Paquetes y Visibilidad

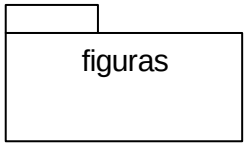
Material de clase elaborado por Sandra Victoria Hurtado Gil

A medida que los programas en Java van incluyendo más clases se tiene la necesidad de organizar esas clases para facilitar su uso. Esto se hace de manera similar a como se organizan los archivos en un computador: creando “carpetas” para guardar cada archivo en la carpeta que le corresponda. En Java, el equivalente a estas carpetas son los paquetes.

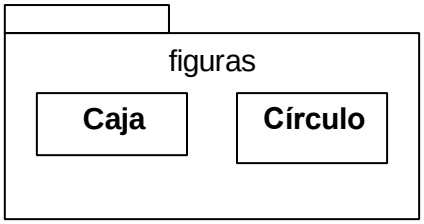
Un **paquete** es un mecanismo para agrupar las clases, de manera similar a como una carpeta guarda varios archivos. Por lo general, las clases que se agrupan en un paquete tienen funciones similares o relacionadas.

Por convención los nombres de los paquetes solo deben contener letras minúsculas, incluso si están formados por varias palabras. Por ejemplo, un nombre adecuado para un paquete sería “mipaquete”, y no “miPaquete”. Además, se recomienda que no tenga números al inicio y que no incluya caracteres extraños.

En los diagramas de clases un paquete se representa con un rectángulo que tiene un recuadro de menor tamaño en la parte superior izquierda, así:



Las clases se muestran dentro del paquete, por ejemplo:



Además de servir para organizar las clases, los paquetes ayudan a:

- Garantizar nombres únicos para las clases.
- Definir permisos para que otras clases puedan usar (o no) atributos y métodos de una clase.

1. Nombres únicos

En Java cada clase debe tener un nombre único, sin embargo, esto es difícil de lograr cuando los programas crecen, cuando hay varios desarrolladores o cuando se usan clases de otros proyectos. La solución para que se puedan tener clases con el mismo nombre es usar los paquetes. Los paquetes permiten asignar nombres únicos a las clases, porque es posible tener clases que se llamen igual, siempre y cuando estén en paquetes diferentes. Por este motivo algunas veces los paquetes se asocian con la idea de un “espacio de nombres”.

Para garantizar que cada elemento de un programa en Java sea fácilmente identificable, se define que **el nombre completo** de las clases y paquetes está compuesto por el nombre del paquete al que pertenece, seguido de un punto y luego su nombre. De esta forma se garantizan nombres únicos.

Por ejemplo, y haciendo referencia al diagrama de clases anterior:

- El nombre completo de la clase Caja es: “figuras.Caja”.
- El nombre completo de la clase Círculo es: “figuras.Círculo”.

En un programa se puede usar este nombre completo para referirse a la clase. Por ejemplo:

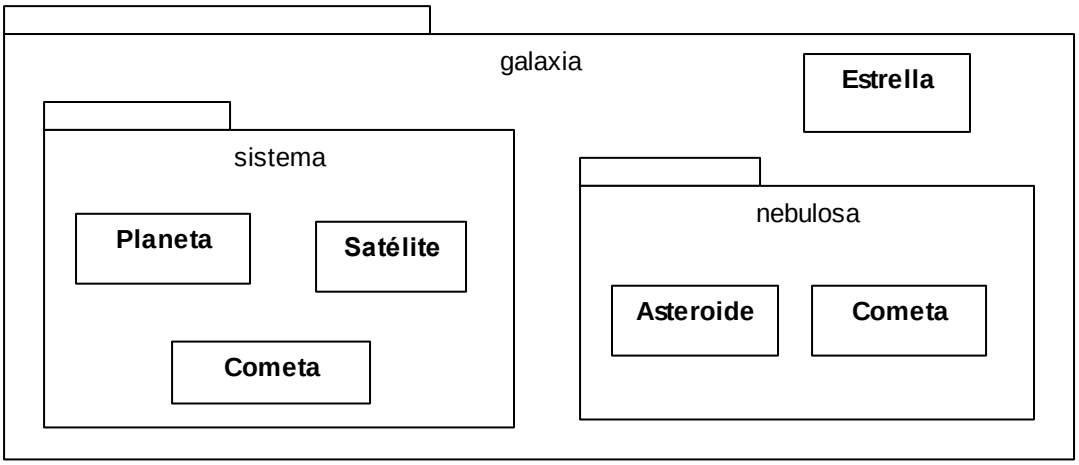
```
/**
 * Muestra el uso del nombre completo de una clase
 * @version 1.0
 */
public class UsoNombreCompletoClase  {

       // Crea un objeto Caja, usando el nombre completo de la clase.
      void crearObjeto() {
            figuras.Caja caja = new figuras.Caja();
      }
}
```

