Packages

Packages enable code to be managed by separating diverse classes from each other.

Packages aren't an ideal solution, and Java 9 will bring in modules, which address the problems with packages.

A package enables the creation of a unique namespace for types (classes & interfaces).

All code belongs to a package, but the package doesn't need to be explicitly named. Namespaces express semantic categories of code.

Accessing Names

Names from packages are accessed by prefixing the package name.

[…]

Import Statement

Import statement is a convenience, allowing you to write e.g. `Scanner` rather than `java.util.Scanner`.

The java.lang package is implicitly imported.

You can either import things using single-type imports (where you specify a type directly) or type-import-on-demand declarations (where you use a wildcard). Use of wildcards can slow down compilation, but is resolved at compilation time and so won't affect runtime performance.

You can also import static members of a class.

Best Practices

Import-on-demand can cause existing code to fail (e.g. if you import from multiple packages that define the same name and your code initially refers to one and then starts referring to the wrong one).

Best practice is to use single-type import only.

Naming New Packages

For private development, a package can be unnamed or have a simple name.

- It's recommended that you shouldn't use the unnamed package at all.

If code is for distributing, a unique package name is necessary to avoid name collisions.

Generating Unique Identifiers

A few options:

- Serial numbers, assigned incrementally or sequentially
- Random number, chosen from a number space much larger than the expected number of identifiers
- Names allocated from a central registry

None of these are used for java packages.

Package Naming Conventions

The `java` and `javax` packages are reserved for use for the Java language.

Organisations use their reverse internet domain name to begin their package names (or altered versions if the domain name isn't a valid Java name).

No Hierarchical Structure

You can put packages inside one another to communicate information about them, but the java language doesn't treat packages as having a structure.

Mapping Packages

A package can be stored in a file system or in a database.

Packages in file systems may be required to be organised in specific ways to allow classes to be easily found. This is needed for compilation and also for running code.

Each host system determines how packages and compilation units are created and stored.

Packages may be represented by e.g. the directory structure the files are placed in.

Package Access

Access specifiers have implications for access to packages.

Members specified as `private` can only be accessed from the class – all others can be accessed by every class in the same package, along with possibly some from different packages.