МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2

по дисциплине «Программирование на языке Джава»

Выполнил студент группы ИКБО-68-24 Данильченко Д.А

Принял Ермаков С.Р.
Старший преподаватель

Москва 2025

Задача 2.1

Описание задачи, цель

Создайте пакет vehicles, который будет содержать классы Car и ElectricCar и пакет app, в котором будет находиться основной класс с методом main. Добавьте в класс Car приватные поля (private) ownerName и insuranceNumber. Создайте методы доступа (геттеры и сеттеры) для полей ownerName и insuranceNumber. Добавьте поле engineType с модификатором доступа protected и создайте методы доступа к этому полю.

Ход работы

Создадим систему папок:

```
src
      app
            Main.java
      vehicles
            Car.java
            ElectricCar.java
```

Для файлов пакета vehicles добавим в начало каждого:

```
package vehicles;
```

В ту же очередь для Main.java

```
package app;
```

Добавим для класса Car поля ownerName и insuranceNumber с модификатором доступа private:

```
public class Car {
    private String ownerName;
    private int insuranceNumber;
```

Реализуем геттеры и сеттеры для заданных полей

```
public String getOwnerName(){
    return this.ownerName;
}
public int getInsuranceNumber(){
    return this.insuranceNumber;
}
```

```
public void setOwnerName(String OwnerName){
    this.ownerName = OwnerName;
}
```

```
public void setInsuranceNumber(int insuranceNumber){
    this.insuranceNumber = insuranceNumber;
}
```

Добавим поле engineType с модификатором доступа protected, а также методы доступа к нему:

```
protected String engineType;
```

```
public String getEngineType(){
    return this.engineType;
}
```

```
public void setEngineType(String engineType){
    this.engineType = engineType;
}
```

Итоговый код программы:

app/Main.java

```
package app;

public class Main {
    public static void main(String[] args) {
    }
}
```

vehicles/Car.java

```
package vehicles;

public class Car {
    private String ownerName;
    private int insuranceNumber;
    protected String engineType;

    public String getOwnerName(){
        return this.ownerName;
    }
    public int getInsuranceNumber(){
        return this.insuranceNumber;
    }
    public String getEngineType(){
        return this.engineType;
    }
```

```
    public void setOwnerName(String OwnerName){
        this.ownerName = OwnerName;
    }
    public void setInsuranceNumber(int insuranceNumber){
        this.insuranceNumber = insuranceNumber;
    }
    public void setEngineType(String engineType){
        this.engineType = engineType;
    }
}
```

vehicles/ElectricCar.java

```
package vehicles;

public class ElectricCar {
}
```

Результаты и выводы

Программа демонстрирует организацию кода с использованием пакетов и реализацию различных модификаторов доступа (private, protected). Освоены навыки правильного построения архитектуры проектов Java.

Задача 2.2

Описание задачи, цель

Создайте новый класс ElectricCar, который наследует класс Car, и добавьте в него поле batteryCapacity. В классе ElectricCar используйте поле engineType, чтобы задать тип двигателя как "Electric". Проверьте работу инкапсуляции и наследования, создав объекты классов Car и ElectricCar и продемонстрируйте доступ к полям с разными модификаторами.

Ход работы

В класс ElectricCar добавляем extends Car для наследования полей и методов класса

```
public class ElectricCar extends Car {}
```

Добавим публичное поле batteryCapacity

```
public int batteryCapacity;
```

Добавим в класс блок инициализации с полем engineType и значением "Electric"

```
{
    engineType = "Electric";
}
```

Создадим объекты классов и реализуем демонстрацию доступа к полям разного доступа в Main.java (Также добавим “import vehicles.*;” в начало файла для доступа к пакету с классами)

```
    //Создание объектов
    Car defaultCar = new Car();
    ElectricCar electrocar = new ElectricCar();

    //Демонстрация обращения к приватным полям (только через сеттеры и
геттеры)
    defaultCar.setOwnerName("Dio");
    electrocar.setOwnerName("Jotaro");

    System.out.println("Default car owner: "+defaultCar.getOwnerName()+";
Electrocar owner: "+electrocar.getOwnerName());

    //Демонстрация обращения к публичным полям
    electrocar.batteryCapacity = 6;
    System.out.println("Battery capacity = "+electrocar.batteryCapacity);
```

