Казанский федеральный
УНИВЕРСИТЕТ
ИНСТИТУТ
информационных технологий
и интеллектуальных систем

09. Объектно-ориентированное проектирование

курс лекций по информатике и программированию
для студентов первого курса ИТИС КФУ (java-поток)
2023/2024

М.М. Абрамский

кандидат технических наук, доцент кафедры программной инженерии

ЗАМЕЧАНИЕ О ВОСХОДЯЩЕМ ПРЕОБРАЗОВАНИИ И ПОЗДНЕМ СВЯЗЫВАНИИ (ЕЩЕ РАЗ – ЗАЧЕМ ОНИ НУЖНЫ)

- Device – электронное устройство.
  - Может включаться и выключаться (но КАК?)
  - Очевидно, работает (но КАК?)

- Phone и Camera – устройства.
  - Понятно, как включается
  - Понятно, как работает

Device

```
public abstract class Device {
    private String name;
    public String getName() {
        return name;
    }
    abstract public void on();
    abstract public void off();
    abstract public String getInfo();
}
```

Абстрактный класс с тремя абстрактными методами.

Абстрактный метод не имеет реализации – значит, чего не может быть у абстрактного класса?

Phone

```
public class Phone extends Device {

    public void on() {…}

    public void off() {…}

    public String getInfo() {
        return "Можно звонить, хранить номера,
                принимать звонки";
    }
    public void makeCall() {…}
}
```

Должны реализовать все абстрактные методы или объявить потомка тоже абстрактным.

Camera

```
public class Camera extends Device {
    public void on() {}
    public void off() {}
    public String getInfo() {
        return "Можно снимать фотографии
                просматривать их";
    }

    public void makePhoto() {}
}
```

Полиморфиииизм

```
Device[] devices = new Device[10];

for (int i = 0; i < devices.length; i += 2) {
    devices[i] = new Phone();
    devices[i+1] = new Camera();
}

for (Device device : devices) {
    System.out.println(device.getInfo());
}
```

А теперь фантазируем

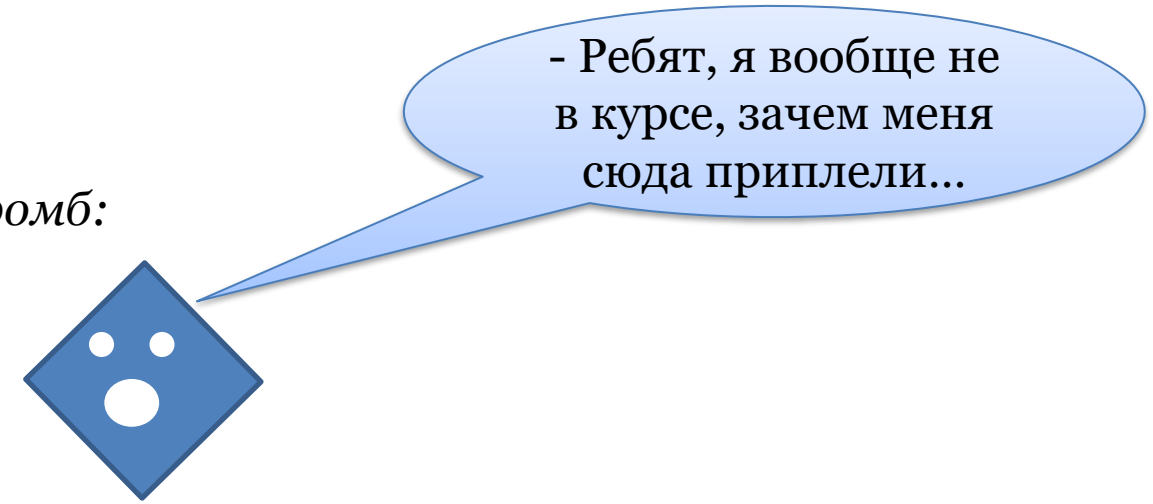
- Пусть у класса – два родителя. Что это означает?

