|  |  |
| --- | --- |
|  | **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**

**Отчет**

**по лабораторной работе № 3**

**Название:** Классы, наследование, полиморфизм

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

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

Москва, 2023

1. ****Введение и задание****

**Целью лабораторной работы является изучение ООП в Java – классы, наследование и полиморфизм.**

**Номер в списке – 10.**

**Вариант 1.**

**Задание 10.** Определить класс Булева матрица (BoolMatrix) размерности (n x m). Класс должен содержать несколько конструкторов. Реализовать методы для логического сложения (дизъюнкции), умножения и инверсии матриц. Реализовать методы для подсчета числа единиц в матрице и упорядочения строк в лексикографическом порядке**.**

**Задание 1.** Определить класс Вектор размерности n. Реализовать методы сложения, вычитания, умножения, инкремента, декремента, индексирования. Определить массив из m объектов. Каждую из пар векторов передать в методы, возвращающие их скалярное произведение и длины. Вычислить и вывести углы между векторами**.**

**Вариант 2.**

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.

**Задание 10.** Train: Пункт назначения, Номер поезда, Время отправления, Число мест (общих, купе, плацкарт, люкс). Создать массив объектов. 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**.**

**Задание 1.** Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс, Группа. Создать массив объектов. Вывести: a) список студентов заданного факультета; b) списки студентов для каждого факультета и курса; c) список студентов, родившихся после заданного года; d) список учебной группы.

Вариант 3.

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

**Задание 10.** Создать объект класса Год, используя классы Месяц, День. Методы: задать дату, вывести на консоль день недели по заданной дате, рассчитать количество дней, месяцев в заданном временном промежутке.

**Задание 11.** Создать объект класса Сутки, используя классы Час, Минута. Методы: вывести на консоль текущее время, рассчитать время суток (утро, день, вечер, ночь).

**Вариант 4.**

**Построить модель программной системы.**

**Задание 10.** Система Железнодорожная касса. Пассажир делает Заявку на станцию назначения, время и дату поездки. Система регистрирует Заявку и осуществляет поиск подходящего Поезда. Пассажир делает выбор Поезда и получает Счет на оплату. Администратор вводит номера Поездов, промежуточные и конечные станции, цены.