Итоговый код программы:

app/Main.java

```
package app;

import vehicles.*;

public class Main {
    public static void main(String[] args) {
        //Создание объектов
        Car defaultCar = new Car();
        ElectricCar electrocar = new ElectricCar();

        //Демонстрация обращения к приватным полям (только через сеттеры и
геттеры)
        defaultCar.setOwnerName("Dio");
        electrocar.setOwnerName("Jotaro");

        System.out.println("Default car owner:
"+defaultCar.getOwnerName()+"; Electrocar owner:
"+electrocar.getOwnerName());

        //Демонстрация обращения к публичным полям
        electrocar.batteryCapacity = 6;
        System.out.println("Battery capacity =
"+electrocar.batteryCapacity);
    }
}
```

vehicles/Car.java

```
package vehicles;

public class Car {
    private String ownerName;
    private int insuranceNumber;
    protected String engineType;

    //геттеры
    public String getOwnerName(){
        return this.ownerName;
    }
    public int getInsuranceNumber(){
        return this.insuranceNumber;
    }
    public String getEngineType(){
        return this.engineType;
    }

    //сеттеры
    public void setOwnerName(String OwnerName){
        this.ownerName = OwnerName;
```

```
    }
    public void setInsuranceNumber(int insuranceNumber){
        this.insuranceNumber = insuranceNumber;
    }
    public void setEngineType(String engineType){
        this.engineType = engineType;
    }
}


    public String getEngineType(){
        return this.engineType;
    }

    public void setOwnerName(String OwnerName){
        this.ownerName = OwnerName;
    }
    public void setInsuranceNumber(int insuranceNumber){
        this.insuranceNumber = insuranceNumber;
    }
    public void setEngineType(String engineType){
        this.engineType = engineType;
    }
}
```

vehicles/ElectricCar.java

```
package vehicles;

public class ElectricCar extends Car {
    public int batteryCapacity;
    public ElectricCar(){
        super();
    }
    { engineType = "Electric"; }
}
```

Результаты и выводы

Вывод программы:

```
Default car owner: Dio; Electrocar owner: Jotaro
Battery capacity = 6

Process finished with exit code 0
```

Программа наглядно демонстрирует ключевые принципы объектно-ориентированного программирования: наследование, доступ к данным через методы и использование общего интерфейса для разных типов автомобилей. Освоены практические навыки работы с модификаторами доступа и создания иерархии классов в Java.

Задача 2.3

Описание задачи, цель

Ваша программа должна быть организована по пакетам:

● Пакет vehicles для классов Vehicle, Car, ElectricCar

● Пакет app для тестового класса TestCar.

Используя программу, выполненную ранее, внести следующие изменения:

1. Добавить абстрактный класс Vehicle, который будет представлять общие свойства всех транспортных средств. В этот класс включите следующие общие поля для транспортных средств: model (модель); license (номерной знак); color (цвет); year (год выпуска); ownerName (имя владельца); insuranceNumber (страховой номер); engineType (тип двигателя, поле должно быть защищённым для наследования). Определите абстрактный метод vehicleType(), который будет возвращать тип транспортного средства. Добавьте методы для получения и изменения значений полей (геттеры и сеттеры).
2. Изменить класс Car, чтобы он наследовал Vehicle. Реализуйте абстрактный метод vehicleType(), чтобы он возвращал "Car". В конструкторе класса Car используйте поля и методы родительского класса.
3. Изменить класс ElectricCar, чтобы он наследовал Car. Добавьте в класс поле batteryCapacity (емкость аккумулятора) и методы для работы с ним. Реализуйте метод vehicleType(), который будет возвращать "Electric Car". Используйте protected-поле engineType для установки значения "Electric" в классе ElectricCar.
4. Использовать полиморфизм в тестовом классе для работы с объектами Car и ElectricCar через ссылки на родительские классы. Создайте объекты Car и ElectricCar, измените их свойства с помощью сеттеров, и выведите информацию на экран с помощью метода toString().
5. Включить инкапсуляцию: убедитесь, что поля каждого класса имеют доступ через методы (геттеры и сеттеры), а не напрямую.

Ход работы

1. Реализуем новый класс Vehicle, с помощью уже приобретенных навыков

Vehicle.java

```
package vehicles;

public abstract class Vehicle {
    protected String model; //(модель)
    protected String license; //(номерной знак)
    protected String color; //(цвет)
    protected int year; //(год выпуска)
    protected String ownerName; //(имя владельца)
    protected int insuranceNumber; //(страховой номер)
    protected String engineType; //(тип двигателя)

    //Конструктор Vehicle
    public Vehicle(String model, String license, String color, int year,
```

```
                    String ownerName, int insuranceNumber, String engineType)
{
        this.model = model; this.license = license; this.color = color;
this.year = year;
        this.ownerName = ownerName; this.insuranceNumber = insuranceNumber;
this.engineType = engineType;
    }
    public Vehicle() {}

    //Геттеры
    public String getModel() {
        return this.model;}
    public String getLicense() {
        return this.license;}
    public String getColor() {
        return this.color;}
    public int getYear() {
        return this.year;}
    public String getOwnerName() {
        return this.ownerName;}
    public int getInsuranceNumber() {
        return this.insuranceNumber;}
    public String getEngineType() {
        return this.engineType;}

    // Сеттеры
    public void setModel(String model) {
        this.model = model;}
    public void setLicense(String license) {
        this.license = license;}
    public void setColor(String color) {
        this.color = color;}
    public void setYear(int year) {
        this.year = year;}
    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;}
    public void setInsuranceNumber(int insuranceNumber) {
        this.insuranceNumber = insuranceNumber;}
    public void setEngineType(String engineType) {
        this.engineType = engineType;}

    //Абстрактный метод vehicleType
    public abstract String vehicleType();
}
```

