МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №5

Выполнил:

студент 3 курса
группы ПО-9
Мельничук В.М.

Проверил:
Крощенко А.А.

Брест 2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования. **Вариант 1**

Задание 1

Абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство). Подклассы Справочник и Энциклопедия.

Выходные данные:

```
C:\Users\vladi\.jdks\openjdk-21.0.2\bin\java.exe "-ja
Справочник
Код: 22110
Автор: Б.А.Балтухин
Название: Биология в таблицах
Год: 2017
Издатель: Айрис-пресс
Предназначение: Школьная программа
Страницы: 113
Энциклопедия
Код: 33210
Автор: В.А.Кудрянова
Название: Все о динозаврах
Год: 2010
Издатель: Рамазан
Серия: Экнциклопедия для детского сада
Страницы: 32
```

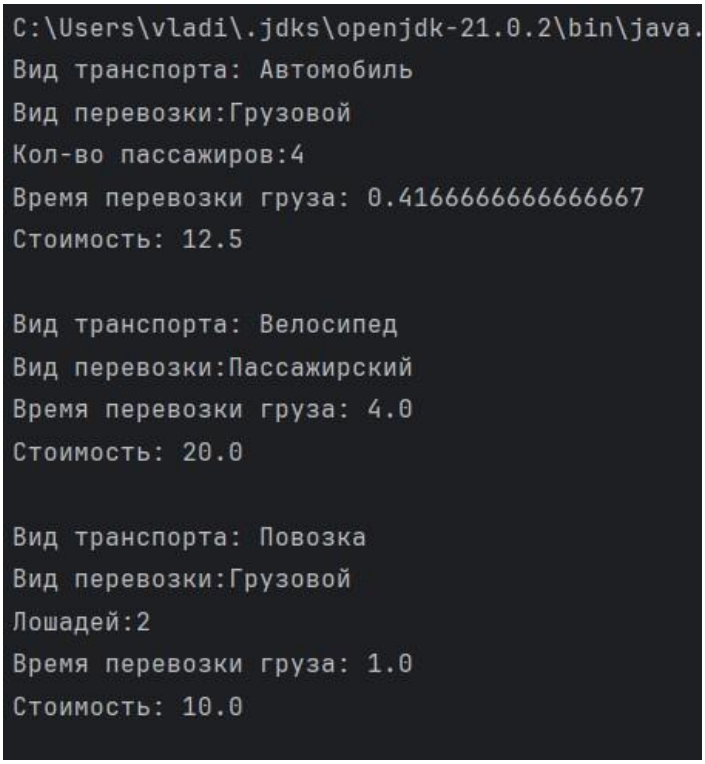
Код программы:

```
abstract class Book {
private int code;
private String author;
private String title;
private int year;
private String publisher;

 public Book(int code, String author, String title, int year, String  publisher) {
        this.code = code;
        this.author = author;
        this.title = title;
        this.year = year;
        this.publisher = publisher;      }
public abstract void display();
public String getPublisher() {
        return publisher;
}
public int getYear() {
        return year;
}
public String getTitle() {
        return title;
}
```

```
public String getAuthor() {
          return author;
}
public Integer getCode(){
          return code;
    }
}
class Guide extends Book{
private String destiny;
private int pages;

public Guide(int code, String author, String title, int year, String publisher, String destiny, int
pages){
         super(code, author, title, year, publisher);
         this.destiny = destiny;
         this.pages = pages;
}

@Override
    public void display(){
        System.out.println("Справочник\nКод: " + super.getCode() + "\nАвтор: " + super.getAuthor() +
"\nНазвание: " + super.getTitle() + "\nГод: " + super.getYear() + "\nИздатель: " + super.getPublisher() +
"\nПредназначение: "  + destiny + "\nСтраницы: " + pages);
}
}

class Encyclopedia extends Book{
private String series;
private int pages;

public Encyclopedia(int code, String author, String title, int year, String publisher, String series,
int pages){
         super(code, author, title, year, publisher);
         this.series = series;
         this.pages = pages;
    }
    public void display(){
        System.out.println("Энциклопедия\nКод: " + getCode() + "\nАвтор: " + getAuthor() + "\nНазвание: "
+ getTitle() + "\nГод: " + getYear() + "\nИздатель: " + getPublisher() + "\nСерия: " + series +
"\nСтраницы: " + pages);
} }
public class task1 {
    public static void main(String[] args) {
        Guide book1 = new Guide(22110, "Б.А.Балтухин", "Биология в таблицах",  2017, "Айрис-пресс",
"Школьная программа", 113);
        book1.display();
        Encyclopedia book2 = new Encyclopedia(33210, "В.А.Кудрянова", "Все о динозаврах", 2010,
"Рамазан", "Экнциклопедия для детского сада", 32);
        book2.display();
    }
}
```

Задание 2

Создать суперкласс Транспортное средство и подклассы Автомобиль, Велосипед, Повозка. Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.
Выходные данные:



