Interfaces

Interface is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. In Kotlin interfaces can have methods with implementation!

In Kotlin, the way we work with *interfaces* is the same as in Java. Let's take a look at example *Interfaces.kt*:

```
interface Audio {
    fun volumeUp();
    fun volumeDown();
    fun setVolume(volume: Int)
}

class Tv : Audio {
    override fun volumeUp() {
        println("Volume up")
    }

    override fun volumeDown() {
        println("Volume down")
    }

    override fun setVolume(volume: Int) {
        println("Set volume to $volume")
    }
}
```

Interfaces can have *properties*. *Property* can be abstract or with value assigned.

```
interface Video {
    val brightness: Int

    val contrast: Int
        get() = 100

    fun play() {
        println("Play")
    }

    fun pause()
}

open class MultimediaDevice : Audio {
    override fun volumeUp() {
        println("Volume up")
    }
```

```
    override fun volumeDown() {
        println("Volume down")
    }

    override fun setVolume(volume: Int) {
        println("Set volume to $volume")
    }
}

class DvdPlayer : Tv(), Video {
    override val brightness: Int = 100

    override fun pause() {
        println("Pause")
    }
}
```