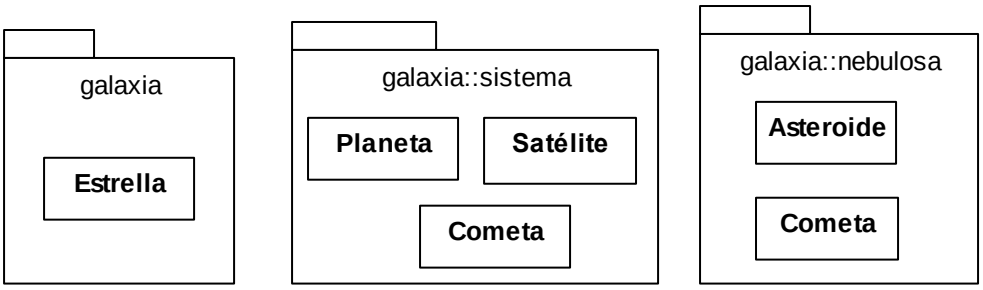
Sin embargo, usar los nombres completos puede ser algo tedioso. Generalmente usamos el nombre simple de las clases para facilitar la escritura de los programas, lo cual es posible gracias al mecanismo de “importar” que tiene Java, el cual se explicará más adelante.

Java permite tener paquetes dentro de paquetes, pero esto solo influye en el nombre. Es decir, aunque se tengan paquetes dentro de otro, **cada uno de estos paquetes internos se comporta como un paquete independiente**.

Por ejemplo, el siguiente diagrama muestra un paquete formado por una clase y dos su-paquetes, cada uno de estos con otras clases:



Sin embargo, para Java, esto es equivalente a tener tres paquetes independientes, así:



Con respecto a los nombres de algunas clases en la figura anterior:

- El nombre completo de Estrella es: “galaxia.Estrella”
- El nombre completo de Planeta es: “galaxia.sistema.Planeta”
- Se tienen dos clases llamadas Cometa, pero son diferentes, porque una está en el paquete “galaxia.sistema” y otra está en el paquete “galaxia.nebulosa”. Los nombres completos son: “galaxia.sistema.Cometa”, y “galaxia.nebulosa.Cometa”.

2. Paquetes en Java

Si se quiere indicar que una clase pertenece a un paquete se debe utilizar la palabra reservada ***package*** seguida del nombre del paquete. Esta debe ser la primera instrucción en la clase, antes de cualquier otra instrucción, así:

```
package nombrepaquete;

public class Clase {
      // código clase
}
```

Por ejemplo, para la clase Estrella y la clase Planeta del diagrama de clases anterior:

```
package galaxia;

public class Estrella  {
    //código clase
}
```

```
package galaxia.sistema;

public class Planeta {
   //código clase
}
```

3. Uso de clases de otros paquetes (importar)

Cuando las clases están en el mismo paquete no es necesario dar el nombre completo para usarlas; pero cuando están en otro paquete sí es necesario usar una de dos formas para que Java las pueda encontrar:

- Dar su nombre completo: "paquete.clase", cada vez que se haga referencia a ella,
  o
- Utilizar la sentencia *import*.

Cuando se incluye la sentencia *import*, se está indicando al programa que debe buscar en otro paquete. La sentencia *import* tiene la siguiente forma:

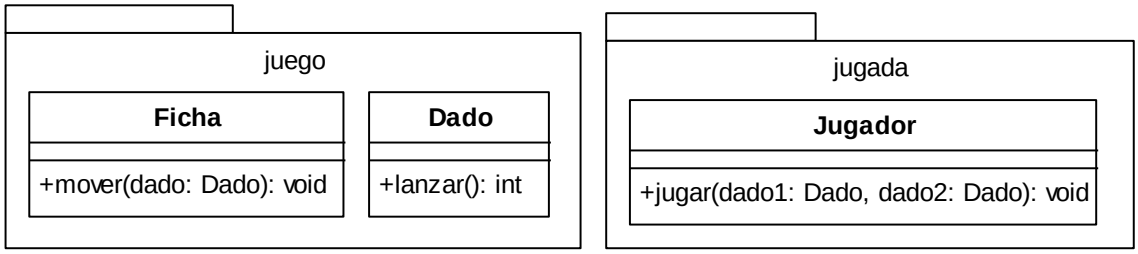
```
import nombre;
```

El nombre se puede dar de dos maneras:

- Dar el nombre completo de la clase -para llegar directamente a ella, o
- Dar el nombre del paquete y luego un asterisco, para que busque en todas las clases del paquete.

