



# **Основы программирования на языке Java**

## **Занятие 8. Интерфейсы**

# Интерфейс

Определяет возможное поведение объектов  
(описывает некоторое семейство типов и включается в себя только набор операций - методов)

- Интерфейс представляет собой набор методов без реализации
- При объявлении класса можно указать, какие интерфейсы он будет поддерживать (impleментировать)

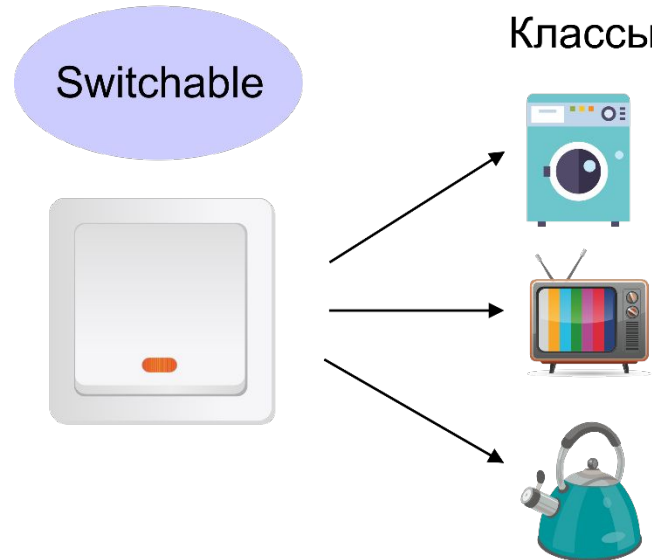
```
interface Switchable {  
    void switchOn();  
    void switchOff();  
}
```

```
class WashingMachine implements Switchable {  
    @Override  
    public void switchOn() {  
        // реализация включения  
    }  
    @Override  
    public void switchOff() {  
        // реализация выключения  
    }  
}
```

Интерфейс

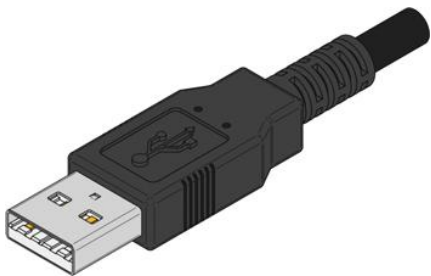
Switchable

Классы



# Объявление интерфейсов

- ✓ **Бывают:**
  - ✓ публичные (*public*)
  - ✓ непубличные – доступны внутри пакета
- ✓ **Могут включать в себя:**
  - ✓ абстрактные методы (методы без реализации)
  - ✓ статические константы
  - ✓ (Java SE 8) статические методы
  - ✓ (Java SE 8) методы по умолчанию (*default methods*) с реализацией
- ✓ **Все элементы являются публичными (*public*)**
  - ✓ все поля интерфейса являются *static* и *final*
- ✓ **Рекомендации по наименованию**
  - Имя интерфейса состоит из одного или нескольких идущих подряд слов
  - Первая буква каждого слова заглавная, остальные буквы – в нижнем регистре
  - Имя интерфейса обычно заканчивается на *'able'*



# Имплементация интерфейсов

При объявлении класса можно указать, какие интерфейсы он будет поддерживать

Класс, реализующий интерфейс:

- ✓ может иметь свои собственные методы (не объявленные в интерфейсе)
- ✓ может иметь свои собственные поля
- ✓ должен реализовать все методы интерфейса, или объявляется как **абстрактный** (*abstract*)



```
public class NewClass
    implements Interface1, Interface2, Interface3 {
    ...
}
```

# Пример



```
public interface Computable {  
    double compute();  
}
```

```
public class Summator implements Computable {  
    private double x = 0;  
    private double y = 0;  
    public Summator(double nx, double ny) {  
        x = nx;  
        y = ny;  
    }  
    @Override  
    public double compute() {  
        return x+y;  
    }  
}
```

```
public class Divider implements Computable {  
    private double x = 0;  
    private double y = 0;  
    Divider(double nx, double ny) {  
        x = nx;  
        y = ny;  
    }  
    @Override  
    public double compute() {  
        return x/y;  
    }  
}
```

# Наследование интерфейсов



```
public interface Switchable {  
    void switchOn();  
    void switchOff();  
}
```

```
public interface MediaPlayer extends Switchable {  
    void play();  
    void pause();  
    void stop();  
}
```

```
public class AudioPlayer implements MediaPlayer{  
    @Override  
    public void switchOn() {...}  
    @Override  
    public void switchOff() {...}  
    @Override  
    public void play() {...}  
    @Override  
    public void pause() {...}  
    @Override  
    public void stop() {...}  
}
```

# Abstract class vs Interface

## Абстрактные классы

- описывают поведение **для иерархии классов**
- могут реализовывать алгоритмы
- могут содержать скрытые и защищенные элементы
- класс может наследоваться только от одного абстрактного класса

## Интерфейсы

- описывают **поведение для группы классов**, реализующих данный интерфейс
- не могут реализовывать алгоритмы;
- содержат только публичные элементы
- класс может реализовывать несколько интерфейсов