- Наверное то, что класс наследует все, что есть в обоих родителях.

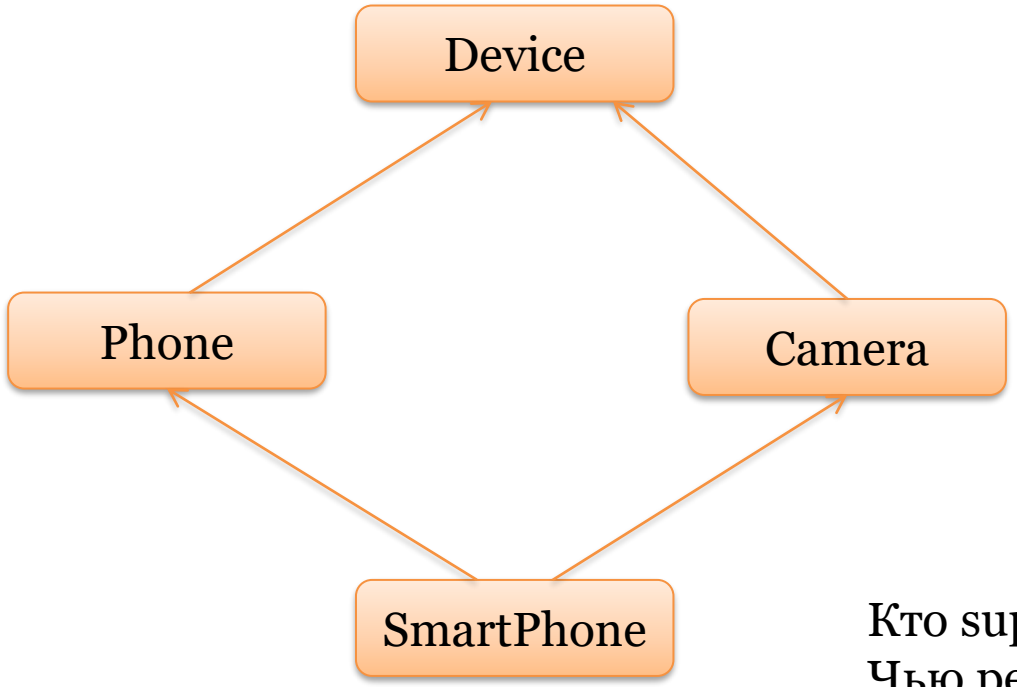
Почему нельзя множественное наследование?

– *Вот есть же Смартфон – он и телефон, и камера. В чем проблема-то?*
– *В ромбе!*

в это время ромб:

