

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРА-
ЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра вычислительных технологий

ЛАБОРАТОРНАЯ РАБОТА №3
Дисциплина: Платформо-независимое программирование

Работу выполнил: _____ А. А. Костров

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

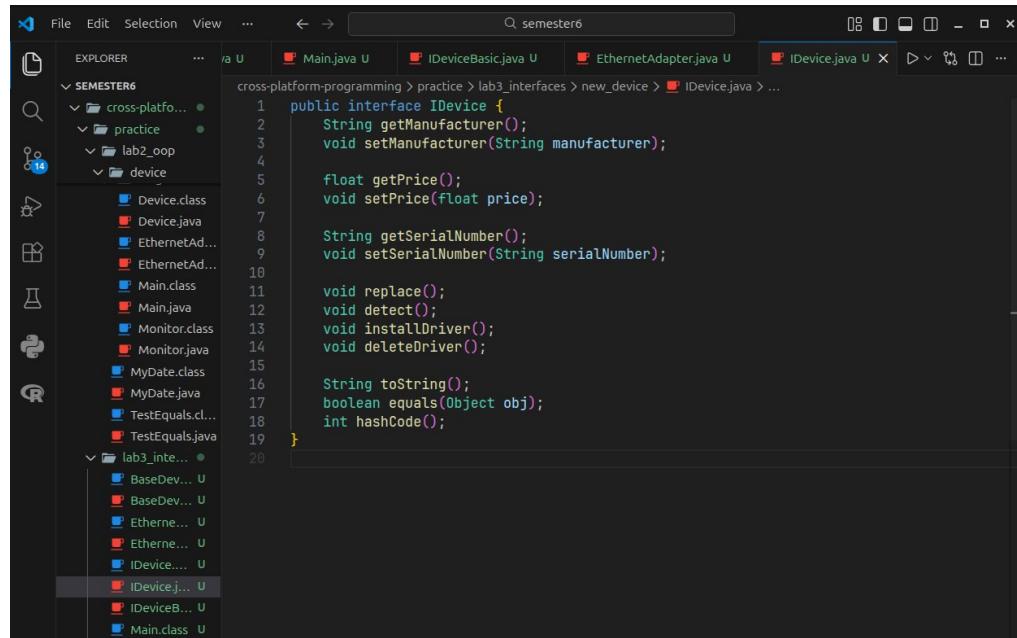
Преподаватель: _____ В. И. Шиян

Краснодар
2026

Тема. Разработка и использование интерфейсов.

Цель. Ознакомиться с понятием интерфейса, правилами разработки, наследования и способами прикладного использования интерфейсов в Java.

1. Заменим базовый класс из предыдущей лабораторной работы интерфейсом (рисунки 1-2).

