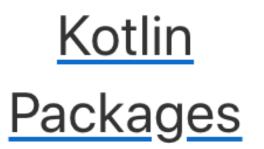
Kotlin Packages







Source files start with a package declaration. This can be used to access (import) features from the source file.

Packages

A source file may start with a package declaration:

```
package foo.bar

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

All the contents (such as classes and functions) of the source file are contained by the package declared. So, in the example above, the full name of baz() is foo.bar.baz, and the full name of Goo is foo.bar.Goo.

If the package is not specified, the contents of such a file belong to "default" package that has no name.

Default Imports

A number of packages are imported into every Kotlin file by default:

- kotlin.*
- kotlin.annotation.*
- kotlin.collections.*
- <u>kotlin.comparisons.*</u> (since 1.1)
- kotlin.io.*
- kotlin.ranges.*
- kotlin.sequences.*
- kotlin.text.*

Imports

Apart from the default imports, each file may contain its own import directives. Syntax for imports is described in the <u>grammar</u>.

We can import either a single name, e.g.

```
import foo.Bar // Bar is now accessible without qualification
```

or all the accessible contents of a scope (package, class, object etc):

```
import foo.* // everything in 'foo' becomes accessible
```

If there is a name clash, we can disambiguate by using as keyword to locally rename the clashing entity:

```
import foo.Bar // Bar is accessible
import bar.Bar as bBar // bBar stands for 'bar.Bar'
```

The import keyword is not restricted to importing classes; you can also use it to import other declarations:

- top-level functions and properties;
- functions and properties declared in <u>object declarations</u>;
- enum constants.

Unlike Java, Kotlin does not have a separate <u>"import static"</u> syntax; all of these declarations are imported using the regular <u>import</u> keyword.