**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «Электронные вычислительные машины»

**Отчет о лабораторной работе № 3**

по теме

«Классы и методы»

**Выполнили:**

Студенты гр. 045

Бригады №10

Анохин В.А.

Вашкулатов Н.А.

**Проверил:**

Асс. Бастрычкин А.С.

Цель работы:изучение принципов ООП, приобретение навыков работы с классами и методами в Java-программах.

Ход работы

Задание №1

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы **set***Тип***()**, **get***Тип***()**, **toString()**. Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. В каждом классе, обладающем информацией, должно быть объявлено несколько конструкторов.

**Train**: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс).

Вывести:

a) список поездов, следующих до заданного пункта назначения;

b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

c) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

Код программы:

Перечисления

public enum SeatTypes {  
 COMMON, COMPARTMENT, RESERVED\_SEAT, LUXURY;  
}

public enum Cities {  
 MOSCOW, RYAZAN, ROSTOV, SOCHI, GROZNIY, KISLOVODSK, KRASNODAR;  
}

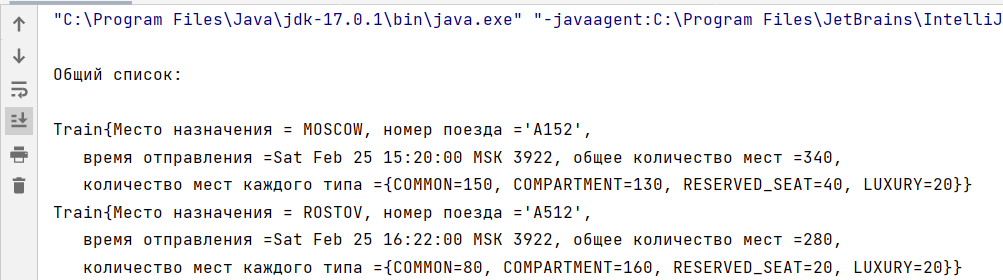
Класс Train

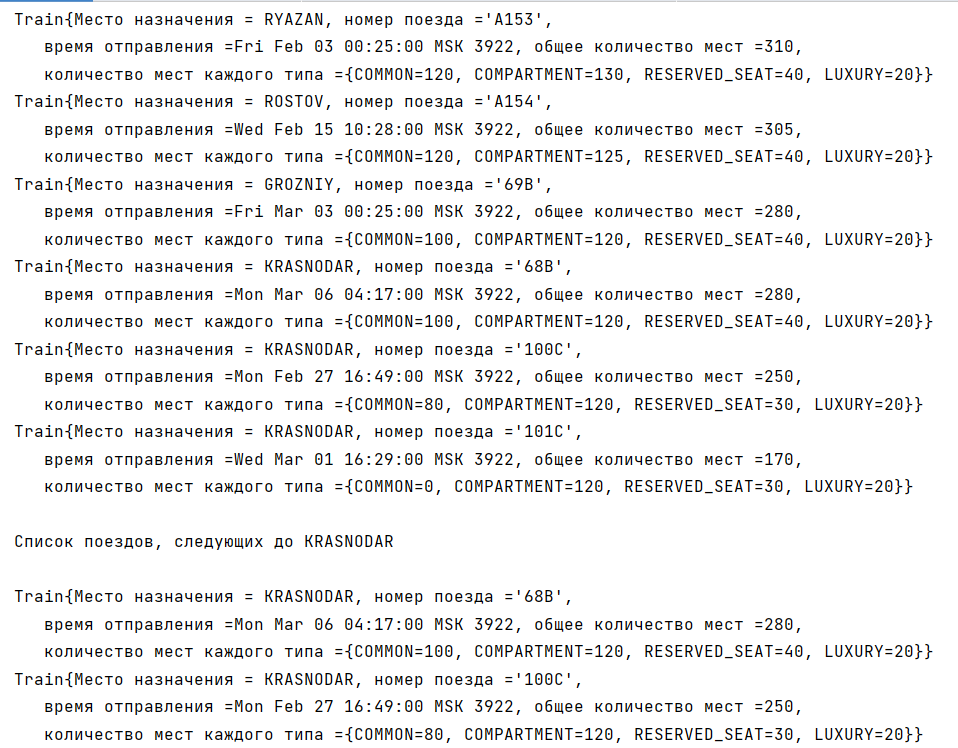
import java.util.Date;  
import java.util.EnumMap;  
  