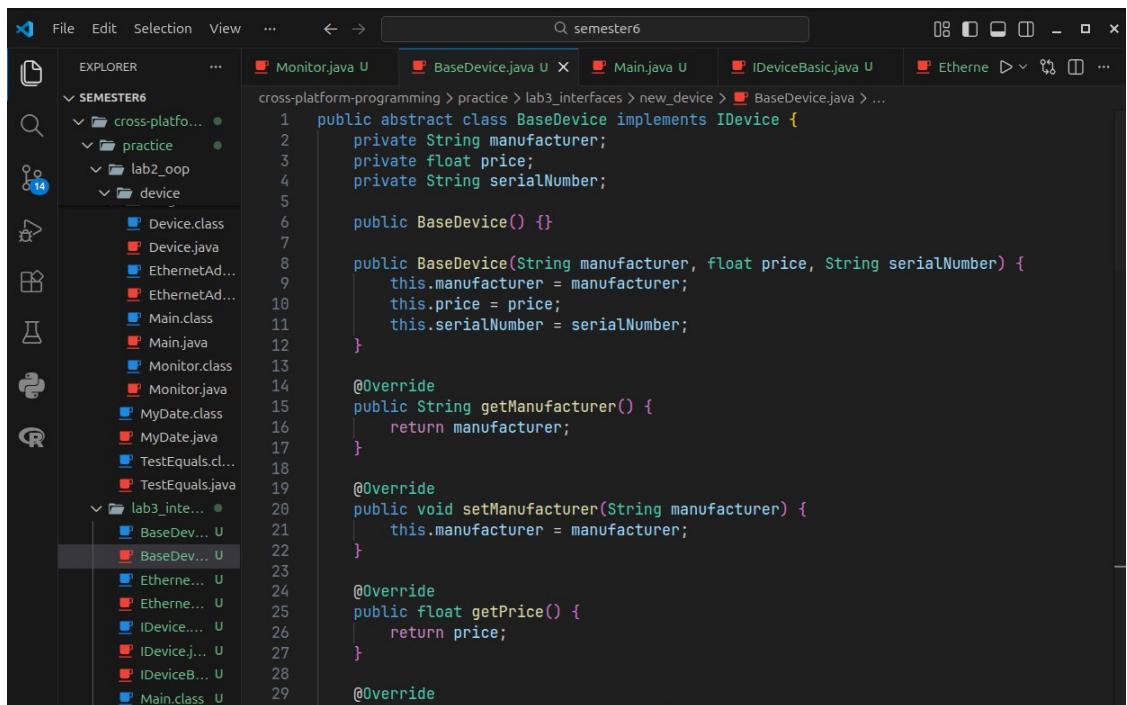


The screenshot shows a Java IDE interface with the following details:

- File Explorer:** Shows a project structure under "SEMINSTER6" with several packages and files, including "Device.class", "Device.java", "EthernetAd...", "EthernetAd...", "Main.class", "Main.java", "Monitor.class", "Monitor.java", "MyDate.class", "MyDate.java", "TestEquals.cl...", and "TestEquals.java".
- Code Editor:** Displays the content of the file "IDevice.java":

```
public interface IDevice {  
    String getManufacturer();  
    void setManufacturer(String manufacturer);  
  
    float getPrice();  
    void setPrice(float price);  
  
    String getSerialNumber();  
    void setSerialNumber(String serialNumber);  
  
    void replace();  
    void detect();  
    void installDriver();  
    void deleteDriver();  
  
    String toString();  
    boolean equals(Object obj);  
    int hashCode();  
}
```
- Status Bar:** Shows the path "cross-platform-programming > practice > lab3_interfaces > new_device > IDevice.java" and the file name "IDevice.java".

Рисунок 1 — Создаём интерфейс *IDevice*



The screenshot shows a Java IDE interface with the following details:

- File Explorer:** Shows a project structure under "SEMINSTER6" with several packages and files, including "Device.class", "Device.java", "EthernetAd...", "EthernetAd...", "Main.class", "Main.java", "Monitor.class", "Monitor.java", "MyDate.class", "MyDate.java", "TestEquals.cl...", and "TestEquals.java".
- Code Editor:** Displays the content of the file "BaseDevice.java":

```
public abstract class BaseDevice implements IDevice {  
    private String manufacturer;  
    private float price;  
    private String serialNumber;  
  
    public BaseDevice() {}  
  
    public BaseDevice(String manufacturer, float price, String serialNumber) {  
        this.manufacturer = manufacturer;  
        this.price = price;  
        this.serialNumber = serialNumber;  
    }  
  
    @Override  
    public String getManufacturer() {  
        return manufacturer;  
    }  
  
    @Override  
    public void setManufacturer(String manufacturer) {  
        this.manufacturer = manufacturer;  
    }  
  
    @Override  
    public float getPrice() {  
        return price;  
    }  
  
    @Override
```
- Status Bar:** Shows the path "cross-platform-programming > practice > lab3_interfaces > new_device > BaseDevice.java" and the file name "BaseDevice.java".

Рисунок 2 — Реализуем интерфейс

2. Разобъём наш интерфейс на два интерфейса (рисунки 3-5).

