МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5

Специальность ПО-5(о)

Выполнил

Д. С. Бриштен,

студент группы ПО-5

Проверил

А. А. Крощенко

ст. преп. каф. ИИТ

Брест 2021

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Вариант 4

Задание 1.Interface Учебное Заведение ← class Колледж ← class Университет.

Выполнение:

Код программы

public class Education {

    public static void main(String[] args) {

        College [] info = new College[2];

        info[0] = new University("Free", "State", 1050, 5);

        info[1] = new University("Charge", "Private", 0, 3);

        for (College temp : info)

            System.out.println(temp);

    }

}

interface EducationInstitution {

    String getOwner(); // получить собственника

    int getPlaces();

}

class College implements EducationInstitution {

    protected String education;

    protected String owner;

    protected int numStateFundedPlaces;

    public College(String education, String owner, int numStateFundedPlaces) {

        this.education = education;

        this.owner = owner;

        this.numStateFundedPlaces = numStateFundedPlaces;

    }

    public String getEducation() { return education; };

    @Override

    public String getOwner() { return owner; }

    @Override

    public int getPlaces() { return numStateFundedPlaces; }

}

class University extends College {

    private int numCorpus;

    public University(String education, String owner, int numStateFundedPlaces, int numCorpus) {

        super(education, owner, numStateFundedPlaces);

        this.numCorpus = numCorpus;

    }

    public int getNumCorpus() { return numCorpus; }

    @Override

    public String toString() {

        return  "\nCost of Education = '" + education + "\';\n" +

                "Number of corpuses = " + numCorpus + ";\n" +

                "Owner institutions = " + getOwner() + ";\n" +

                "Number state-funded places = " + getPlaces() + ".\n" +

                "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

    }

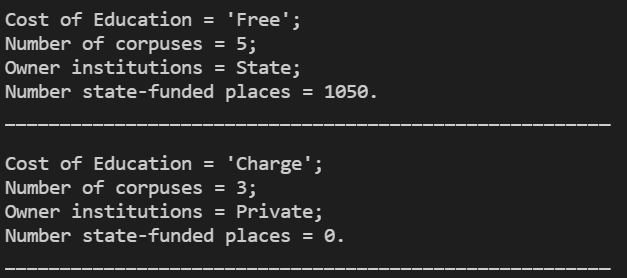
}

Спецификация вывода:

Используя переопределённый метод toString()

<education> + <numCorpus> + <owner> + < numStateFundedPlaces > and text.

Пример:



Задание 2. Создать суперкласс Грузоперевозчик и подклассы Самолет, Поезд, Автомобиль. Определить время и стоимость перевозки для указанных городов и расстояний.

Выполнение:

Код программы

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;

public class SuperClass {

    public static void main(String[] args) {

        CargoСarrier cargoCompany = new CargoСarrier();

        cargoCompany.Initialization();

        int num = cargoCompany.ChooseCity();

        cargoCompany.showPrice(num);

    }

}

class CargoСarrier

{

    private int numCity, num;

    private List<Integer> Distance = new ArrayList<Integer>();

    private List<String> City = new ArrayList<String>();

    Scanner in = new Scanner(System.in);

    public void Initialization()

    {

        char[] c = new char[10];

        Arrays.fill(c, '\*');

        String str = new String(c);

        str.replace('\n', '\0');

        System.out.println(str + " Cargo carrier parameters " + str);

        System.out.print("Enter the number of cities to which the cargo carrier delivers the cargo: ");

        this.numCity = in.nextInt();

        in.nextLine();

        for (int i = 0; i < this.numCity; i++)

        {

            System.out.print("Enter the name of the city at #" + i + ": ");

            this.City.add(in.nextLine());

            System.out.print("Specify the distance to the city " + this.City.get(i) + ": ");

            this.Distance.add(in.nextInt());

            in.nextLine();

        }

        System.out.println(str + " The end of initialization " + str);

    }

    public int ChooseCity()

    {

        do {

            System.out.print("Enter the number of the city where you want to deliver: ");

            num = in.nextInt();

            in.nextLine();

            if(num > this.numCity) {

                System.out.println("Error! There is no city with this number");

            }

            else { return num; }

        } while(true);

    }

    public void showPrice(int num)

    {

        String city = this.City.get(num);

        int s = this.Distance.get(num);

        System.out.println("\*\*\*\*\*\*\*\*" + " Pricelist for cargo transportation to the city " + city + "\*\*\*\*\*\*\*\*");

        System.out.println("\nDelivery by plane to the city " + city + ", distance to the city " + s);