public class Train {  
 private Cities destination;  
 private String trainNumber;  
 private Date departureTime;  
 private int summarySeatsCount;  
 private EnumMap<SeatTypes, Integer> allSeatsCountByTypes;  
  
 private void setSummarySeatsCount() {  
 for (SeatTypes type: SeatTypes.values()) {  
 summarySeatsCount += allSeatsCountByTypes.get(type);  
 }  
 }  
 public int getSummarySeatsCount() {  
 return summarySeatsCount;  
 }  
  
 public Train(Cities destination, String trainNumber, Date departureTime, int commonSeatsCount, int compartmentSeatsCount, int reservedSeatsCount,int luxurySeatsCount ) {  
 this.destination = destination;  
 this.trainNumber = trainNumber;  
 this.departureTime = departureTime;  
 this.allSeatsCountByTypes = new EnumMap<SeatTypes, Integer>(SeatTypes.class);  
 allSeatsCountByTypes.put(SeatTypes.COMMON, commonSeatsCount);  
 allSeatsCountByTypes.put(SeatTypes.COMPARTMENT, compartmentSeatsCount);  
 allSeatsCountByTypes.put(SeatTypes.RESERVED\_SEAT, reservedSeatsCount);  
 allSeatsCountByTypes.put(SeatTypes.LUXURY, luxurySeatsCount);  
 setSummarySeatsCount();  
 }  
  
 public Cities getDestination() {  
 return destination;  
 }  
  
 public void setDestination(Cities destination) {  
 this.destination = destination;  
 }  
  
 public String getTrainNumber() {  
 return trainNumber;  
 }  
  
 public void setTrainNumber(String trainNumber) {  
 this.trainNumber = trainNumber;  
 }  
  
 public Date getDepartureTime() {  
 return departureTime;  
 }  
  
 public void setDepartureTime(Date departureTime) {  
 this.departureTime = departureTime;  
 }  
  
 public void setSeatsCountByType(SeatTypes type, int newCount){  
 allSeatsCountByTypes.put(type, newCount);  
 setSummarySeatsCount();  
 }  
 public int getSeatsCountByType(SeatTypes type){  
 return allSeatsCountByTypes.get(type);  
 }  
  
 @Override  
 public String toString() {  
 return "Train{" +  
 "Место назначения = " + destination +  
 ", номер поезда ='" + trainNumber + "', \n" +  
 " время отправления =" + departureTime +  
 ", общее количество мест =" + summarySeatsCount +", \n" +  
 " количество мест каждого типа =" + allSeatsCountByTypes +  
 '}';  
 }  
}