The screenshot shows a Java development environment with the following details:

- File Bar:** File, Edit, Selection, View, ..., <, >, Search bar (semester6), Minimize, Maximize, Close.
- Explorer:** SEMESTER6 folder contains cross-platfo..., practice, lab2_oop, and lab3_interfaces. The lab3_interfaces folder is expanded, showing Device.class, Device.java, EthernetAd..., EthernetAd..., Main.class, Main.java, Monitor.class, Monitor.java, MyDate.class, MyDate.java, TestEquals.cl..., and TestEquals.java.
- Editor:** The code for **IDeviceBasic.java** is displayed:

```
public interface IDeviceBasic {
    String getManufacturer();
    void setManufacturer(String manufacturer);

    float getPrice();
    void setPrice(float price);

    String getSerialNumber();
    void setSerialNumber(String serialNumber);

    String toString();
    boolean equals(Object obj);
    int hashCode();
}
```

Рисунок 3 — Создаём интерфейс *IDeviceBasic*

The screenshot shows a Java development environment with the following details:

- File Bar:** File, Edit, Selection, View, ..., <, >, Search bar (semester6), Minimize, Maximize, Close.
- Explorer:** SEMESTER6 folder contains cross-platfo..., practice, lab2_oop, and lab3_interfaces. The lab3_interfaces folder is expanded, showing Device.class, Device.java, EthernetAd..., EthernetAd..., Main.class, Main.java, Monitor.class, Monitor.java, MyDate.class, MyDate.java, TestEquals.cl..., and TestEquals.java.
- Editor:** The code for **IDeviceOperations.java** is displayed:

```
public interface IDeviceOperations {
    void replace();
    void detect();
    void installDriver();
    void deleteDriver();
}
```

Рисунок 4 — Создаём интерфейс *IDeviceOperations*

```

public abstract class BaseDevice implements IDeviceBasic, IDeviceOperations {
    private String manufacturer;
    private float price;
    private String serialNumber;

    public BaseDevice() {}

    public BaseDevice(String manufacturer, float price, String serialNumber) {
        this.manufacturer = manufacturer;
        this.price = price;
        this.serialNumber = serialNumber;
    }

    @Override
    public String getManufacturer() {
        return manufacturer;
    }

    @Override
    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    @Override
    public float getPrice() {
        return price;
    }

    @Override
    public void setPrice(float price) {
        this.price = price;
    }

    @Override
    public void detect() {
    }

    @Override
    public void installDriver() {
    }

    @Override
    public void deleteDriver() {
    }

    @Override
    public String toString() {
        return "BaseDevice{" +
            "manufacturer=" + manufacturer +
            ", price=" + price +
            ", serialNumber=" + serialNumber +
            '}';
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj instanceof BaseDevice) {
            BaseDevice other = (BaseDevice) obj;
            return manufacturer.equals(other.manufacturer) &&
                price == other.price &&
                serialNumber.equals(other.serialNumber);
        }
        return false;
    }

    @Override
    public int hashCode() {
        return Objects.hash(manufacturer, price, serialNumber);
    }

    public void replace() {
    }

    public void setSerialNumber(String serialNumber) {
    }

    public String getSerialNumber() {
        return serialNumber;
    }

    public void setPrice(float price) {
    }

    public void getSerialNumber() {
    }

    public void setReplace() {
    }

    public void getReplace() {
    }

    public void installDriver() {
    }

    public void deleteDriver() {
    }

    public String toString() {
        return "BaseDevice{" +
            "manufacturer=" + manufacturer +
            ", price=" + price +
            ", serialNumber=" + serialNumber +
            '}';
    }

    public void equals(Object obj) {
        if (this == obj) return true;
        if (obj instanceof BaseDevice) {
            BaseDevice other = (BaseDevice) obj;
            return manufacturer.equals(other.manufacturer) &&
                price == other.price &&
                serialNumber.equals(other.serialNumber);
        }
        return false;
    }

    public int hashCode() {
        return Objects.hash(manufacturer, price, serialNumber);
    }
}

```

Рисунок 5 — Реализуем оба интерфейса

3. Разработаем UML-диаграммы (рисунки 6-7).