        System.out.println("Price: " + this.AirplaneDeliveryCharges(s) +

        "; Duration of delivery: " + this.AirplaneDeliveryTime(s));

        System.out.println("\nDelivery by train to the city " + city + ", distance to the city " + s);

        System.out.println("Price: " + this.TrainDeliveryCharges(s) +

        "; Duration of delivery: " + this.TrainDeliveryTime(s));

        System.out.println("\nDelivery by car to the city " + city + ", distance to the city " + s);

        System.out.println("Price: " + this.CarDeliveryCharges(s) +

        "; Duration of delivery: " + this.CarDeliveryTime(s));

    }

    public int AirplaneDeliveryTime(int s)

    {

        return new Airplane().DeliveryTime(s);

    }

    public int TrainDeliveryTime(int s)

    {

        return new Train().DeliveryTime(s);

    }

    public int CarDeliveryTime(int s)

    {

        return new Car().DeliveryTime(s);

    }

    public int AirplaneDeliveryCharges(int s)

    {

        return new Airplane().DeliveryCharges(s);

    }

    public int TrainDeliveryCharges(int s)

    {

        return new Train().DeliveryCharges(s);

    }

    public int CarDeliveryCharges(int s)

    {

        return new Car().DeliveryCharges(s);

    }

}

class Airplane extends CargoСarrier

{

    private int price = 100000;

    private int speed = 500;

    public int DeliveryTime(int s)

    {

        return s / speed;

    }

    public int DeliveryCharges(int s)

    {

        int t = DeliveryTime(s);

        if (t >= 0 && t <= 10) return price;

        else if (t > 20) return price + 5000;

        else return price + 10000;

    }

}

class Train extends CargoСarrier

{

    private int price = 40000;

    private int speed = 120;

    public int DeliveryTime(int s)

    {

        return s / speed;

    }

    public int DeliveryCharges(int s)

    {

        int t = DeliveryTime(s);

        if (t >= 0 && t <= 10) return price;

        else if (t > 20) return price + 5000;

        else return price + 10000;

    }

}

class Car extends CargoСarrier

{

    private int price = 20000;

    private int speed = 80;

    public int DeliveryTime(int s)

    {

        return s / speed;

    }

    public int DeliveryCharges(int s)

    {

        int t = DeliveryTime(s);

        if (t >= 0 && t <= 10) { return price; }

        else if (t > 20) { return price + 5000; }

        else { return price + 10000; }

    }

}

Спецификация ввода:

numCity = <количество городов>

City = <названия городов>

Distance = <расстояния до городов>

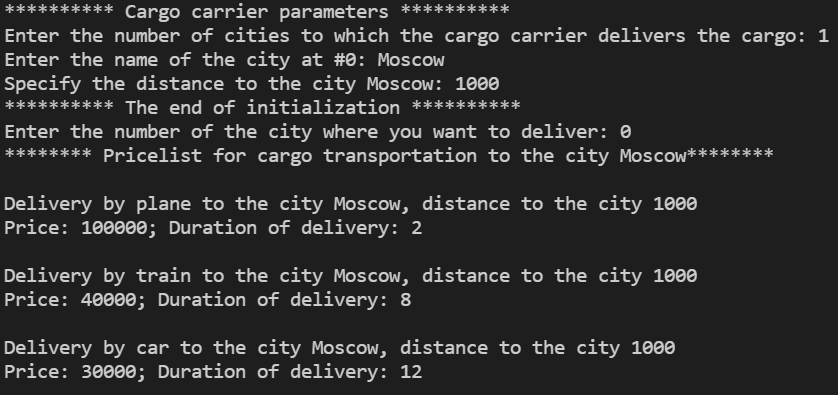
num = <номер города доставки>

Спецификация вывода:

Используя метод showPrice()

<city> + <s> + <price> + <t>, где s – расстояние, t – длительность доставки конкретным способом (самолёт, поезд, машина).

Пример:



Задание 3. В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных  
классов или интерфейсов.

Выполнение:

Код программы

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

abstract class Person {

    private String name, surname;

    public String getName() { return name; }

    public String getSurname() { return surname; }

    public Person(String name, String surname){

        this.name = name;

        this.surname = surname;

    }

    public abstract void show();

}

class Abiturient extends Person {

    public Integer assessment;

    private Faculty facultet; // Ассоциация

    private ArrayList<Exams> exams; // Ассоциация

    private Exams exam = new Exams("Show");;

    Abiturient(String name, String surname, String Facultet, ArrayList<Exams> exams) {

        super(name, surname);

        this.facultet = new Faculty(name, surname, Facultet); // Композиция

        this.exams = exams; // Агрегация

    }

    Abiturient(String name, String surname) {

        super(name, surname);