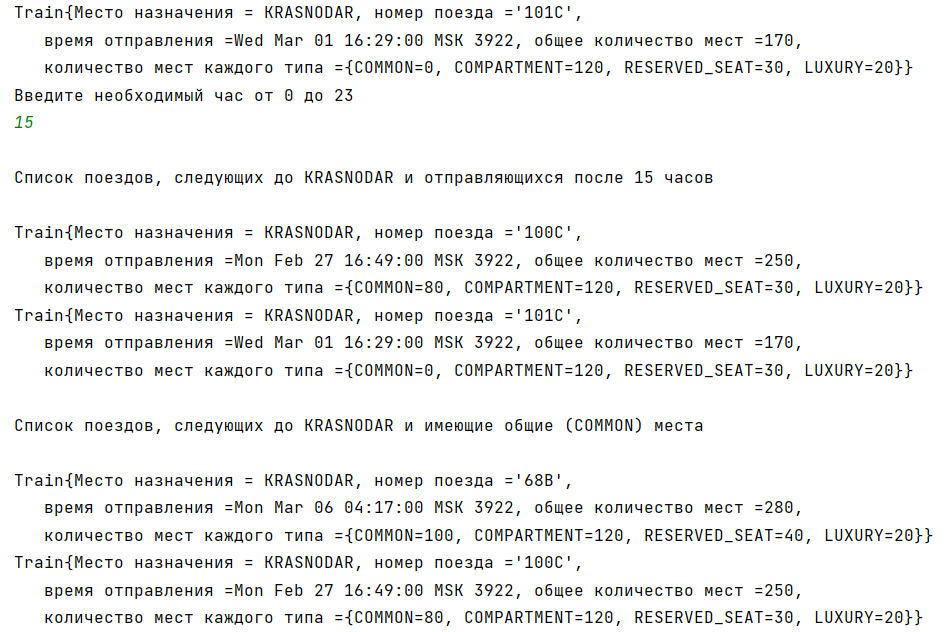
Основная программа

import java.sql.SQLOutput;  
import java.util.\*;  
  
public class App {  
 final static int MIN\_HOUR = 0;  
 final static int MAX\_HOUR = 23;  
 public static void main(String[] args) {  
 ArrayList<Train> trains = new ArrayList(){  
 {  
 add(new Train(Cities.MOSCOW, "A152", new Date(2022, Calendar.FEBRUARY,25,15,20), 150,130,40,20));  
 add(new Train(Cities.ROSTOV, "A512", new Date(2022,Calendar.FEBRUARY,25,16,22),80,160,20,20));  
 add(new Train(Cities.RYAZAN,"A153", new Date(2022,Calendar.FEBRUARY,3,0,25),120,130,40,20));  
 add(new Train(Cities.ROSTOV,"A154", new Date(2022,Calendar.FEBRUARY,15,10,28),120,125,40,20));  
 add(new Train(Cities.GROZNIY,"69B", new Date(2022,Calendar.MARCH,3,0,25),100,120,40,20));  
 add(new Train(Cities.KRASNODAR,"68B", new Date(2022,Calendar.MARCH,6,4,17),100,120,40,20));  
 add( new Train(Cities.KRASNODAR,"100C", new Date(2022,Calendar.FEBRUARY,27,16,49),80,120,30,20));  
 add( new Train(Cities.KRASNODAR,"101C", new Date(2022,Calendar.FEBRUARY,29,16,29),0,120,30,20));  
  
 }  
 };  
 System.out.println("\nОбщий список:\n");  
 trains.forEach(System.out::println);  
 //Фильтрация по городу  
 Cities filterDestination = Cities.KRASNODAR;  
 System.out.println("\nСписок поездов, следующих до "+ filterDestination + " \n");  
 trains.stream()  
 .filter(x -> x.getDestination() == filterDestination)  
 .forEach(System.out::println);  
 //Считываю часы для фильрации  
 Scanner sc = new Scanner(System.in);  
 int filterHour = 255;  
 while ((filterHour < MIN\_HOUR) || (filterHour > MAX\_HOUR)) {  
 System.out.println("Введите необходимый час от 0 до 23");  
 try {  
 filterHour = Integer.parseInt(sc.nextLine());  
 } catch (NumberFormatException e){ // Если была введена буква, выполнение не завершится  
 System.err.println("Введи число");  
 }  
 }  
  
 //Фильтрация по городу и по часу отправления  
 System.out.println("\nСписок поездов, следующих до "+ filterDestination + " и отправляющихся после " + filterHour + " часов "+"\n");  
 int finalFilterHour = filterHour; // Компилятор требовал, чтобы переменная была final в лямбда выражении  
 trains.stream()  
 .filter(x -> (x.getDepartureTime().getHours() >= finalFilterHour))  
 .filter(x -> (x.getDestination() == filterDestination))  
 .forEach(System.out::println);  
  
