МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе №2.

Выполнил: студент группы БВТ2402

Безматерных Иван Алексеевич

Москва, 2025

Цель работы: ознакомиться с основами объектно-ориентрованного программирования на языке Java, создать программу по заданным условиям.

Задание.

Создайте иерархию классов в соответствии вариантом. Ваша иерархия должна содержать:

• абстрактный класс;

• два уровня наследуемых классов (классы должны содержат в себе минимум 3 поля и 2 метода, описывающих поведение

объекта);

• демонстрацию реализации всех принципов ООП;

• наличие конструкторов (в том числе по умолчанию);

• наличие геттеров и сеттеров;

• ввод/вывод информации о создаваемых объектах;

• предусмотрите в одном из классов создание счетчика созданных объектов с использованием статической переменной,

продемонстрируйте работу.

4. Базовый класс: Транспортное средство. Дочерние классы: Легковой автомобиль, Грузовой автомобиль, Мотоцикл.

Ход работы:

1) Создал абстрактный класс Transport, имеющий базовые характеристики всех представителей. Далее создал конструктор по умолчанию, который задает базовые значения, если они не были указаны. Также создал параметризованный конструктор, принимающий значения для всех полей. Оба конструкторы имеют счетчик количества созданных представителей класса.

```
abstract class Transport{
    protected String marka;
    protected int release_year;
    protected double engine_capacity;
    private static int objectCount = 0;

    public Transport(){
        this.marka = "Unknown";
        this.release_year = 0;
        this.engine_capacity = 5;
        objectCount++;
    }

    public Transport(String marka, int release_year, double engine_capacity){
        this.marka = marka;
        this.release_year = release_year;
        this.engine_capacity = engine_capacity;
        objectCount++;
    }
```

2) В абстрактном класс создал методы get и set, позволяющие посмотреть заданные условия или поменять их. Также создал методы запуска и остановки двигателя и метод Out, определенный для каждого класса по-своему.

```java
public String getMarka(){
    return marka;
}

public void setMarka(String newMarka){
    this.marka = newMarka;
}

public int getRelease_year(){
    return release_year;
}

public void setRelease_year(int newYear){
    this.release_year = newYear;
}

public double getEngine_capacity(){
    return engine_capacity;
}

public void setEngine_capacity(double  NewCapacity){
    this.engine_capacity = NewCapacity;
}
```

```java
public void StartEngine(){
    System.out.println(x:"Двигатель запущен.");
}

public void StopEngine(){
    System.out.println(x:"Двигатель заглушен.");
}

public void Out(){
}
```

3) Создал метод Info для вывода информации об объекте. Метод getObject показывает количество созданных экземпляров.

```
public void Info(){
    System.out.println("Марка авто: " + marka);
    System.out.println("Год выпуска авто: " + release_year);
    System.out.println("Объем двигателя авто: " + engine_capacity);
}

public int getObject(){
    return objectCount;
}
```

4) Создал класс Car, наследующий абстрактный класс Transport. Ввел новые переменные. Реализовал конструктор по умолчанию и конструктор с параметрами, которые наследуются от абстрактного класса для совпадающих полей, для остальных дописал значения.

```
class Car extends Transport{
    protected int number_of_seats;
    protected String body_type;

    public Car(){
        super();
        this.number_of_seats = 5;
        this.body_type = "Unknown";
    }

    public Car(String marka, int release_year, double engine_capacity, int number_of_seats, String body_type){
        super(marka, release_year, engine_capacity);
        this.number_of_seats = number_of_seats;
        this.body_type = body_type;
    }
```

5) Переписал метод Out и Info под необходимые параметры, создал методы get и set для новых полей, реализовал новые методы для этого класса.

```java
@Override
public void Out(){
    System.out.println(x:"Это легковой автомобиль.");
}

@Override
public void Info(){
    super.Info();
    System.out.println("Количество посадочных мест: " + number_of_seats);
    System.out.println("Тип кузова: " + body_type);
}

public void setNumber_of_seats(int number){
    this.number_of_seats = number;
}

public int getNumber_of_seats(){
    return number_of_seats;
}

public void setBody_type(String type){
    this.body_type = type;
}
```

```java
public String getBody_type(){
    return body_type;
}

public void Open_trunk(){
    System.out.println(x:"Багажник открыт.");
}

public void Close_trunk(){
    System.out.println(x:"Багажник закрыт.");
}
```

6) Создал класс Tank, наследуемый от Car. В нем по умолчанию ввел значение марки, остальные поля совпадают с родительским классом.

```java
class Tank extends Car{
    public Tank(){
        super();
        this.marka = "TANK";
    }
    public Tank(String marka, int release_year, double engine_capacity, int number_of_seats, String body_type){
        super(marka, release_year, engine_capacity, number_of_seats, body_type);
    }

    public void Out(){
        super.Out();
    }

    public void Info(){
        super.Info();;
    }

    public void setNumber_of_seats(int number){
        this.number_of_seats = number;
    }

    public int getNumber_of_seats(){
        return number_of_seats;
    }
```

```java
public void setBody_type(String type){
    this.body_type = type;
}

public String getBody_type(){
    return body_type;
}

public void Open_trunk(){
    System.out.println(x:"Багажник открыт.");
}

public void Close_trunk(){
    System.out.println(x:"Багажник закрыт.");
}
```

7) Создал класс Truck, наследуемый от класса Transport, реализовал новые поля, базовый конструктор и конструктор с параметрами, методы get и set, переписал методы Info, Out, создал новый метод Signal.