Задание 1. Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

1. ****Ход работы****

**Выполнение задания №10 первого варианта лабораторной работы:**

**import java.util.Arrays;**

**import java.util.Comparator;**

**import static java.lang.Math.min;**

**public class Main {**

**public static void main(String[] args) {**

**boolean [][] test = {{true, false, true, true}, {false, false, true, false}, {true, false, false, false}};**

**BoolMatrix bm1 = new BoolMatrix(test);**

**BoolMatrix bm2 = BoolMatrix.inverse(bm1);**

**System.out.println(Arrays.deepToString(BoolMatrix.conjunct(bm1, bm2).getMatrix()));**

**System.out.println(Arrays.deepToString(BoolMatrix.disjunct(bm1, bm2).getMatrix()));**

**System.out.println(bm1.trueCount());**

**System.out.println(bm2.trueCount());**

**System.out.println(Arrays.deepToString(bm1.sorted().getMatrix()));**

**System.out.println(Arrays.deepToString(bm2.sorted().getMatrix()));**

**}**

**}**

**class BoolMatrix {**

**private boolean [][] matrix;**

**private int width;**

**private int height;**

**public BoolMatrix(BoolMatrix other) {**

**this.matrix = other.getMatrix();**

**this.width = other.getWidth();**

**this.height = other.getHeight();**

**}**

**public BoolMatrix(boolean [][] other) {**

**this.matrix = other;**

**this.width = other[0].length;**

**this.height = other.length;**

**}**

**public boolean [][] getMatrix() {**

**return this.matrix;**

**}**

**public int getWidth() {**

**return this.width;**

**}**

**public int getHeight() {**

**return this.height;**

**}**

**public static BoolMatrix disjunct(BoolMatrix a, BoolMatrix b) {**

**boolean [][] res = new boolean [min(a.height,b.height)][min(a.width, b.width)];**

**for (int i = 0; i < min(a.height, b.height); i++) {**

**for (int j = 0; j < min(a.width, b.width); j++) {**

**res[i][j] = a.getMatrix()[i][j] && b.getMatrix()[i][j];**

**}**

**}**

**return new BoolMatrix(res);**

**}**

**public static BoolMatrix conjunct(BoolMatrix a, BoolMatrix b) {**

**boolean [][] res = new boolean [min(a.height,b.height)][min(a.width, b.width)];**

**for (int i = 0; i < min(a.height, b.height); i++) {**

**for (int j = 0; j < min(a.width, b.width); j++) {**

**res[i][j] = a.getMatrix()[i][j] || b.getMatrix()[i][j];**

**}**

**}**

**return new BoolMatrix(res);**

**}**

**public static BoolMatrix inverse(BoolMatrix a) {**

**boolean [][] res = new boolean [a.height][a.width];**

**for (int i = 0; i < a.height; i++) {**

**for (int j = 0; j < a.width; j++) {**

**res[i][j] = !a.getMatrix()[i][j];**

**}**

**}**

**return new BoolMatrix(res);**

**}**

**public int trueCount() {**

**int res = 0;**

**for (boolean[] row : this.getMatrix()) for (boolean col : row) if (col) res++;**

**return res;**

**}**

**public BoolMatrix sorted() {**

**boolean [][] res = Arrays.copyOf(this.getMatrix(), this.height);**

**Arrays.sort(res, Comparator.comparingInt(o -> {**

**int a = 0;**

**for (boolean d : o) {**

**a = (a << 1) | (d ? 1 : 0);**

**}**

**return a;**

**}));**

**return new BoolMatrix(res);**

**}**

**}**

**Выполнение задания №1 первого варианта лабораторной работы:**

**import static java.lang.Math.\*;**

**public class Main {**

**public static void main(String[] args) {**

**Vector [] array = {new Vector(1.2, 2.3, 3.4, 5.6),**

**new Vector(2.3, 4.5, 6.7, 8.9)};**

**System.out.println(Vector.dotProduct(array[0], array[1]));**

**System.out.println(array[0].magnitude());**

**System.out.println(array[1].magnitude());**

**}**

**}**

**class Vector {**

**private double [] coords;**

**public Vector(double... coords) {**

**this.coords = coords;**

**}**

**public double [] getCoords() {**

**return this.coords;**

**}**

**public static Vector add(Vector a, Vector b) {**

**double [] res = new double[min(a.getCoords().length, b.getCoords().length)];**

**for (int i = 0; i < res.length; i++) {**

**res[i] = ind(a, i) + ind(b, i);**

**}**

**return new Vector(res);**

**}**

**public static Vector substract(Vector a, Vector b) {**

**double [] res = new double[min(a.getCoords().length, b.getCoords().length)];**

**for (int i = 0; i < res.length; i++) {**

**res[i] = ind(a, i) - ind(b, i);**

**}**

**return new Vector(res);**

**}**

**public static Vector multiply(Vector a, int n) {**

**double [] res = new double[a.getCoords().length];**

**for (int i = 0; i < res.length; i++) {**

**res[i] = ind(a, i) \* n;**

**}**

**return new Vector(res);**

**}**

**public Vector incr() {**

**for (int i = 0; i < this.coords.length; i++) {**

**this.coords[i]++;**

**}**

**return this;**

**}**