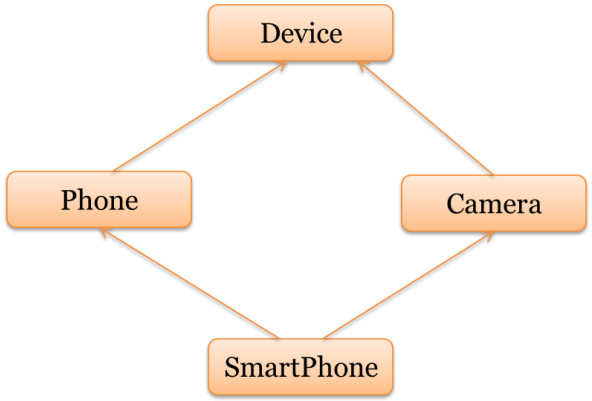


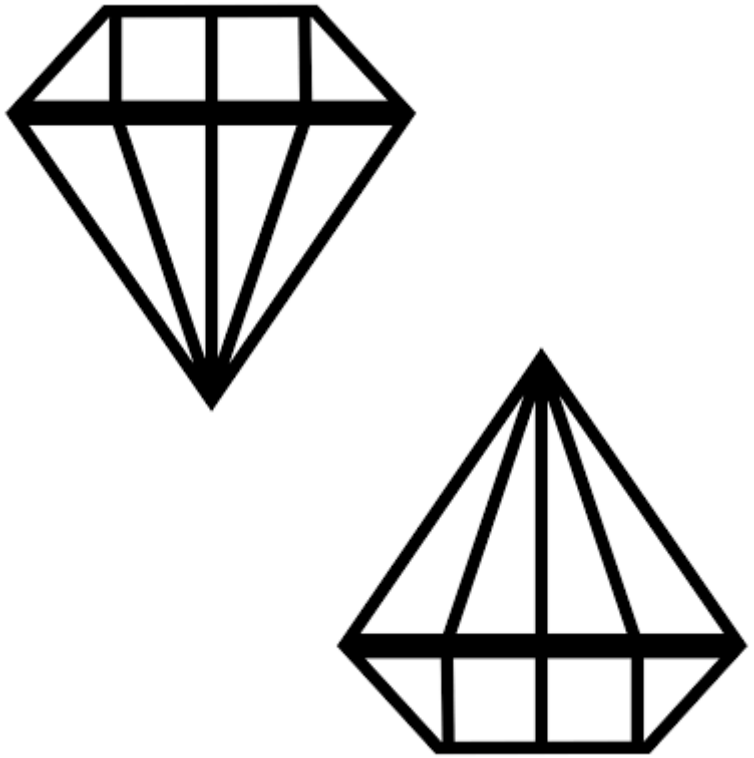
Ромб наследования



Кто super для SmartPhone?
Чью реализацию брать
getInfo? a on? a off?

Ромб и Алмаз - Diamond





Мешает реализация

- Заголовки-то одинаковые…
- … а это идея!

DeviceInterface

```
public interface DeviceInterface {
    void on();
    void off();
    String getInfo();
}
```

Все методы – public abstract (до Java 8).

Разрешены final static константы.

Модификатор – всегда public, но не пишется – не путать с default.

CallInterface и MakePhotoInterface

```
public interface CallInterface {
    void makeCall();
}
```

```
public interface MakePhotoInterface {
    void makePhoto();
}
```

Наследование интерфейсов - множественное

- Нет тела – нет проблем

```
public interface PhoneInterface
            extends DeviceInterface, CallInterface {

}
```

Реализация интерфейса в классе

Класс обязан переопределить все методы интерфейса или остаться абстрактным.

```
public class Phone implements PhoneInterface {
    public void on() {}
    public void off() {}
    public String getInfo() {
        return "Можно звонить, хранить номера, принимать звонки";
    }
    public void makeCall() {
        System.out.println("Calling");
    }
}
```

java 8 - default

- Методы по умолчанию в интерфейсе

```
interface Formula {
      double calculate(int a);
      default double sqrt(int a) {
            return Math.sqrt(a);
      }
}
```

При реализации можно не определять!
Зачем ввели – еще поговорим.

РЕМАРКА О «РЕШЕНИИ» ПРОБЛЕМЫ МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ ИНТЕРФЕЙСАМИ

Полиморфизм работает на уровне интерфейсов

Не забываем про восходящее преобразование:

```
DeviceInterface phone1 = new SmartPhone();
PhoneInterface phone2 = new SmartPhone();
MakePhotoInterface phone3 = new SmartPhone();
```

Решение проблемы множественного наследования для SmartPhone

```
public class SmartPhone extends Phone
             implements MakePhotoInterface{
    public void makePhoto() {…}

    }
}
```

или

```
public class SmartPhone
      implements PhoneInterface, MakePhotoInterface{
    public void makePhoto() {…}
    public void makeCall() {…}
    public void on() {…}
    public void off() {…}
    public String getInfo() {…}
}
```

Разница между абстрактным классом и интерфейсом

- Абстрактный класс может иметь реализованные методы
- Наследование от класса – 1, реализованы могут быть несколько интерфейсов
- Есть атрибуты у абстрактного класса
- Могут быть private, protected, default (пакет) поля и методы