Anexo II.- Paquetes e imports en Java

Un paquete (package) es un contenedor de un grupo de clases, interfaces (interface), etc. relacionadas entre sí.

Las ventajas de organizar los archivos en paquetes son:

- Agrupar clases que tienen algo en común.
- Reutilización de código.
- Seguridad gracias a los modificadores de acceso.

Aunque volveremos sobre los paquetes más adelante, vamos a ver unas pinceladas interesantes.

Una clase puede hacerse miembro de un paquete mediante la cláusula **package**, la cual deberá ser la primera línea de código. Ejemplo:

```
package mipaquete;
class MiClase {
        …
}
```

Tenemos la clase MiClase del paquete mipaquete, de hecho cualquier otra clase declarada en este mismo archivo de código fuente pertenecerá también a mipaquete.

Recuerda:

- Java es sensible a mayúsculas y minúsculas. Por tanto "mipaquete" no es el mismo que "MiPaquete".
- No podemos tener en un mismo paquete dos clases con el mismo nombre, pero si en diferentes paquetes.

Los paquetes están representados físicamente en el sistema, la jerarquía de paquetes y sub-paquetes corresponde a la existencia de sus correspondientes directorios y sub-directorios, por ejemplo el paquete mipaquete.misubpaquete corresponde al directorio mipaquete / misubpaquete /, por lo tanto podemos decir lo siguiente: Paquete = Directorio.

Para la ejecución del programa la JVM se encargará de buscar los ficheros java correspondientes en los directorios según la jerarquía indicada a través de las cláusulas **package**.

Algunos paquetes estándar más comunes son:

Paquete	**Descripción**
java.applet	Contiene las clases necesarias para crear applets que se ejecutan en la ventana del navegador web.
java.awt	Contiene las clases para crear interfaces de usuario nativas del sistema.
java.io	Entrada/Salida. Clases que definen distintos flujos de datos.
java.lang	Clases esenciales de Java. Se importa implícitamente sin necesidad de una sentencia import.
java.net	Se utiliza en combinación con las clases del paquete java.io para leer y escribir datos de la red.

java.swing	Clases para crear interfaces de usuario independientes del sistema.
java.util	Contiene otras clases con utilidades que ayudan al programador.

La sentencia **import** permite referirse a las clases de un paquete sin tener que poner el nombre canónico completo. Es decir, si hacemos el siguiente import:
import java.util.Scanner;
Nos permite poder utilizar después solo el nombre de la clase de la siguiente forma:
Scanner lector = new Scanner(System.in);
En lugar de tener que poner el nombre completo del paquete cada vez:
java.util.Scanner lector = new java.util.Scanner(System.in);

Cuando hacemos un import podemos:

- Importar una clase individual. Ejemplo: *import java.awt.Font;*
- Importar un conjunto de clases declaradas dentro de un paquete, haciendo uso de un asterisco (*). Ejemplo: *importe java.awt.*;*

Consideraciones:

- La interacción entre diferentes paquetes se ve afectada por los modificadores de acceso, por eso una de las ventajas del uso de paquetes es la seguridad.
- La cláusula import no funciona como el "include" de C que incluye más código, sino que solo indica al compilador la ubicación del código externo que queremos utilizar, es más como un acceso directo puesto que como hemos visto, ni siquiera es obligatorio usarlo, simplemente podríamos escribir en nuestro código las ubicaciones completas del código externo lo cual podría volverse repetitivo y nos dejaría un código más extenso. De aquí la ventaja práctica al usar import.
- Los IDEs como por ejemplo NetBeans, Eclipse, IntelliJ nos facilitan notoriamente la tarea de declarar paquetes y hacer importaciones.