**public Vector decr() {**

**for (int i = 0; i < this.coords.length; i++) {**

**this.coords[i]--;**

**}**

**return this;**

**}**

**public static double ind(Vector a, int index) {**

**return a.getCoords()[index];**

**}**

**public static double dotProduct(Vector a, Vector b) {**

**double res = 0;**

**for (int i = 0; i < min(a.getCoords().length, b.getCoords().length); i++) {**

**res += ind(a, i) \* ind(b, i);**

**}**

**return res;**

**}**

**public double magnitude() {**

**double res = 0;**

**for (double coord : this.coords) {**

**res += pow(coord, 2);**

**}**

**return sqrt(res);**

**}**

**public static double angle(Vector a, Vector b) {**

**return acos(dotProduct(a, b)/(a.magnitude()\*b.magnitude()));**

**}**

**}**

**Выполнение задания №10 второго варианта лабораторной работы:**

**import java.sql.Time;**

**import java.time.LocalDateTime;**

**import java.util.Calendar;**

**public class Main {**

**public static void main(String[] args) {**

**Train [] trains = {**

**new Train("Msk", 1337, LocalDateTime.of(2023, 02, 12, 22, 55),**

**4, 8, 15, 16),**

**new Train("Spb", 420, LocalDateTime.of(2023, 03, 01, 12, 30),**

**7, 3, 55, 6),**

**new Train("Msk", 69, LocalDateTime.of(2023, 4, 1, 7, 31),**

**0, 0, 0, 1)**

**};**

**System.out.println("Trains to Msk");**

**for (Train train : trains) {**

**if (train.getDestination() == "Msk") {**

**System.out.println(train.toString());**

**}**

**}**

**System.out.println("Trains to Msk after 2023-03-01 00:00");**

**for (Train train : trains) {**

**if (train.getDestination() == "Msk" && train.getDeparture().isAfter(LocalDateTime.of(2023, 3, 1, 0, 0))) {**

**System.out.println(train.toString());**

**}**

**}**

**System.out.println("Trains to Msk with general seats");**

**for (Train train : trains) {**

**if (train.getDestination() == "Msk" && train.getGeneralSeats() > 0) {**

**System.out.println(train.toString());**

**}**

**}**

**}**

**}**

**class Train {**

**private String destination;**

**private int trainNumber;**

**private LocalDateTime departure;**

**private int generalSeats;**

**private int coupeSeats;**

**private int plazkartSeats;**

**private int luxSeats;**

**public Train (String destination, int trainNumber, LocalDateTime departure,**

**int generalSeats, int coupeSeats, int plazkartSeats, int luxSeats) {**

**this.destination = destination;**

**this.trainNumber = trainNumber;**

**this.departure = departure;**

**this.generalSeats = generalSeats;**

**this.coupeSeats = coupeSeats;**

**this.plazkartSeats = plazkartSeats;**

**this.luxSeats = luxSeats;**

**}**

**public Train() {**

**}**

**public String getDestination() {**

**return this.destination;**

**}**

**public void setDestination(String destination) {**

**this.destination = destination;**

**}**

**public int getTrainNumber() {**

**return this.trainNumber;**

**}**

**public void setTrainNumber(int trainNumber) {**

**this.trainNumber = trainNumber;**

**}**

**public LocalDateTime getDeparture() {**

**return this.departure;**

**}**

**public void setDeparture(LocalDateTime departure) {**

**this.departure = departure;**

**}**

**public int getGeneralSeats() {**

**return this.generalSeats;**

**}**

**public void setGeneralSeats(int generalSeats) {**

**this.generalSeats = generalSeats;**

**}**

**public int getCoupeSeats() {**

**return this.coupeSeats;**

**}**

**public void setCoupeSeats(int coupeSeats) {**

**this.coupeSeats = coupeSeats;**

**}**

**public int getPlazkartSeats() {**

**return this.plazkartSeats;**

**}**

**public void setPlazkartSeats(int plazkartSeats) {**

**this.plazkartSeats = plazkartSeats;**

**}**

**public int getLuxSeats() {**

**return this.luxSeats;**

**}**

**public void setLuxSeats(int luxSeats) {**

**this.luxSeats = luxSeats;**

**}**

**public String toString() {**

**return "Train #" + this.trainNumber +**

**"\tdestinated to " + this.destination +**

**"\tdepartures " + this.departure +**

**"\tgeneral seats " + this.generalSeats +**

**"\tcoupe seats " + this.coupeSeats +**

**"\tplazkart seats " + this.plazkartSeats +**

**"\tlux seats " + this.luxSeats;**

**}**

**}**

**Выполнение задания №1 второго варианта лабораторной работы:**

**import java.time.LocalDate;**

**import java.util.