 //Фильтрация по городу и по наличию общих мест  
 System.out.println("\nСписок поездов, следующих до "+ filterDestination + " и имеющие общие (COMMON) места"+"\n");  
 trains.stream()  
 .filter(x->(x.getDestination()) == filterDestination)  
 .filter(x -> (x.getSeatsCountByType(SeatTypes.COMMON) > 0 ))  
 .forEach(System.out::println);  
  
 }  
}

Окно вывода:







Задание №2

Реализовать методы сложения, вычитания, умножения и деления объектов.

Определить класс **Дробь (Рациональная Дробь)** в виде пары чисел m и n. Объявить и инициализировать массив из k дробей, ввести/вывести значения для массива дробей. Создать массив/список/множество объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента.

Код программы:

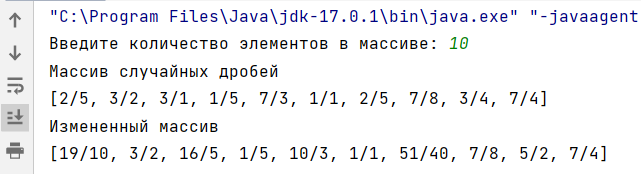
Класс Fraction

public class Fraction {  
 int m, n;  
  
 Fraction(int numerator, int denominator){  
 int divider = nod(numerator, denominator);  
 m = numerator/divider;  
 n = denominator/divider;  
 }  
  
 //Алгоритм Евклида для нахождения НОД  
 private int nod(int a, int b) {  
 a = Math.abs(a);  
 b = Math.abs(b);  
 while (b !=0) {  
 int tmp = a % b;  
 a = b;  
 b = tmp;  
 }  
 return a;  
 }  
  
 private Fraction reduce(int numerator, int denominator){  
 int divider = nod(numerator, denominator);  
 return new Fraction(numerator/divider, denominator/divider);  
 }  
  
 public Fraction add(Fraction secondFraction){  
 int numerator = this.m \* secondFraction.n + secondFraction.m \* this.n;  
 int denominator = this.n \* secondFraction.n;  
 return reduce(numerator, denominator);  
 }  
  
 public Fraction sub(Fraction secondFraction){  
 int numerator = this.m \* secondFraction.n - secondFraction.m \* this.n;  
 int denominator = this.n \* secondFraction.n;  
 return reduce(numerator, denominator);  
 }  
  
 public Fraction mul(Fraction secondFraction){  
 int numerator = this.m \* secondFraction.m;  
 int denominator = this.n \* secondFraction.n;  
 return reduce(numerator, denominator);  
 }  
  
 public Fraction div(Fraction secondFraction){  
 int numerator = this.m \* secondFraction.n;  
 int denominator = this.n \* secondFraction.m;  
 return reduce(numerator, denominator);  
 }  
  
 @Override  
 public String toString() {  
 return m + "/" + n;  
 }  
}

Основная программа

import java.util.Arrays;  
import java.util.Random;  
import java.util.Scanner;  
  
public class App {  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.in);  
 Random random = new Random();  
 System.out.print("Введите количество элементов в массиве: ");  
 int count = scan.nextInt();  
 Fraction[] array = new Fraction[count];  
 for (int i=0; i<count;++i){  
 array[i]=new Fraction(random.nextInt(9)+1, random.nextInt(9)+1);  
 }  
 System.out.println("Массив случайных дробей ");  
 System.out.println(Arrays.toString(array));  
 adder(array);  
 System.out.println("Измененный массив");  
 System.out.println(Arrays.toString(array));  
 }  
  
 public static void adder(Fraction [] arr){  
 for (int i = 0; i< arr.length-1; i+=2){  
 arr[i] = arr[i].add(arr[i+1]);  
 }  
 }  
}

Окно вывода:



Задание №3

Определить класс **Вектор**. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Объявить массив объектов. Написать метод, который для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.

