Tema 9: Paquetes

Antonio J. Sierra

Índice

1. Introducción.
2. Protección de acceso.
3. Importación de paquetes.
4. Modelado UML de un paquete.

Introducción

- Justificación
  - Unicidad en el identificador de la clase.
  - Asociarlo a un espacio de nombres.
  - Unicidad en el espacio de nombres.
- Mecanismo para organizar de forma estructurada las clases.
- Todas las clases incorporadas en Java se almacenan en paquetes.

Introducción

- Son contenedores de clases
- Utilizan el mismo espacio de nombres para dividir compartimentos.
- Los paquetes son un mecanismo que permiten dar nombres y restringir la visibilidad.
- Se pueden declarar clases dentro un paquete sin hacerlas accesibles fuera del paquete.
- Se pueden declarar miembros que sólo están accesibles a otros miembros del mismo paquete.

Partes de un archivo fuente

- Un archivo fuente de Java puede contener las cuatro partes internas siguientes:
  1. Una única sentencia de paquete (opcional).
  2. Las sentencias de importación deseadas (opcional).
  3. Una única declaración de clase pública (obligatorio).
  4. Las clases privadas de paquetes deseadas (opcional).

Definición

- Creación de un paquete

```
package paq1[.paq2[.paq3]];
```

- Ejemplo

```
package java.awt.Image;
```

- Asociado a una estructura de directorios.

Variable de entorno: CLASSPATH

- La localización específica que el compilador de Java considera como raíz de cualquier jerarquía de paquetes está controlada por **CLASSPATH**.
- Hasta ahora, hemos almacenando todas las clases en el mismo paquete por omisión y sin nombre.
  - Esto nos permitía compilar el código fuente y ejecutar el intérprete de Java sobre el resultado simplemente nombrando la clase en la línea de órdenes.
- La razón por la cual esto funcionaba es que, por defecto, el directorio de trabajo actual (".") está en la variable de entorno **CLASSPATH** definida por el intérprete de Java.

Definición

- Creación de un paquete

```
package paq1[.paq2[.paq3]];
```

- Ejemplo

```
package java.awt.Image;
```

- Asociado a una estructura de directorios.

Ejemplo

```
package MiPaq;

class Balance {
   String nombre;
   double bal;
   Balance(String n, double b) {
     nombre = n;
     bal = b;
   }
   void muestra() {
     if (bal<0)
       System.out.print("-->> ");
     System.out.println(nombre +
          ": $" + bal);
   }
}
```

```
class RecuentoBalance {
  public static void main(String args [ ]) {
   Balance actual [ ] = new Balance [ 3 ];
   actual [0] = new Balance("K.J.
   Fielding",123.23);
   actual [1] = new Balance("Will Tell",157.02);
   actual [2] = new Balance("Tom Jackson",-
   12.33);
   for(int i = 0; i<3; i++)
      actual [i].muestra();
  }
}
/*
E:>java  MiPaq.RecuentoBalance
K.J. Fielding: $123.23
Will Tell: $157.02
-->> Tom Jackson: $-12.33
*/
```

Protección de acceso

- Las clases y los paquetes son dos medios de encapsular y contener el espacio de nombres y el ámbito de las variables y métodos.
- Los paquetes actúan como recipientes de datos y código.
- La clase es la unidad de abstracción más pequeña de Java.
  - Un miembro **privado** de una clase está permitido sólo al resto de miembros de esa clase.
- Debido a la relación entre clases y paquetes, Java distingue cuatro categorías de visibilidad entre elementos de clase:
  - **Subclase del mismo paquete.**
  - **No subclases del mismo paquete.**
  - **Subclase en paquetes distintos.**
  - **Clases que no están ni en el mismo paquete ni en las subclases.**

Protección de acceso

	`private`	Sin modificador	`protected`	`public`
Misma clase	SI	SI	SI	SI
Subclase del mismo paquete	SI	SI	SI	SI
No subclase del mismo paquete	No	SI	SI	SI
Subclase de diferente paquete	No	No	SI	SI
No subclase de diferente paquete	No	No	No	SI

Importar paquetes

- No hay ninguna clase del núcleo de Java en el paquete sin nombre.
- Esto significa que todas las clases estándares están almacenadas en algún paquete con nombre.
- La sentencia **`import`** para que se puedan ver ciertas clases o paquetes enteros.
- Una vez importada, una clase puede ser referenciada directamente, utilizando sólo su nombre.
- Sintaxis:

```
import paquete1[.paquete2].(nombre_clase|*);
```