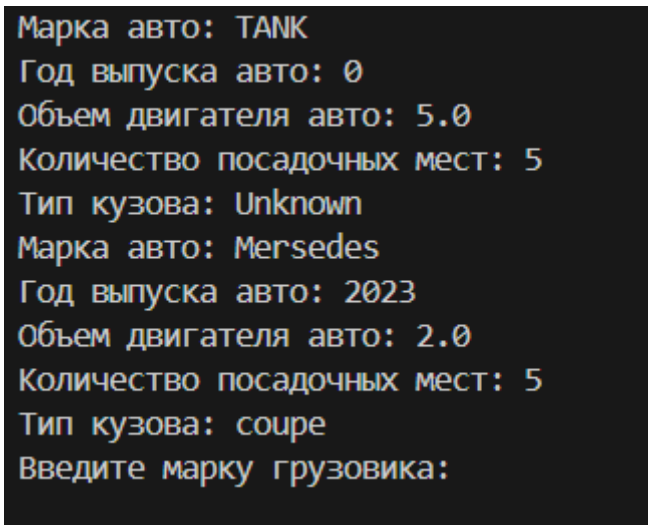
```java
class Truck extends Transport{
    protected int wheels_count;

    public Truck(){
        super();
        this.wheels_count = 4;
    }

    public Truck(String marka, int release_year, double engine_capacity, int wheelCount){
        super(marka, release_year, engine_capacity);
        this.wheels_count = wheelCount;
    }

    @Override
    public void Out(){
        System.out.println(x:"Это грузовой автомобиль.");
    }

    @Override
    public void Info(){
        super.Info();
        System.out.println("Количество колес: " + wheels_count);
    }

    public int getWheels_count(){
        return wheels_count;
    }

    public void setEhwwls_count(int count){
        this.wheels_count = count;
    }

    public void Signal(){
        System.out.println(x:"Tyyyyy Tyyyyy");
    }
}
```

8) Создал класс Bicy, наследуемый от класса Transport, реализовал новые поля, базовый конструктор и конструктор с параметрами, методы get и set, переписал методы Info, Out, создал новый метод Helmet.

```
class Bicy extends Transport{
    protected int max_speed;

    public Bicy(){
        super();
        this.max_speed = 100;
    }

    public Bicy(String marka, int release_year, double engine_capacity, int max_speed){
        super(marka, release_year, engine_capacity);
        this.max_speed = max_speed;
    }

    @Override
    public void Info(){
        super.Info();
        System.out.println("Максимальная скорость: " + max_speed);
    }

    @Override
    public void Out(){
        System.out.println(x:"Это мотоцикл.");
    }

    public int getMax_speed(){
        return max_speed;
    }

    public void setMax_speed(int speed){
        this.max_speed = speed;
    }

    public void Helmet(){
        System.err.println(x:"Шлем надет.");
    }
}
```

9) Создал тестовый экземпляр test1 для проверки создания класса Lada. Далее создал экземпляр BMW с заданными параметрами, проверил метод Info. Далее создал экземпляр, используя введенные с консоли параметры. Продемонстрировал методы Info, Out, get, set, Signal. Продемонстрировал работу счетчик.

```java
public class Indicator{
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Tank test1 = new Tank();
        test1.Info();

        Scanner sc = new Scanner(System.in);
        Car MERS = new Car(marka:"Mersedes",release_year:2023, engine_capacity:2.0, number_o…5, "coupe");
        MERS.Info();
        System.out.print(s:"Введите марку грузовика: ");
        String mr = sc.nextLine();

        System.out.print(s:"Введите год выпуска грузовика: ");
        int ye = sc.nextInt();

        System.out.print(s:"Введите объем двигателя грузовика через ,: ");
        double en = sc.nextDouble();

        System.out.print(s:"Введите количество колес: ");
        int nu = sc.nextInt();

        Truck user = new Truck(mr, ye, en, nu);
        user.Info();
        user.Out();
        System.out.print(s:"введите новое количество колес: ");
        int nenu = sc.nextInt();
        user.setEhwwls_count(nenu);
        System.out.println("Количество колес: " + user.getWheels_count());
        user.Signal();
        System.out.println("Количество транспортных средств: " + user.getObject());
        Bicy pokaSz = new Bicy();
        System.out.println("Количество транспортных средств: " + user.getObject());
    }
```



Вывод: ознакомился с основѓми ООП на языке Java, создал программу по заданным условиям.