    }

    public void show() {

        System.out.println("\nAbiturient: " + super.getSurname() + " " + super.getName());

    }

    public void showResult() {

        System.out.println("Abiturient " + super.getSurname() + " " + super.getName() +

        " of faculty " + this.facultet.getFaculty() + " passed exams " + exam.show());

    }

}

class Faculty extends Abiturient {

    private String Namefacult, name, surname;

    Faculty(String name, String surname, String namefacult) {

        super(name, surname);

        this.name = name;

        this.surname = surname;

        this.Namefacult = namefacult;

    }

    String getFaculty() { return this.Namefacult; }

    void showFacult() {

        System.out.println("Abiturient " + this.name + " "

        + this.surname + " registered with the faculty: " + this.Namefacult);

    }

}

class Exams implements Printable {

    private String exam;

    static private ArrayList<Exams> exams = new ArrayList<Exams>();

    Exams(String Exam) {

        this.exam = Exam;

    }

    void setExams(Exams oun, Exams two) {

        exams.add(oun);

        exams.add(two);

    }

    ArrayList<Exams> getExams() { return exams; }

    public String show() {

        return exams.get(0).exam + " and " + exams.get(1).exam;

    }

}

class Teacher {

    public int Assessment;

    Teacher(Integer assessment) {

        this.Assessment = assessment;

    }

    public void show() {

        System.out.println("The teacher gave the grade " + this.Assessment);

    }

}

class Assessment {

    private int max = 10;

    private int min = 0;

    Integer getAssessment() { return (int)(Math.random()\*((max-min)+1))+min; }

}

interface Printable {

    String show();

}

public class Abstractclass {

    public static void main(String[] args) {

        HashMap<String, Integer> studentsResult = new HashMap<String, Integer>();

        Abiturient Putin = new Abiturient("Joseph", "Baiden");

        Putin.show();

        String Facultet = "FEIS";

        Exams math = new Exams("language");

        Exams history = new Exams("literature");

        Assessment assessment = new Assessment();

        math.setExams(math, history);

        ArrayList<Exams> exams = history.getExams();

        Putin = new Abiturient(Putin.getName(), Putin.getSurname(), Facultet, exams);

        Faculty facultet = new Faculty(Putin.getName(), Putin.getSurname(), Facultet);

        facultet.showFacult();

        Putin.showResult();

        Teacher Petya = new Teacher(assessment.getAssessment());

        Putin.assessment = Petya.Assessment;

        Petya.show();

        studentsResult.put(Putin.getSurname(), Putin.assessment);

        Abiturient Kim = new Abiturient("Bashar", "Asad");

        Kim.show();

        Kim = new Abiturient(Kim.getName(), Kim.getSurname(), Facultet, exams);

        Faculty facultet2 = new Faculty(Kim.getName(), Kim.getSurname(), Facultet);

        facultet2.showFacult();

        Kim.showResult();

        Teacher Petya2 = new Teacher(assessment.getAssessment());

        Kim.assessment = Petya2.Assessment;

        Petya2.show();

        studentsResult.put(Kim.getSurname(), Kim.assessment);

        System.out.println("\nStudents who passed the exams");

        for (Map.Entry<String, Integer> entry : studentsResult.entrySet())

            if (entry.getValue() >= 4)

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

    }

}

Спецификация вывода:

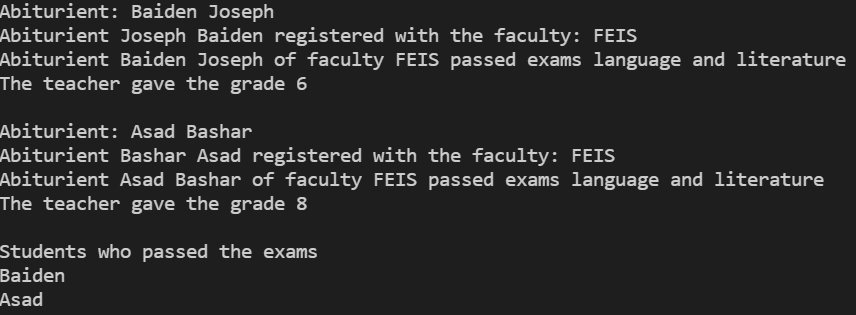
Метод Abiturient::show(): <фамилия> <имя>

Метод Abiturient::showResult(): <фамилия> <имя> <факультет> <экзамены>

Метод Faculty::showFacult(): <имя> <факультет>

Метод Teacher::show(): <оценка>

Пример:



Вывод: Получил начальные навыки в работе с классами и объектами в языке программирования Java.