`private[X]` where X is the directly enclosing package.

The Scala member hides the Java one.

They make no sense because singletons have no subclasses.

`private[this]`

It's only accessible by this instance of a class, not others.

- Scala's are hierarchical like Java's, but they also properly nest.
- Java's package statement in Scala is just syntactic sugar for

```
package X {
  //compilation unit
}
```

Scala is more consistent, denying visibility to containing instances of nested classes.

Scala gives only subclass access, not package members too.

It is the reliance of various parts of the program on one another.

`private[X]` where X is the containing class.

`private[X]` where X is the outermost class.

In curly braces, allows
- Simple name.
- Renaming clause `x => y`.
- Hiding clause `x => _`.
- catch-all _, all members not in previous clauses, must be last.

You can leave off the curlies if only importing 4 or one of 1.

`import fruit._`, where fruit is the param to a method.

It's useful when you use objects as modules.

- From the normal root of the hierarchy.
- From __root__ package to force absolute lookup when inner scope masks.
- Relative to current package.

In curly braces. allows:
1) Simple name.
2) Renaming clause `x => y`
3) Hiding clause `x => _`
4) Catch-all _, all members not in previous clauses, must be last.

You can leave off curlies if you're only importing #4 or one of #1.