\*;**

**public class Main {**

**public static void main(String[] args) {**

**Student [] students = {**

**new Student(1, "Kapkin", "Ivan", "Dmitrievich", LocalDate.of(2001, 02, 12),**

**"Moscow", "+79859300594", "IU", 1, "22M"),**

**new Student(1, "Awoind", "Oawnd", "Hawdja", LocalDate.of(2000, 02, 12),**

**"Spb", "+79823420594", "SM", 3, "63"),**

**new Student(1, "Aawdawi", "Kajwnda", "Nawjdna", LocalDate.of(2001, 02, 12),**

**"Nsk", "+79523400594", "IU", 2, "42M"),**

**new Student(1, "Iaowidjao", "Jaawid", "Najwdnawo", LocalDate.of(2002, 02, 12),**

**"York", "+79823420594", "IBM", 1, "12B"),**

**new Student(1, "Nawdajn", "Adwoanon", "Kawjnajod", LocalDate.of(2000, 02, 12),**

**"Moscow", "+7234200594", "IU", 1, "22M")**

**};**

**System.out.println("IU students");**

**for (Student student : students) {**

**if (student.getFaculty() == "IU") {**

**System.out.println(student.toString());**

**}**

**}**

**System.out.println("Students for faculties and courses");**

**Map<String, List<Student>> mapFaculty = new HashMap<>();**

**Map<Integer, List<Student>> mapCourse = new HashMap<>();**

**for (Student student : students) {**

**if (!mapFaculty.containsKey(student.getFaculty())) {**

**mapFaculty.put(student.getFaculty(), new ArrayList<>());**

**}**

**mapFaculty.get(student.getFaculty()).add(student);**

**if (!mapCourse.containsKey(student.getCourse())) {**

**mapCourse.put(student.getCourse(), new ArrayList<>());**

**}**

**mapCourse.get(student.getCourse()).add(student);**

**}**

**for (Map.Entry<String, List<Student>> entry : mapFaculty.entrySet()) {**

**System.out.println(entry.getKey());**

**for (Student student : entry.getValue()) {**

**System.out.println(student.toString());**

**}**

**}**

**for (Map.Entry<Integer, List<Student>> entry : mapCourse.entrySet()) {**

**System.out.println(entry.getKey());**

**for (Student student : entry.getValue()) {**

**System.out.println(student.toString());**

**}**

**}**

**System.out.println("Students after 2001");**

**for (Student student : students) {**

**if (student.getBirth().getYear() >= 2001) {**

**System.out.println(student.toString());**

**}**

**}**

**System.out.println("Students of IU 1 course Group 22M");**

**for (Student student : students) {**

**if (student.getFacultyCourseGroup().equals("IU 1 22M")) {**

**System.out.println(student.toString());**

**}**

**}**

**}**

**}**

**class Student {**

**private int id;**

**private String surname;**

**private String name;**

**private String lastname;**

**private LocalDate birth;**

**private String address;**

**private String phone;**

**private String faculty;**

**private int course;**

**private String group;**

**Student (int id, String surname, String name, String lastname, LocalDate birth,**

**String address, String phone, String faculty, int course, String group) {**

**this.id = id;**

**this.surname = surname;**

**this.name = name;**

**this.lastname = lastname;**

**this.birth = birth;**

**this.address = address;**

**this.phone = phone;**

**this.faculty = faculty;**

**this.course = course;**

**this.group = group;**

**}**

**Student() {}**

**public int getId() {**

**return this.id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getFullName() {**

**return this.surname + " " + this.name + " " + this.lastname;**

**}**

**public void setFullName(String surname, String name, String lastname) {**

**this.surname = surname;**

**this.name = name;**

**this.lastname = lastname;**

**}**

**public LocalDate getBirth() {**

**return this.birth;**

**}**

**public void setBirth(LocalDate birth){**

**this.birth = birth;**

**}**

**public String getAddress() {**

**return this.address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public String getPhone() {**

**return this.phone;**

**}**

**public void setPhone(String phone) {**

**this.phone = phone;**

**}**

**public String getFaculty() {**

**return this.faculty;**

**}**

**public int getCourse() {**

**return this.course;**

**}**

**public String getFacultyCourseGroup() {**

**return this.faculty + " " + this.course + " " + this.group;**

**}**

**public void setFacultyCourseGroup(String faculty, int course, String group) {**

**this.faculty = faculty;**

**this.course = course;**

**this.group = group;**

**}**

**public String toString() {**

**return "id " + this.id +**

**"\tFull name: " + this.getFullName() +**

**"\tbirth: " + this.birth +**

**"\taddress: " + this.address +**

**"\tphone: " + this.phone +**

**"\tgroup: " + this.getFacultyCourseGroup();**

**}**

**}**

**Выполнение задания №10 третьего варианта лабораторной работы:**

import java.time.DayOfWeek;

import java.time.LocalDate;

import java.time.Period;

import java.util.Arrays;

import java.util.Objects;

public class Main {

public static void main(String[] args) {

Year y1 = new Year(2023);

y1.setDate(03, 01);

y1.printDayOfWeek();

