Access Levels for Interfaces

Source http://www.artima.com/objectsandjava/webuscript/PackagesAccess1.html

Access Levels for Interfaces

Interfaces have slightly different rules for access levels, because every field and method defined by an interface is implicitly public. You can't use the keywords `private` or `protected` on the fields and methods of interfaces. If you leave off the `public` keyword when declaring interface members, as is officially recommended by the Java Language Specification, you do not get package access. You still get public access. Therefore, you can't hide any implementation details of a package inside an interface (You can't hide an interface's members). On the other hand, you can hide the entire interface. If you don't declare an interface public, the interface as a whole will only be available to other types in the same package. As with classes, you should make interfaces public only if they are needed by classes and interfaces defined in other packages.

Here's an example of two interfaces. Interface `Soakable` is part of the internal implementation of a package. Interface `Washable` is part of the external implementation of the package:

```
// In Source Packet in file
// packages/ex5/com/artima/vcafe/dishes/Washable.java
package com.artima.vcafe.dishes;

public interface Washable {

    void wash();
}

// In Source Packet in file
// packages/ex5/com/artima/vcafe/dishes/Soakable.java
package com.artima.vcafe.dishes;

interface Soakable extends Washable {

    void soak();
}
```

In this example, `wash()` and `breakIt()` are not explicitly declared public, because they are public by default. Because the `Washable` interface as a whole is not explicitly declared as public, however, it has package access. Interface `Washable` is only be accessible to other types declared in the `com.artima.vcafe.dishes` package. Interface `Breakable`, because it is declared as public, is available to any type declared in any package.

*~~~ End of Article ~~~*