Las sentencias *import* van después de la sentencia *package*, y antes del encabezado de la clase. Si es necesario incluir varias clases o varios paquetes se debe usar **un *import* para cada uno**.

Por ejemplo, se tiene el siguiente diagrama:



Como puede verse, se necesita usar objetos Dado, tanto en la clase Ficha como en la clase Juego.
En la clase Ficha se puede usar directamente la clase Dado, sin usar un *import*, porque están en el mismo paquete. El código es:

```
package juego;
/**
 * Una ficha de un juego, que usa un dado para avanzar.
 * @version 1.0
 */
public class Ficha {
	/**
	 * Mover una ficha, el número de veces que saque el dado
	 * @param   unDado      el dado para obtener el número
	 */
	public void mover(Dado unDado) {
		int casillas = unDado.lanzar();
		System.out.println("Se mueve "+casillas+ " casillas");
	}
}
```

Sin embargo, en la clase Juego se debe usar un *import* para poder usar la clase Dado porque están en diferentes paquetes. El código es:

```
package jugada;

import juego.Dado;

/**
 * Un jugador que practica con dos dados (clase que está en otro paquete)
 * @version 1.0
 */
public class Jugador {
	/**
	 * Juego que compara los números que salen en dos dados
	 * @param   dado1       el primer dado participante
	 * @param   dado2       el segundo dado participante
	 */
	public void jugar(Dado dado1, Dado dado2) {
		int valor1 = dado1.lanzar();
		int valor2 = dado2.lanzar();

		if (valor1 > valor2) {
			System.out.println("Ganó el primer dado");
		}
		else if (valor2 > valor1) {
			System.out.println("Ganó el segundo dado");
		}
		else {
			System.out.println("Empataron");
		}
	}
} // fin clase Jugador
```

Se usa el *import* para poder trabajar con la clase Dado, porque está en otro paquete

4. Visibilidad

Agrupar las clases en paquetes, no sólo permite una mejor organización, sino que también tiene efectos sobre el acceso a las clases y los miembros de una clase (atributos y métodos).

En Java hay cuatro niveles o tipos de visibilidad:

- Público: los miembros definidos como públicos pueden ser usados por TODAS las demás clases del programa.
  La palabra en Java para este nivel de visibilidad es: *public*
  Por ejemplo:

```
public void unMetodo()        // un método público
```

- Protegido: los miembros protegidos pueden ser usados por todas las clases que **estén en el mismo paquete y las clases hijas.**
  La palabra en Java para este nivel de visibilidad es: *protected*
  Por ejemplo:

```
protected void unMetodo()     // un método protegido
```

- De paquete, amigable u omitido: los miembros con visibilidad de paquete solo pueden ser usados por otras clases que **estén en el mismo paquete**.
  No hay una palabra en Java para este nivel de visibilidad. Cuando no se coloca ningún nivel de visibilidad, Java identifica ese elemento como amigable o de paquete. Como no se tiene ningún nombre, a este nivel de visibilidad en Java también se le llama "omitido". Por ejemplo:

```
void unMetodo()               // un método amigable o de paquete
```

- Privado: Solo se pueden usar dentro de la clase donde están definidos. **No** pueden ser usados por otras clases.
  La palabra en Java para este nivel de visibilidad es: *private*
  Por ejemplo:

