Object-oriented programming in Kotlin

Feels the same

…with some improvements

The defaults are different

`public`, `private`, `internal`

`final`, `open`, `abstract`

The defaults are different

visible in a module

internal

public, private, ~~/*package private*/~~

explicit: non-final

final, open, abstract

A module

a set of Kotlin files compiled together

- an IntelliJ IDEA module
- a Maven project
- a Gradle source set

Find the correspondence between modifiers and their meaning:

```
final
open
abstract
override
```

- overrides a member in a superclass or interface
- must be overridden (can't have an implementation)
- cannot be overridden
- can be overridden

?

Find the correspondence between modifiers and their meaning:

`final` (used by default): cannot be overridden

`open`: can be overridden

`abstract`: must be overridden (can't have an implementation)

`override` (mandatory): overrides a member in a superclass or interface

Fill the table with the values:

everywhere, in a module, in a file, in a class, in a subclass

Modifier	**Class member**	**Top-level declaration**
`public`	visible ?	visible ?
`internal`	visible ?	visible ?
`protected`	visible ?	———
`private`	visible ?	visible ?

?

Fill the table with the values:

everywhere, in a module, in a file, in a class, in a subclass

Modifier	Class member	Top-level declaration
`public`	visible everywhere	
`internal`	visible in a module	
`protected`	visible in a subclass	———
`private`	visible in a class	visible in a file

Visibility modifiers and Java

Kotlin modifier	JVM level
`public`	`public`
`private`	`private / package private`
`protected`	`protected`
`internal`	`public` & name mangling

`internal` members are mangled

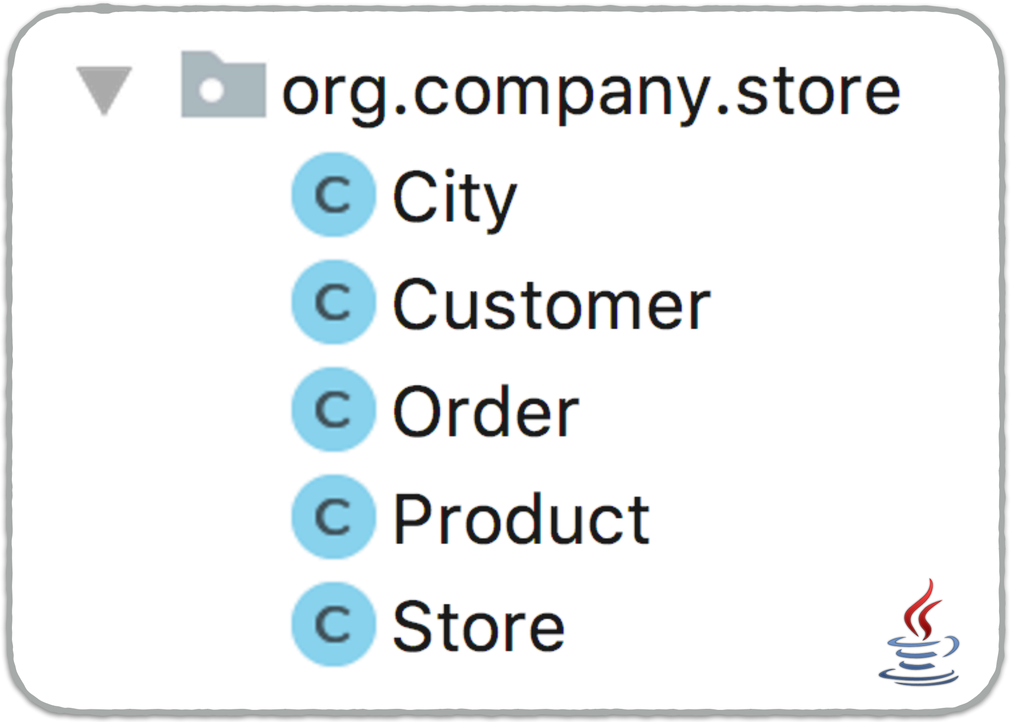
```
class MyClass {
    internal fun foo() {}
}
```

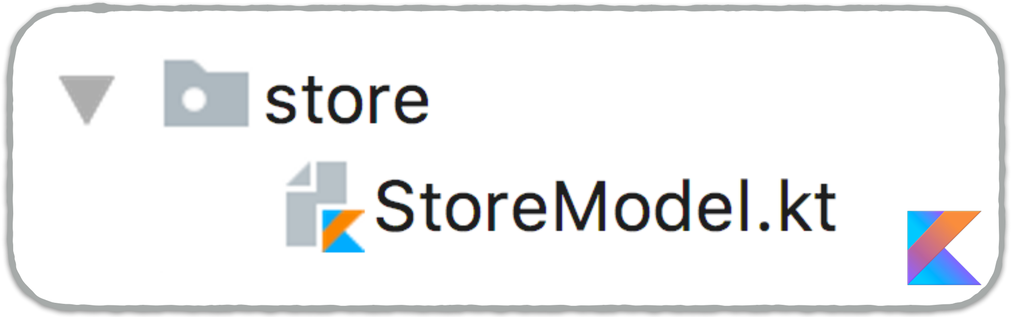
Under the hood:

```
public final class MyClass {
   public final void foo$production_sources_for_module_examples_main()
}
```

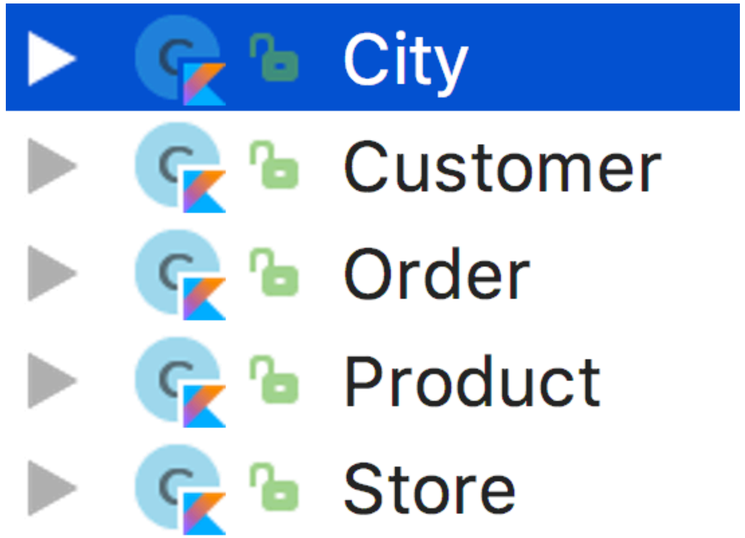
Packages

Package structure





File structure:



One file may contain several classes and top-level functions

Package structure