Cómo trabajar con paquetes

En Java podemos agrupar varias clases en lo que se llama un "`package`".

Si en un proyecto hay muchas clases, los paquetes nos permiten organizarlas agrupándolas bien por finalidad, por ámbito o por herencia, lo que nos facilitará encontrar una clase determinada cuando lo necesitemos.

Además, al utilizar paquetes se evitan los conflictos de nombres entre clases y se facilita que puedan ser utilizadas por otros programadores.

Un paquete puede almacenar una o más clases y también interfaces. Las clases tienen ciertos privilegios de acceso a los miembros dato y a las funciones miembro de otras clases **dentro de un mismo paquete**.

Un proyecto nuevo se crea en un subdirectorio que tiene el nombre del proyecto y a continuación se crea la aplicación. Al organizar una aplicación en paquetes, todas las clases se guardan en un paquete concreto que no sea el `default package`. Esto incluye a la clase que contiene el método `main` de la aplicación, que se suele guardar en un directorio con nombre similar al nombre de la aplicación.

Cuando guardamos una clase en un paquete, la primera sentencia de la clase debe ser la que especifica el nombre del paquete.

Después de la sentencia `package` se escriben las sentencias `import` que sean necesarias para la clase que estamos creando

```
//archivo MiApp.java

package nombrePaquete;
public class MiApp{
        //miembros dato
        //funciones miembro
}
//archivo MiClase.java

package nombrePaquete;
public class MiClase{
        //miembros dato
        //funciones miembro
}
```

La palabra reservada *import*

La API standard de Java tiene unos 200 `packages`.

El paquete `java.util` contiene unas 50 clases, entre ellas la clase `Scanner`. Esta clase tiene dos nombres:

- El nombre **completo cualificado**: `java.util.Scanner`. obtenemos este nombre poniendo el nombre del paquete delante del nombre de la clase.
- El nombre **simple**: `Scanner`.

Una sentencia `import` nos permite abreviar el nombre de la clase. Si escribimos

```
import java.util.Scanner
```

en la primera línea del archivo de una clase, en el resto del código podemos usar el nombre simple para referirnos a ella.

La clase java.lang.System

El paquete `java.lang` contiene unas 35 clases entre las que se encuentra la clase `System`, cuyo nombre completo cualificado es `java.lang.System` y el simple `System`. Pero siempre podemos usar el nombre simple porque el paquete `java.lang` se importa implícitamente sin necesidad de una sentencia **import**.

Los paquetes estándar son

Paquete	Descripción
`java.applet`	Contiene las clases necesarias para crear applets que se ejecutan en la ventana del navegador
`java.awt`	Contiene clases para crear una aplicación GUI independiente de la plataforma
`java.io`	Entrada/Salida. Clases que definen distintos flujos de datos
`java.lang`	Contiene clases esenciales, se importa impícitamente sin necesidad de una sentencia **import**.
`java.net`	Se usa en combinación con las clases del paquete java.io para leer y escribir datos en la red.
`java.util`	Contiene otras clases útiles que ayudan al programador

El atributo estático System.out

En la expresión `System.out`, out es un **atributo** estático de la clase System. cuando la máquina virtual de Java se inicia, se invoca al método **`initializeSystemClass()`**, que hace exactamente lo que dice su nombre y , a su vez, llama al método `setOut()` para inicializar el atributo `out`.

Dentro de la clase System la declaración de out es como sigue:

```
//la clase System pertenece al java.lang package
class System {
  public static final PrintStream out;//objeto de la clase PrintStream
  //...
}

// la clase Prinstream pertenece al java.io package
class PrintStream{
  public void println();
//...
}
```

Por eso escribimos `System.out.println()` y, si lo queremos escribir de forma abreviada tenemos que hacer la siguiente importación:

```
import static java.lang.System.out;
```

Ahora ya podemos escribir `out.println();`