Код программы:

Класс Vector

public class Vector {  
 private double coordX;  
 private double coordY;  
 private double coordZ;  
  
 public double getABSofVector(){  
 return Math.sqrt(coordX \* coordX + coordY \* coordY + coordZ \* coordZ);  
 };  
  
 //Конструктор  
 public Vector(double coordX, double coordY, double coordZ) {  
 this.coordX = coordX;  
 this.coordY = coordY;  
 this.coordZ = coordZ;  
 }  
  
 @Override  
 public String toString() {  
 return "Vector{" +  
 "coordX=" + coordX +  
 ", coordY=" + coordY +  
 ", coordZ=" + coordZ +  
 '}';  
 }  
  
 //геттеры и сеттеры  
 public double getCoordX() {  
 return coordX;  
 }  
  
 public void setCoordX(double coordX) {  
 this.coordX = coordX;  
 }  
  
 public double getCoordY() {  
 return coordY;  
 }  
  
 public void setCoordY(double coordY) {  
 this.coordY = coordY;  
 }  
  
 public double getCoordZ() {  
 return coordZ;  
 }  
  
 public void setCoordZ(double coordZ) {  
 this.coordZ = coordZ;  
 }  
}

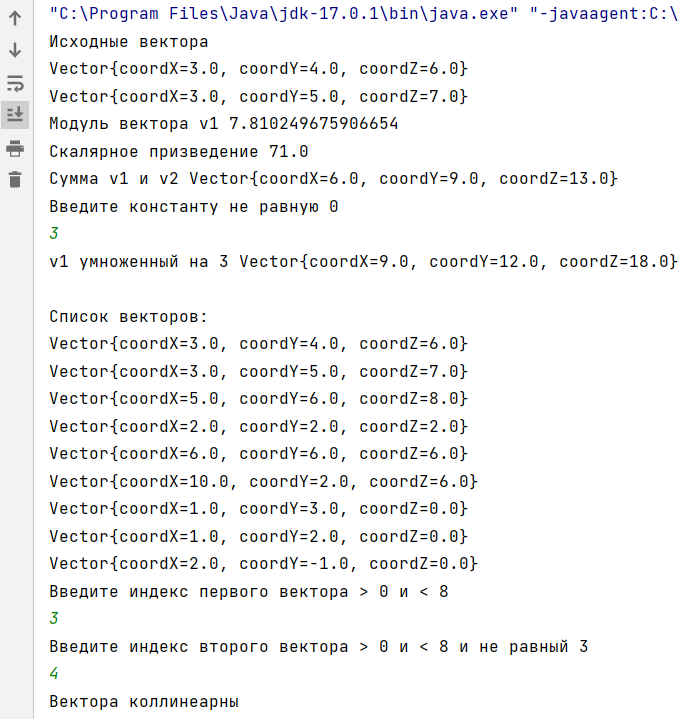
Класс VectorsAction

public class VectorsAction {  
 public static double getScalarMultiplication(Vector v1, Vector v2){  
 return v1.getCoordX() \* v2.getCoordX() + v1.getCoordY() \* v2.getCoordY() + v1.getCoordZ() \* v2.getCoordZ();  
 }  
  
 public static Vector sumTwoVectors(Vector v1, Vector v2){  
 return new Vector((v1.getCoordX() + v2.getCoordX()), (v1.getCoordY()+ v2.getCoordY()), (v1.getCoordZ() + v2.getCoordZ()));  
 }  
  
 public static Vector multiplyingByConst(Vector v1, int constant ){  
 return new Vector(v1.getCoordX() \* constant, v1.getCoordY() \* constant, v1.getCoordZ() \* constant);  
 }  
  
 public static boolean isCollinear(Vector v1, Vector v2){  
 double m = v1.getCoordX() / v2.getCoordX();  
 return (m == v1.getCoordZ()/ v2.getCoordZ()) && (m == v1.getCoordY()/ v2.getCoordY());  
 }  
  
 public static boolean isOrthogonal(Vector v1, Vector v2){  
 return (getScalarMultiplication(v1,v2) == 0);  
 }  
}

