Packages are containers for classes. They are used to keep the class name space compartmentalized.

For example, a package allows you to create a class named List, which you can store in your own package without

concern that it will collide with some other class named List stored elsewhere. Packages are stored in a hierarchical

manner and are explicitly imported into new class definitions.

The package is both a naming and a visibility control mechanism.

The following statement creates a package called MyPackage: package MyPackage;

Java uses file system directories to store packages. For example, the .class files for any classes you declare to be

part of MyPackage must be stored in a directory called MyPackage. Remember that case is significant, and the directory

name must match the package name exactly.

A package hierarchy must be reflected in the file system of your Java development system.

For example, a package declared as

```
package java.awt.image;
```

needs to be stored in java\awt\image in a Windows environment. Be sure to choose your package names carefully.

You cannot rename a package without renaming the directory in which the classes are stored.

How does the Java run-time system know where to look for packages that you create? The answer has three parts.

- First, by default, the Java run-time system uses the current working directory as its starting point.

Thus, if your package is in a subdirectory of the current directory, it will be found.

- Second, you can specify a directory path or paths by setting the CLASSPATH environmental variable.

- Third, you can use the -classpath option with java and javac to specify the path to your classes.

When a package is imported, only those items within the package declared as public will be available to non-subclasses

in the importing code.