Importando paquetes

- No hay ninguna clase del núcleo de Java en.

```
import java.util.Date;
import java.io.*;
import java.lang.*;
```

Ejemplo I

```
package p1;
public class Proteccion {
   int n = 1;
   private int n_pri = 2;
   protected int n_pro = 3;
   public int n_pub = 4;

   public Proteccion() {
    System.out.println("constructor base ");
    System.out.println("n =  " + n);
    System.out.println("n_pri =  " + n_pri);
    System.out.println("n_pro =  " + n_pro);
    System.out.println("n_pub =  " + n_pub);
   }
}
class Derivada extends Proteccion {
   Derivada() {
    System.out.println("constructor de Derivada ");
    System.out.println("n =  " + n);

   //solo para su clase
   //System.out.println("n_pri =  " + n_pri);

    System.out.println("n_pro =  " + n_pro);
    System.out.println("n_pub =  " + n_pub);
   }
}
```

```
class MismoPaquete {
 MismoPaquete() {
    Proteccion p = new Proteccion();
    System.out.println("constructor de
     MismoPaquete");
    System.out.println("n =  " + p.n);

   //solo para su clase
   //System.out.println("n_pri =  " + p.n_pri);
    System.out.println("n_pro =  " + p.n_pro);
    System.out.println("n_pub =  " + p.n_pub);
   }
}
class Demo{
    public static void main(String args[]){
        Proteccion   ob1 = new Proteccion   ();
        Derivada     ob2 = new Derivada     ();
        MismoPaquete ob3 = new MismoPaquete ();
    }
}
```

Salida del Ejemplo I

```
E:\F_Telematica\Tema9>javac .\p1\Proteccion.java
E:\F_Telematica\Tema9>java p1.Demo
constructor base
n =  1
n_pri =  2
n_pro =  3
n_pub =  4
constructor base
n =  1
n_pri =  2
n_pro =  3
n_pub =  4
constructor de Derivada
n =  1
n_pro =  3
n_pub =  4
constructor base
n =  1
n_pri =  2
n_pro =  3
n_pub =  4
constructor de MismoPaquete
n =  1
n_pro =  3
n_pub =  4

E:\F_Telematica\Tema9>
```

Ejemplo II

```
package p2;
class Proteccion2 extends p1.Proteccion {
    Proteccion2() {
    System.out.println("constructor Proteccion2 ");

    //solo para su clase o paquete
    //System.out.println("n =  " + n);

    //solo para su clase
    //System.out.println("n_pri =  " + n_pri);

    System.out.println("n_pro =  " + n_pro);
    System.out.println("n_pub =  " + n_pub);
   }
}
class OtroPaquete {
   OtroPaquete() {
     p1.Proteccion p = new p1.Proteccion();
     System.out.println("constructor de
     OtroPaquete");

    //solo para su clase o paquete
    //System.out.println("n =  " + p.n);

    //solo para su clase
    //System.out.println("n_pri =  " + p.n_pri);

    //solo para su clase, subclase o paquete
    //System.out.println("n_pro =  " + p.n_pro);

    System.out.println("n_pub =  " + p.n_pub);
   }
}
```

```
class Demo{
   public static void main(String args[]) {
    Proteccion2 ob1 = new Proteccion2();
    OtroPaquete ob2 = new OtroPaquete();
  }
}
/*
E:\F_Telematica\Tema9>javac .\p2\Proteccion2.java

E:\F_Telematica\Tema9>java p2.Demo
constructor base
n =  1
n_pri =  2
n_pro =  3
n_pub =  4
constructor Proteccion2
n_pro =  3
n_pub =  4
constructor base
n =  1
n_pri =  2
n_pro =  3
n_pub =  4
constructor de OtroPaquete
n_pub =  4

E:\F_Telematica\Tema9>
*/
```

Modelado UML de un paquete

- Son las partes organizativas de los modelos UML.
- Hay un elemento de agrupación principal, los paquetes. Un paquete es un mecanismo de propósito general para organizar elementos en grupos.
- Los elementos estructurales, los elementos de comportamiento, e incluso otros elementos de agrupación pueden incluirse en un paquete.
- Al contrario de los componentes (que existen en tiempo de ejecución), un paquete es puramente conceptual (sólo existe en tiempo de desarrollo).

Representación UML de un paquete