Код программы

```
abstract class Transport{
     private String name;
     private String type;
     private double speed;
     private double costPerHour;

     public Transport(String name, String type, double speed, double costPerHour){
        this.name = name;
        this.type = type;
        this.speed = speed;
        this.costPerHour = costPerHour;
    }

    public String getType(){
         return type;
    }

    public double calculatePassengerTime(double distance){
         return distance / speed;
    }

    public double calculateCargoTime(double distance){
         return distance / speed;
    }

    public double calculatePassengerCost(double distance){
         double time = calculatePassengerTime(distance);
         return time * costPerHour;
    }

    public double calculateCargoCost(double distance){
         double time = calculateCargoTime(distance);
         return time * costPerHour;
    }

    public void information(){
        System.out.println("Вид транспорта: " + name + "\nВид перевозки: " + type);
    }
}
```

```
    class Car extends Transport{
     private int passengerCapacity;
     public Car(String name, String type, int speed, int costPerHour, int passengerCapacity){
          super(name, type,speed, costPerHour);
          this.passengerCapacity = passengerCapacity;
    }
     @Override
    public void information(){
        super.information();
        System.out.println("Кол-во пассажиров: " + passengerCapacity);
    }
}

    class Bicycle extends Transport{
    public Bicycle(String name, String type, int speed, int costPerHour){
         super(name, type, speed, costPerHour);
    }
}
    class Cart extends Transport{
     private int horsePower;

     public Cart(String name, String type, int speed, int costPerHour, int horsePower){
        super(name, type,speed, costPerHour);
        this.horsePower = horsePower;
    }

    @Override
    public void information(){
        super.information();
        System.out.println("Лошадей: " + horsePower);
    }

}

public class task2 {
    public static void main(String[] args){
         Transport[] transports = new Transport[3];
         transports[0] = new Car("Автомобиль", "Грузовой", 120, 30, 4);
         transports[1] = new Bicycle("Велосипед", "Пассажирский", 25, 5);
         transports[2] = new Cart("Повозка", "Грузовой", 50, 10, 2);

        for(Transport transport : transports){
             transport.information();
            if(transport.getType().equals("Пассажирский")){
                double       passengerTime  = transport.calculatePassengerTime(100);
                double       passengerCost  = transport.calculatePassengerCost(100);
                System.out.println("Время перевозки груза: " + passengerTime + "\nСтоимость: " +
passengerCost + "\n");
            }else {
                double cargoTime = transport.calculateCargoTime(50);
                doublecargoCost = transport.calculateCargoCost(50);
                System.out.println("Время перевозки груза: " + cargoTime + "\nСтоимость: " + cargoCost
+ "\n");
           }
        }
}
```

Задание 3

Код программы:

```
import java.util.ArrayList;
import java.util.List;

interface Registrable{
    void advertReg(Elective elective);
}
interface Markable{
```

```
    void setMark(Student student, int mark, Elective elective);
}
class Elective{
    private String name;
    private Teacher teacher;
    private List<Student> students;

    public Elective(String name, Teacher teacher){
        this.name = name;
        this.teacher = teacher;
        students = new ArrayList<>();
    }

    public void registration(Student student){
        students.add(student);
    }
    public String getName(){
        return name;
    }

    public void learning(){
        System.out.println("Обучение на факультативе " + name + " началось.");
    }

    public void finish(){
        System.out.println("Обучение на факультативе " + name + " завершено.");
    }

}

class Teacher implements Registrable, Markable{
    private String name;

    public Teacher(String name){
        this.name = name;
    }
    public void advertReg(Elective elective){
        System.out.println("Запись на курс \"" + elective.getName() + "\" объявлена преподавателем " +
name + ".");
    }

    public void setMark(Student student, int mark, Elective elective){
        Archive.saveMark(student, mark, elective);
        System.out.println("Оценка " + mark + " выставлена " + student.getName());
    }
}

class Student implements Registrable{
    private int id;
    private String name;

    public Student(Integer id, String name){
        this.id = id;
        this.name = name;
    }

    public void advertReg(Elective elective){
        elective.registration(this);
        System.out.println("Студент " + name + " записался на курс " + elective.getName());
    }

    public String getName(){
        return name;
    }
}

class Archive{
    private static List<String> marks;

    public Archive() {
```

```
        marks = new ArrayList<>();
    }
    public static void saveMark(Student student, int mark, Elective elective){
        marks.add("Студент: " + student.getName() + ", Курс: " + elective.getName() +  ", Оценка: " +
mark);
    }

    public List<String> getMarks(){
        return marks;
    }
}
public class task3{
    public static void main(String[] args){
        Teacher teacher = new Teacher("Иванченко А.В.");
        Teacher teacher1 = new Teacher("Василенко В.Д.");

        Elective elective = new Elective("Математика", teacher);
        Elective elective1 = new Elective("Физика", teacher1);
        Archive archive = new Archive();

        teacher.advertReg(elective);
        teacher1.advertReg(elective1);

        Student student1 = new Student(210663, "Melenkov");
        Student student2 = new Student(210664, "Molankova");

        student1.advertReg(elective);
        student2.advertReg(elective);

        student1.advertReg(elective1);
        student2.advertReg(elective1);

        elective.learning();
        elective1.learning();
        elective.finish();
        elective1.finish();

        teacher.setMark(student1, 4, elective);
        teacher1.setMark(student1, 2, elective1);

        teacher.setMark(student2, 5, elective);
        teacher1.setMark(student2, 3, elective1);

        List<String> marks = archive.getMarks();
        for(String mark : marks){
            System.out.println(mark);
        }
    }
}
```