Основная программа

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class App2 {  
 public static void main(String[] args) {  
 Vector v1 = new Vector(3,4,6);  
 Vector v2 = new Vector(3,5,7);  
 System.out.println("Исходные вектора \n" + v1 + "\n" + v2);  
 System.out.println("Модуль вектора v1 " + v1.getABSofVector());  
 System.out.println("Скалярное призведение " + VectorsAction.getScalarMultiplication(v1,v2));  
 System.out.println("Сумма v1 и v2 " + VectorsAction.sumTwoVectors(v1,v2));  
 Scanner sc = new Scanner(System.in);  
 int constant = 0;  
 while (constant == 0) {  
 System.out.println("Введите константу не равную 0");  
 try {  
 constant = Integer.parseInt(sc.nextLine());  
 } catch (NumberFormatException e){ // Если была введена буква, выполнение не завершится  
 System.err.println("Введите число");  
 }  
 }  
 System.out.println("v1 умноженный на " + constant + " " + VectorsAction.multiplyingByConst(v1, constant));  
  
 ArrayList<Vector> vectors = new ArrayList<>(){  
 {  
 add(v1);  
 add(v2);  
 add(new Vector(5,6,8));  
 add(new Vector(2,2,2));  
 add(new Vector(6,6,6));  
 add(new Vector(10,2,6));  
 add(new Vector(1,3,0));  
 add(new Vector(1,2,0));  
 add(new Vector(2,-1,0));  
  
 }  
 };  
 System.out.println("\nСписок векторов: ");  
 vectors.forEach(System.out::println);  
 int indexOfFirstVector = 0;  
 int indexOfSecondVector = 0;  
  
 while (indexOfFirstVector == 0) {  
 System.out.println("Введите индекс первого вектора > 0 и < " + (vectors.size() -1) );  
 try {  
 indexOfFirstVector = Integer.parseInt(sc.nextLine());  
 if ((indexOfFirstVector < 0) || (indexOfFirstVector >= vectors.size())){  
 indexOfFirstVector = 0;  
 }  
 } catch (NumberFormatException e){ // Если была введена буква, выполнение не завершится  
 System.err.println("Введите число");  
 }  
 }  
  
 while (indexOfSecondVector == 0) {  
 System.out.println("Введите индекс второго вектора > 0 и < " + (vectors.size() -1) + " и не равный " + indexOfFirstVector );  
 try {  
 indexOfSecondVector = Integer.parseInt(sc.nextLine());  
 if ((indexOfSecondVector < 0) || (indexOfSecondVector >= vectors.size()) || (indexOfSecondVector == indexOfFirstVector)){  
 indexOfSecondVector = 0;  
 }  
 } catch (NumberFormatException e){ // Если была введена буква, выполнение не завершится  
 System.err.println("Введите число");  
 }  
 }  
  
 if (VectorsAction.isCollinear(vectors.get(indexOfFirstVector), vectors.get(indexOfSecondVector))){ //3 и 4 коллиниарны  
 System.out.println("Вектора коллинеарны");  
 }  
  
 if(VectorsAction.isOrthogonal(vectors.get(indexOfFirstVector), vectors.get(indexOfSecondVector))){ // 7 и 8 ортоганальны  
 System.out.println("Вектора ортоганальны");  
 }  
 }  
}

Окно вывода:



**Вывод:** мы изучили принципы ООП, приобрели навыки работы с классами и методами в Java-программах.