Рисунок 6 — Иерархия с одним интерфейсом

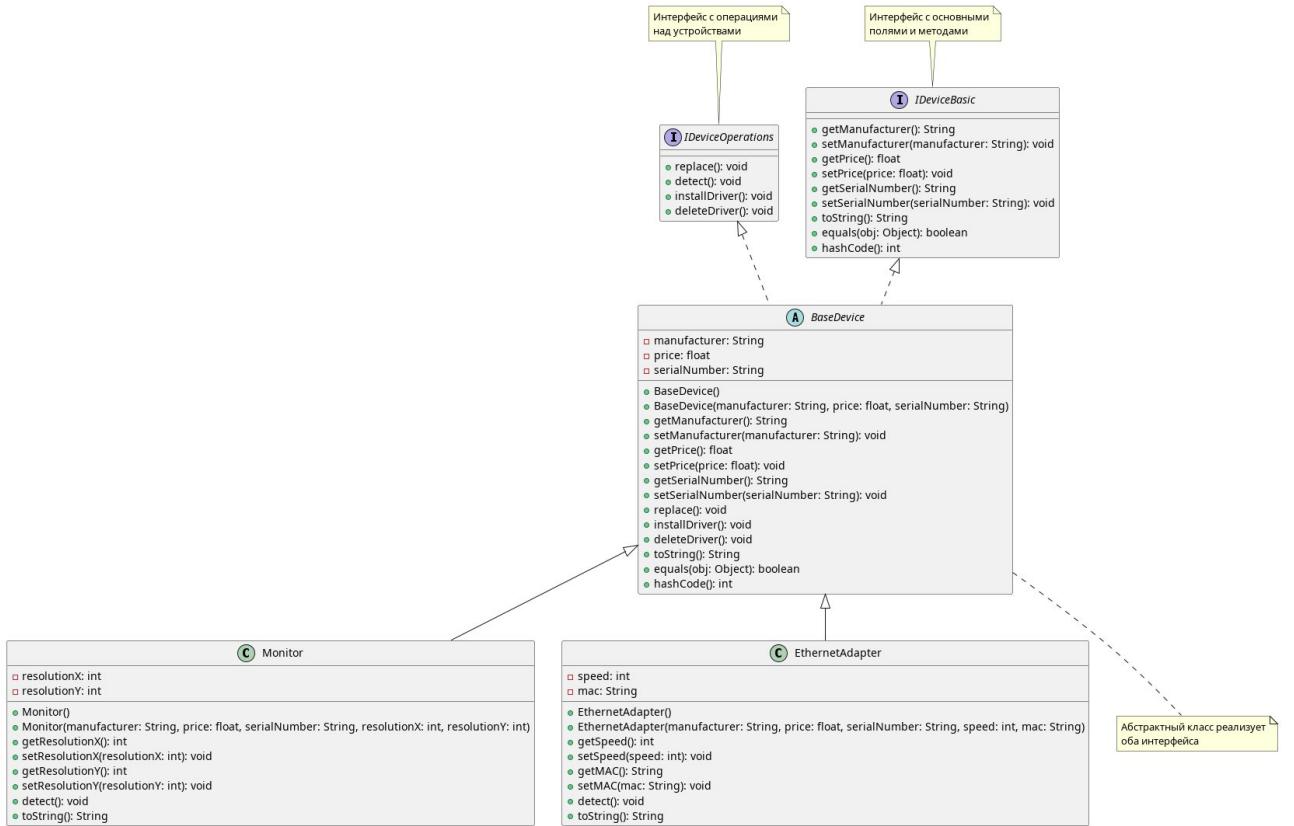


Рисунок 7 — Иерархия с двумя интерфейсами