2. Применим наследование класса Vehicle для класса Car

Car.java

```
public class Car extends Vehicle {...
```

Для класса Car реализуем метод `vehicleType()`

```
    public String vehicleType() {
        return "Car";
    }
```

Создадим конструкторы для Car используя super для вызова конструктора родительского класса

```
    public Car(String model, String license, String color, int year,
               String ownerName, int insuranceNumber, String engineType) {
        super(model, license, color, year, ownerName, insuranceNumber,
engineType);
    }
    public Car() {
    }
```

3. Наследование класса Car для ElectricCar

ElectricCar.java

```
public class ElectricCar extends Car {...
```

Добавим поле batteryCapacity и методы работы с ним

```
    public int batteryCapacity;

    public void setBatteryCapacity(int batteryCapacity) {
        this.batteryCapacity = batteryCapacity;
    }
    public int getBatteryCapacity() {
        return batteryCapacity;
    }
```

Добавим метод vehicleType()

```
    public String vehicleType() {
        return ("Electric Car");
    }
```

Для поля engineType установим значение “Electric”

```
    { engineType = "Electric"; }
```

4. На этом моменте удалим лишний код из Car.java

```
package vehicles;
```

```
public class Car extends Vehicle {
    private String ownerName;
    private int insuranceNumber;
    protected String engineType;

    //геттеры
    public String getOwnerName(){
        return this.ownerName;
    }
    public int getInsuranceNumber(){
        return this.insuranceNumber;
    }
    public String getEngineType(){
        return this.engineType;
    }

    //сеттеры
    public void setOwnerName(String OwnerName){
        this.ownerName = OwnerName;
    }
    public void setInsuranceNumber(int insuranceNumber){
        this.insuranceNumber = insuranceNumber;
    }
    public void setEngineType(String engineType){
        this.engineType = engineType;
    }
}
    public String getEngineType(){
        return this.engineType;
    }

    public void setOwnerName(String OwnerName){
        this.ownerName = OwnerName;
    }
    public void setInsuranceNumber(int insuranceNumber){
        this.insuranceNumber = insuranceNumber;
    }
    public void setEngineType(String engineType){
        this.engineType = engineType;
    }
    public String vehicleType() {
        return "Car";
    }
    public Car(String model, String license, String color, int year,
               String ownerName, int insuranceNumber, String engineType) {
        super(model, license, color, year, ownerName, insuranceNumber,
engineType);
    }
    public Car() {
    }
}
```

Добавим в класс Vehicle.java метод toString()

```
    @Override
    public String toString(){
        return (getClass().getSimpleName()+":"+
                "\n  model: "+this.model+
                "\n  license: "+this.license+
                "\n  color: "+this.color+
                "\n  year: "+this.year+
                "\n  ownerName: "+this.ownerName+
                "\n  insuranceNumber: "+this.insuranceNumber+
                "\n  engineType: "+this.engineType
        );
```

Добавим в Main.java реализацию 4 пункта задания

```
        //Создание объектов через родительский класс
        Vehicle car = new Car();
        Vehicle electroCar = new ElectricCar();

        //Изменение параметров через сеттеры и вывод
        car.setColor("white");
        car.setOwnerName("Mr.White");
        car.setInsuranceNumber(322);
        System.out.println(car.toString());
```

5. Сделаем проверку на поля объекта через геттеры

```
        System.out.println(car.getModel()+" "+car.getLicense()+"
"+car.getColor()+" "+car.getYear()+" "+car.getOwnerName()+"
"+car.getInsuranceNumber()+" "+car.getEngineType());
```

Итоговый код программы:

Main.java

```
 package app;

import vehicles.*;

public class Main {
    public static void main(String[] args) {
        System.out.println("Обращение к закрытым и открытым полям
объектов:");
        //Создание объектов
        Car defaultCar = new Car();
        ElectricCar electrocar = new ElectricCar();

        //Демонстрация обращения к приватным полям (только через сеттеры и
геттеры)
```

```
        defaultCar.setOwnerName("Dio");
        electrocar.setOwnerName("Jotaro");

        System.out.println("Default car owner:
"+defaultCar.getOwnerName()+"; Electrocar owner:
"+electrocar.getOwnerName());

        //Демонстрация обращения к публичным полям
        electrocar.batteryCapacity = 6;
        System.out.println("Battery capacity =
"+electrocar.batteryCapacity);

        //Задание 3
        System.out.println("\n3 часть задания:\n");

        //Создание объектов
        Vehicle car = new Car();
        Vehicle electroCar = new ElectricCar();

        //Изменение параметров через сеттеры и вывод
        car.setColor("white");
        car.setOwnerName("Mr.White");
        car.setInsuranceNumber(322);
        System.out.println(car.toString()+"\n");

        //Проверка доступа через сеттеры
        System.out.println(car.getModel()+" "+car.getLicense()+"
"+car.getColor()+" "+car.getYear()+" "+car.getOwnerName()+"
"+car.getInsuranceNumber()+" "+car.getEngineType());

        //Проаерка прямого доступа

        //System.out.println(car.model);
        //System.out.println(car.license);
        //System.out.println(car.color);
        //System.out.println(car.year);
        //System.out.println(car.ownerName);
        //System.out.println(car.insuranceNumber);
        //System.out.println(car.engineType);

        //Не работают - ошибка!
    }
}
```

Vehicle.java

```
 package vehicles;

public abstract class Vehicle {
    private String model; //(модель)
    private String license; //(номерной знак)
    private String color; //(цвет)
```

```
    private int year; //(год выпуска)
    private String ownerName; //(имя владельца)
    private int insuranceNumber; //(страховой номер)
    protected String engineType; //(тип двигателя)

    //Конструктор Vehicle
    public Vehicle(String model, String license, String color, int year,
                   String ownerName, int insuranceNumber, String engineType)
{
        this.model = model; this.license = license; this.color = color;
this.year = year;
        this.ownerName = ownerName; this.insuranceNumber = insuranceNumber;
this.engineType = engineType;
    }
    public Vehicle() {}

    //Геттеры
    public String getModel() {
        return this.model;}
    public String getLicense() {
        return this.license;}
    public String getColor() {
        return this.color;}
    public int getYear() {
        return this.year;}
    public String getOwnerName() {
        return this.ownerName;}
    public int getInsuranceNumber() {
        return this.insuranceNumber;}
    public String getEngineType() {
        return this.engineType;}

    // Сеттеры
    public void setModel(String model) {
        this.model = model;}
    public void setLicense(String license) {
        this.license = license;}
    public void setColor(String color) {
        this.color = color;}
    public void setYear(int year) {
        this.year = year;}
    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;}
    public void setInsuranceNumber(int insuranceNumber) {
        this.insuranceNumber = insuranceNumber;}
    public void setEngineType(String engineType) {
        this.engineType = engineType;}

    //Абстрактный метод vehicleType
    public abstract String vehicleType();


    @Override
    public String toString(){
```

```
        return (getClass().getSimpleName()+":"+
                "\n  model: "+this.model+
                "\n  license: "+this.license+
                "\n  color: "+this.color+
                "\n  year: "+this.year+
                "\n  ownerName: "+this.ownerName+
                "\n  insuranceNumber: "+this.insuranceNumber+
                "\n  engineType: "+this.engineType
        );
    }
}
```

Car.java

```
 package vehicles;

public class Car extends Vehicle {

    //@Override
    public String vehicleType() {
        return "Car";
    }
    //Конструктор Car использующий поля и методы родительского класса
    public Car(String model, String license, String color, int year,
               String ownerName, int insuranceNumber, String engineType) {
        super(model, license, color, year, ownerName, insuranceNumber,
engineType);
    }
    public Car() {
    }
}
```

ElectricCar.java

```
 package vehicles;

public class ElectricCar extends Car {
    public int batteryCapacity;

    //Сеттеры
    public void setBatteryCapacity(int batteryCapacity) {
        this.batteryCapacity = batteryCapacity;
    }

    //Геттеры
    public int getBatteryCapacity() {
        return batteryCapacity;
    }
```

```
    public String vehicleType() {
        return ("Electric Car");
    }

    public ElectricCar(){
        super();
    }
    public ElectricCar(String model, String license, String color, int year,
                String ownerName, int insuranceNumber, String engineType) {
        super(model, license, color, year, ownerName, insuranceNumber,
engineType);
    }
    { engineType = "Electric"; }
}
```

Результаты и выводы

Вывод программы:

```
Обращение к закрытым и открытым полям объектов:
Default car owner: Dio; Electrocar owner: Jotaro
Battery capacity = 6

3 часть задания:

Car:
  model: null
  license: null
  color: white
  year: 0
  ownerName: Mr.White
  insuranceNumber: 322
  engineType: null

null null white 0 Mr.White 322 null

Process finished with exit code 0
```

В ходе работы были закреплены основные принципы ООП: создана иерархия классов, реализовано наследование, инкапсуляция и полиморфизм. Освоена работа с пакетами, модификаторами доступа и абстрактными классами.