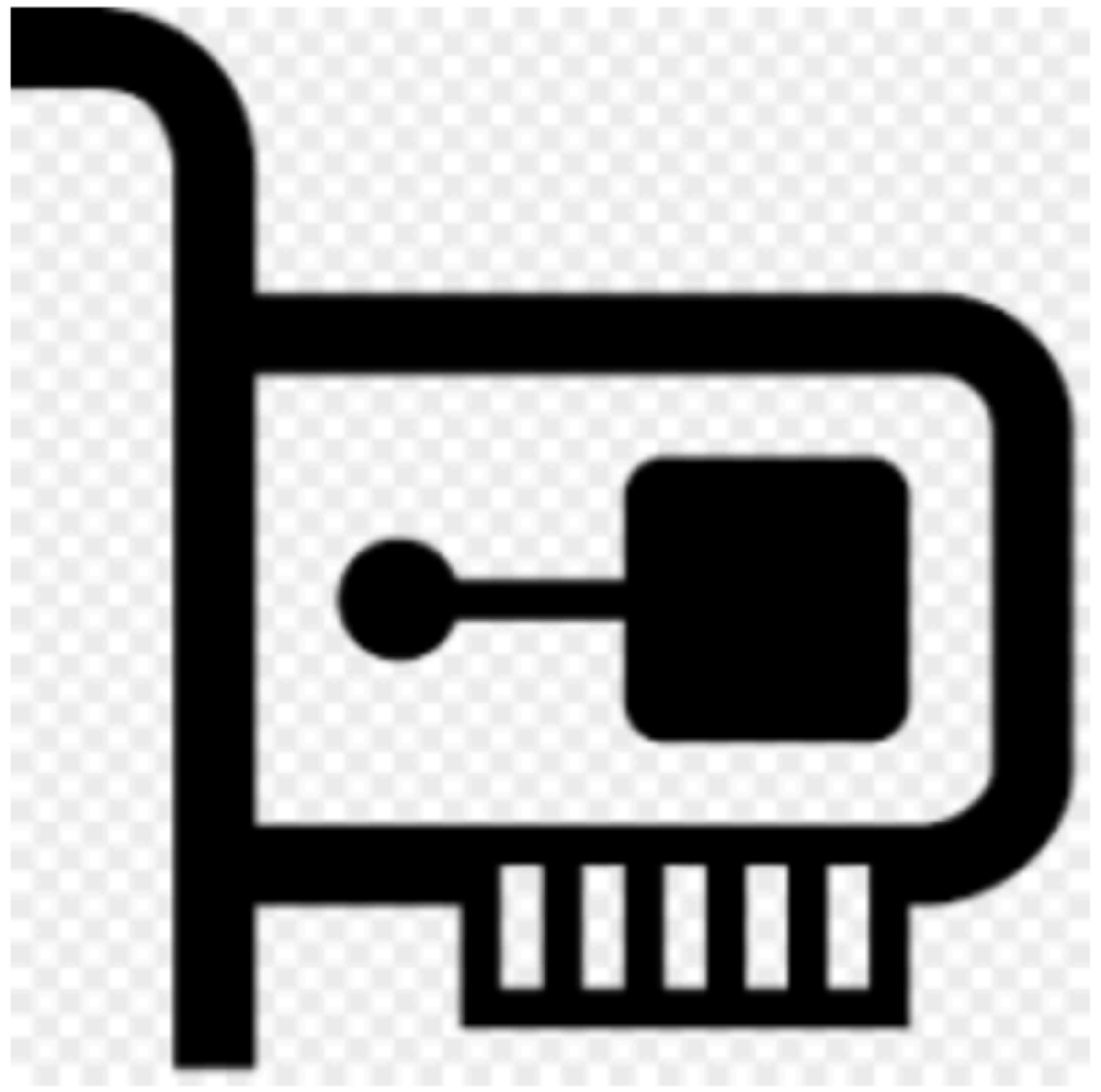
Visibility

Interfaces



Kotlin interfaces largely follow Java 8 conventions

Visibility Modifiers

Classes, objects, interfaces, constructors, functions, properties and their setters can have *visibility modifiers*. (Getters always have the same visibility as the property.) There are four visibility modifiers in Kotlin: `private`, `protected`, `internal` and `public`. The default visibility, used if there is no explicit modifier, is `public`.

Packages

Functions, properties and classes, objects and interfaces can be declared on the "top-level", i.e. directly inside a package:

```
// file name: example.kt
package foo

fun baz() { ... }
class Bar { ... }
```

- If you do not specify any visibility modifier, `public` is used by default, which means that your declarations will be visible everywhere;
- If you mark a declaration `private`, it will only be visible inside the file containing the declaration;
- If you mark it `internal`, it is visible everywhere in the same module;
- `protected` is not available for top-level declarations.

Examples:

```kotlin
// file name: example.kt
package foo

private fun foo() { ... } // visible inside example.kt

public var bar: Int = 5 // property is visible everywhere
    private set         // setter is visible only in example.kt

internal val baz = 6    // visible inside the same module
```

Classes and Interfaces

For members declared inside a class:

- `private` means visible inside this class only (including all its members);
- `protected` — same as `private` + visible in subclasses too;
- `internal` — any client *inside this module* who sees the declaring class sees its `internal` members;
- `public` — any client who sees the declaring class sees its `public` members.

NOTE for Java users: outer class does not see private members of its inner classes in Kotlin.

If you override a `protected` member and do not specify the visibility explicitly, the overriding member will also have `protected` visibility.

```kotlin
open class Outer {
    private val a = 1
    protected open val b = 2
    internal val c = 3
    val d = 4  // public by default

    protected class Nested {
        public val e: Int = 5
    }
}

class Subclass : Outer() {
    // a is not visible
    // b, c and d are visible
    // Nested and e are visible

    override val b = 5   // 'b' is protected
}

class Unrelated(o: Outer) {
    // o.a, o.b are not visible
    // o.c and o.d are visible (same module)
    // Outer.Nested is not visible, and Nested::e is not visible either
}
```

Constructors

To specify a visibility of the primary constructor of a class, use the following syntax (note that you need to add an explicit `constructor` keyword):

```
class C private constructor(a: Int) { ... }
```

Here the constructor is private. By default, all constructors are `public`, which effectively amounts to them being visible everywhere where the class is visible (i.e. a constructor of an `internal` class is only visible within the same module).

Local declarations

Local variables, functions and classes can not have visibility modifiers.