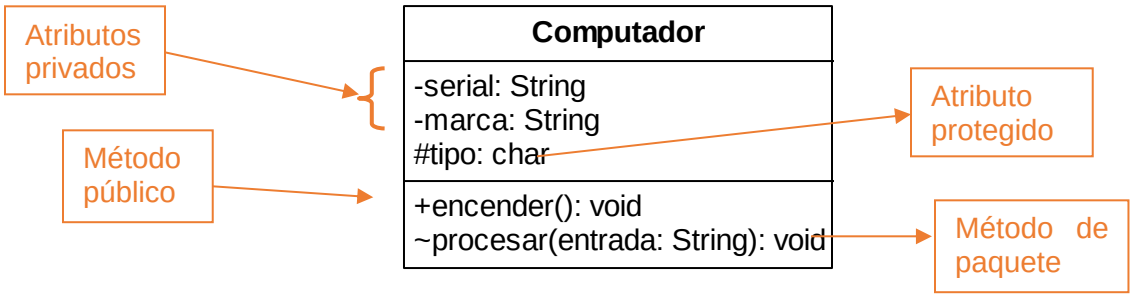
```
private void unMetodo()       // un método privado
```

Por lo tanto, los paquetes permiten definir niveles de acceso, para garantizar así más control sobre el código.

La forma de representar la visibilidad en el diagrama de clases es:

- privado
+ público
# protegido
~ de paquete

Por ejemplo:



Supongamos que tenemos la clase Vehículo, representada en el siguiente diagrama:

Vehículo
capacidadTanque: double cantidadGasolina: double
Vehículo(capacidadTanque: double) adicionarGasolina(cantidad: double): boolean

En el método “adicionarGasolina” se valida que la cantidad no supere la capacidad del tanque. Sin embargo, como no se ha establecido la visibilidad, los atributos pueden ser modificados en cualquier momento desde otros objetos, lo cual no es conveniente.

Por ejemplo, se puede tener el siguiente código, de otra clase:

```
/**
 * Un conductor despistado que desea adicionar más gasolina
 * @version 1.0
 */
public class Conductor {
   public static void main(String arg[]) {
         Vehiculo carro1 = new Vehiculo(10);
         boolean pudo;
         pudo = carro1.adicionarGasolina(5);      //BIEN, pudo=true
         pudo = carro1.adicionarGasolina(6);      //BIEN, pudo=false
         carro1.cantidadGasolina = 11;    //MAL: cambia sin validar
   }
}
```

Para evitar que esto suceda los objetos pueden ocultar sus atributos, usando la visibilidad.
En la clase Vehículo se definirán los atributos como privados, para que desde ninguna otra clase se tenga acceso a ellos. Los métodos, por otra parte, se definirán como públicos, para que sí puedan ser usados. El diagrama se modifica de la siguiente forma:

Vehículo
-capacidadTanque: double -cantidadGasolina: double
+Vehículo(capacidadTanque: double) +adicionarGasolina(cantidad: double): boolean

El código de la clase queda:

```
/**
 * Medio de transporte al cual se le adiciona gasolina que funcione
 * @version 1.0
 */
public class Vehiculo {
    private double capacidadTanque;
    private double cantidadGasolina;

    /**
     * Constructor de objetos Vehículo
     * @param  capacidadTanque   la capacidad del tanque, en litros
     */
    public Vehiculo(double capacidadTanque) {
         this.capacidadTanque = capacidadTanque;
         this.cantidadGasolina = 0;
    }

    /**
     * Adicionar gasolina, máximo hasta la capacidad del tanque
     * @param  cantidad    litros de gasolina que se desean adicionar
     * @return si pudo o no adicionar la cantidad.
     */
    public boolean adicionarGasolina(double cantidad) {
         double maximo = this.capacidadTanque - this.cantidadGasolina;
         if (cantidad <= maximo) {
               this.cantidadGasolina += cantidad;
               return true;
         }
         else {
               return false;
         }
    }
} // fin clase Vehiculo
```

De esta manera se resolvió el problema de que se pudiera modificar un atributo de manera incorrecta desde otros objetos. El código que se presentó al comienzo ya no permite hacer el cambio en el atributo, porque es privado:

```
        // Lo siguiente ya NO COMPILA:
        carro1.cantidadGasolina = 11;
```

Algunas observaciones adicionales:

- El nivel de visibilidad es diferente al uso del *import*. El *import* es solo para decir dónde está la clase, pero tener permiso de usar los atributos y métodos de esa clase es diferente. Eso lo define el nivel de visibilidad que tenga cada uno.
- Cuando no se define ningún paquete para las clases, Java las coloca en un “paquete por defecto” (sin nombre). Esto puede servir para programas pequeños, pero no es práctico para programas grandes porque se empiezan a presentar problemas de ubicación de las clases.
- Las clases pueden tener visibilidad pública o de paquete. No se puede usar visibilidad protegida ni privada para las clases[1].

[1] Hay algunas excepciones, pero eso se escapa del alcance de este material.