System.out.println(Year.getDaysBetween(2024, 02, 28, 2024, 03, 01));

System.out.println(Year.getMonthsBetween(2023, 02, 2024, 07));

Year y2 = new Year(2023);

y2.setDate(03, 01);

System.out.println(y1.equals(y2));

System.out.println(y1.toString());

System.out.println(y1.hashCode());

}

}

class Day {

private int day;

public Day(int day) {

this.day = day;

}

public int getDay() {

return day;

}

public void setDay(int day) {

this.day = day;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Day)) return false;

Day d = (Day) o;

return day == d.getDay();

}

}

class Month {

private int month;

public Month(int month) {

this.month = month;

}

public int getMonth() {

return month;

}

public void setMonth(int month) {

this.month = month;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Month)) return false;

Month m = (Month) o;

return month == m.getMonth();

}

}

class Year {

private int year;

private Month month;

private Day day;

public Year(int year) {

this.year = year;

}

public void setDate(int month, int day) {

this.month = new Month(month);

this.day = new Day(day);

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

public Month getMonth() {

return month;

}

public Day getDay() {

return day;

}

public void printDayOfWeek() {

LocalDate date = LocalDate.of(year, month.getMonth(), day.getDay());

DayOfWeek dayOfWeek = date.getDayOfWeek();

System.out.println(dayOfWeek);

}

public static int getDaysBetween(int startYear, int startMonth, int startDay, int endYear, int endMonth, int endDay) {

LocalDate startDate = LocalDate.of(startYear, startMonth, startDay);

LocalDate endDate = LocalDate.of(endYear, endMonth, endDay);

Period period = Period.between(startDate, endDate);

return period.getDays();

}

public static int getMonthsBetween(int startYear, int startMonth, int endYear, int endMonth) {

return endMonth - startMonth + 1 + (endYear - startYear) \* 12;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Year)) return false;

Year y = (Year) o;

return year == y.getYear() && month.equals(y.getMonth()) && day.equals(y.getDay());

}

@Override

public int hashCode() {

return Objects.hash(year, month, day);

}

@Override

public String toString() {

return "Year{" +

"year=" + year +

", month=" + month.getMonth() +

", day=" + day.getDay() +

'}';

}

}

**Выполнение задания №11 третьего варианта лабораторной работы:**

import java.util.Objects;

public class Main {

public static void main(String[] args) {

Minute m = new Minute(42);

Hour h = new Hour(13);

Day d = new Day(h, m);

d.printCurrentTime();

System.out.println(d.getTimeOfDay());

System.out.println(d.equals(new Day(new Hour(13), new Minute(42))));

System.out.println(d.hashCode());

System.out.println(d.toString());

}

}

class Minute {

private int minute;

public Minute (int minute) {

this.minute = minute%60;

}

public int getMinute() {

return minute;

}

public void setMinute(int minute) {

this.minute = minute%60;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Minute)) return false;

Minute m = (Minute) o;

return minute == m.minute;

}

}

class Hour {

private int hour;

public Hour (int hour) {

this.hour = hour%24;

}

public int getHour() {

return hour;

}

public void setHour(int hour) {

this.hour = hour%60;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Hour)) return false;

Hour h = (Hour) o;

return hour == h.hour;

}

}

class Day {

private Hour hour;

private Minute minute;

public Day(Hour hour, Minute minute) {

this.hour = hour;

this.minute = minute;

}

public void printCurrentTime() {

System.out.println("Current time is: " + hour.getHour() + ":" + minute.getMinute());

}

public String getTimeOfDay() {

int hourValue = hour.getHour();

if (hourValue >= 6 && hourValue < 12) {

return "morning";

} else if (hourValue >= 12 && hourValue < 18) {

return "afternoon";

} else if (hourValue >= 18 && hourValue < 22) {

return "evening";

} else {

return "night";

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Day)) return false;

Day day = (Day) o;

return hour.equals(day.hour) && minute.equals(day.minute);

}

@Override

public int hashCode() {

return Objects.hash(hour, minute);

}

@Override

public String toString() {

return "Day{" +

"hour=" + hour.getHour() +

", minute=" + minute.getMinute() +

'}';

