интервале.

Часть кода задания приведена в листинге 3, результат выполнения – на рисунке 3.

Листинг 3 – Переопределение метода toString для класса Patient

|  |
| --- |
| @Override  public String toString() {  return "Patient{" +  "id=" + id + ", surname=" + surname + ", name=" + name + ", patronymic="  + patronymic + ", address=" + address + ", phoneNumber=" + phoneNumber +  ", medicalCardId=" + medicalCardId + ", diagnosis=" + diagnosis + '}';S |

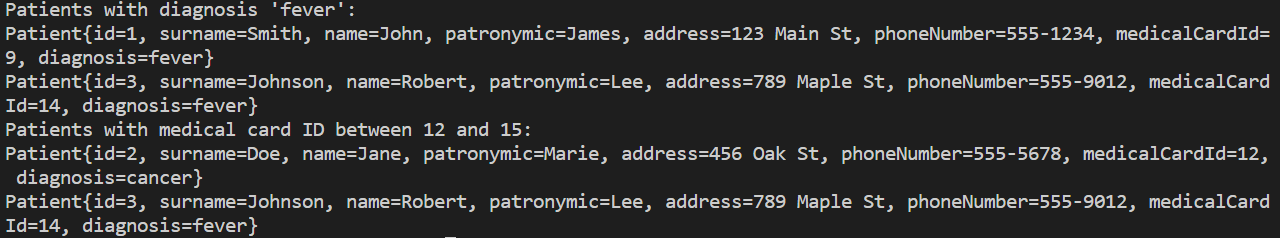


Рисунок 3 – Вывод по условиям задачи

* 1. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: a) список абитуриентов, имеющих неудовлетворительные оценки; b) список абитуриентов, средний балл у которых выше заданного; c) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

Часть кода задания приведена в листинге 4, результат выполнения – на рисунке 4.

Листинг 4 – Три абитуриента с самыми высокими средними оценками

|  |
| --- |
| int n = 3;  Abiturient[] sortedAbiturients = Arrays.copyOf(abiturients, abiturients.length);  Arrays.sort(sortedAbiturients, (a1, a2) -> Float.compare(a2.getAverageScore(), a1.getAverageScore()));  System.out.println("List of " + n + " abiturients with the highest average score:");  for (int i = 0; i < n && i < sortedAbiturients.length; i++) {  System.out.println(sortedAbiturients[i]);  } |

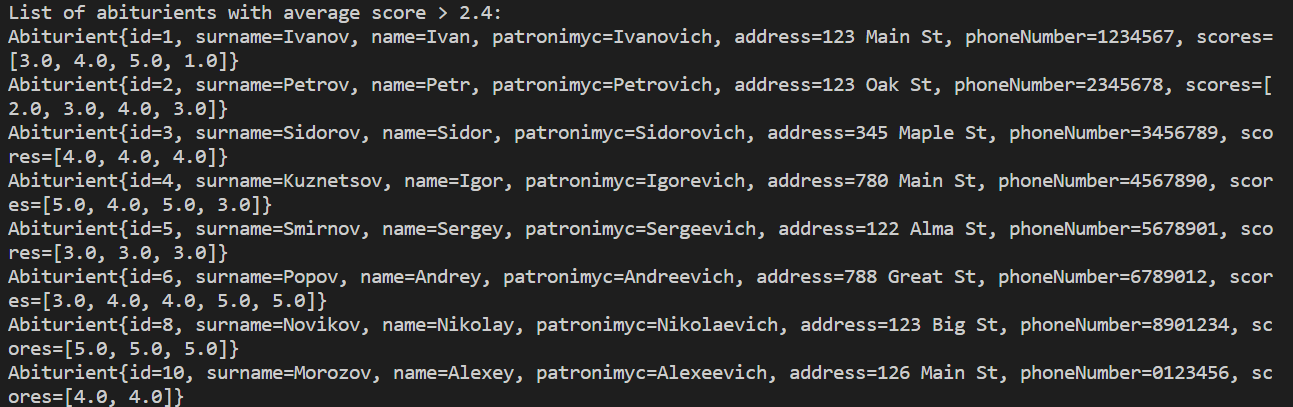


Рисунок 4 – Полный список абитуриентов, имеющих полупроходной балл

Полный код заданий размещен в репозитории по ссылке - https://github.com/moroz-matros/BDL.

**Вывод** — в ходе работы были получены навыки работы с арифметическими операциями в языке Java.