}

}

**Выполнение задания №10 четвёртого варианта лабораторной работы:**

import java.time.LocalDateTime;

import java.util.\*;

public class Main {

public static void main(String[] args) {

List<Train> trains = new ArrayList<>();

trains.add(Administrator.makeTrain(1337, 215, "Msk", "Khi", "Skh", "Krk"));

trains.add(Administrator.makeTrain(420, 1050, "Msk", "Krk", "Pds", "Tvr"));

trains.add(Administrator.makeTrain(69, 10500, "Msk", "Tvr", "Blg", "Spb"));

Request request = Passenger.makeRequest("Krk", LocalDateTime.of(2023, 03, 17, 21, 45));

List<Train> filtered = trains.stream().filter(o -> o.hasStation(request.getDestination())).toList();

Train chosen = Passenger.pickOne(filtered);

Passenger passenger = new Passenger();

Cheque cheque = new Cheque(chosen);

passenger.takeCheque(cheque);

}

}

class Cheque {

private int cost;

public Cheque (Train train) {

this.cost = train.getPrice();

}

public int getCost() {

return cost;

}

}

class Request {

private String destination;

private LocalDateTime departure;

public Request (String destination, LocalDateTime departure) {

this.destination = destination;

this.departure = departure;

}

public String getDestination() {

return destination;

}

}

class Passenger {

private Cheque cheque;

public static Request makeRequest(String destination, LocalDateTime departure) {

return new Request(destination, departure);

}

public static Train pickOne(List<Train> trains) {

return trains.get(new Random(Calendar.getInstance().getTimeInMillis()).nextInt(trains.size()));

}

public void takeCheque(Cheque cheque) {

this.cheque = cheque;

}

}

class Administrator {

public static Train makeTrain(int trainNumber, int price, String... stations) {

return new Train(trainNumber, price, stations);

}

}

class Train {

private int trainNumber;

private String [] stations;

private int price;

public Train(int trainNumber, int price, String... stations) {

this.trainNumber = trainNumber;

this.price = price;

this.stations = stations;

}

public int getTrainNumber() {

return trainNumber;

}

public int getPrice() {

return price;

}

public boolean hasStation (String destination) {

return Arrays.stream(stations).anyMatch(str -> str.equals(destination));

}

}

**Выполнение задания №1 червёртого варианта лабораторной работы:**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Tutor [] tutors = {new Tutor("Stepanov P.V."),

new Tutor("Papulin S.Yu.")};

Course [] courses = {

tutors[0].announceCourse("Programming languages for work with big data"),

tutors[0].announceCourse("Languages of internet programming"),

tutors[1].announceCourse("Methods of machine learning")

};

Student [] students = {new Student("Kapkin I.D."),

new Student("Barishpol I.L.")};

for (Student student : students) for (Course course : courses) student.enlist(course);

for (Student student : students) for (Course course : courses) student.study(course);

for (Tutor tutor : tutors) for (Student student : students) for (Course course : courses) tutor.evaluate(student, course);

for (Student student : students) System.out.println(student.toString());

}

}

class Tutor {

private List<Course> courses;

public String name;

private Random evaluator;

public Tutor (String name) {

this.name = name;

this.courses = new ArrayList<>();

this.evaluator = new Random(Calendar.getInstance().getTimeInMillis());

}

public Course announceCourse (String name) {

Course course = new Course(name);

this.courses.add(course);

return course;

}

public boolean evaluate (Student student, Course course) {

return student.graduate(course, evaluator.nextInt(60, 100));

}

}

class Student {

private Map<Course, Integer> marks;

public String name;

public Student (String name) {

this.name = name;

this.marks = new HashMap<>();

}

public void enlist (Course course) {

this.marks.put(course, null);

}

public boolean study (Course course) {

return this.marks.containsKey(course);

}

public boolean graduate (Course course, int mark) {

if (this.marks.containsKey(course)) {

this.marks.put(course, mark);

return true;

}

return false;

}

@Override

public String toString() {

return this.name + marks.toString();

}

}

class Course {

public String name;

public Course (String name) {

this.name = name;

}

@Override

public String toString() {

return this.name;

}

}

1. ****Заключение****

**В процессе выполнения лабораторной работы были получены базовые навыки работы с языком программирования java, были изучены основные элементы ООП – классы, наследование и полиморфизм.**