Transparencias

de

Java

Tema 3:
Paquetes

TEMA 3: PAQUETES

¿PARA QUÉ SIRVEN LOS PAQUETES?

- Un paquete es un mecanismo Java de agrupación y organización de clases.
- Tres propósitos:
  - Mejorar la organización de las clases
  - Reducir los problemas de colisión de nombres entre clases
  - Controlar la visibilidad de las clases, atributos y métodos definidos en él
- Las propias clases del núcleo de Java (API) se encuentran organizadas en paquetes:
  - package java.applet
  - package java.awt
  - package java.beans
  - package java.io
  - package java.lang
  - package java.sql
  - package java.util

◆ CARACTERÍSTICAS DE LOS PAQUETES

- ✶ Paquete ➔ conjunto de clases relacionadas
- ✶ Un paquete crea un nuevo espacio de nombres
  - ❑ No puede haber dos clases con el mismo nombre en un mismo paquete
  - ❑ Sí puede haber clases con el mismo nombre en distintos paquetes
- ✶ Los paquetes pueden anidarse formando jerarquías
  - ❑ Muy similar a como ocurre con los directorios

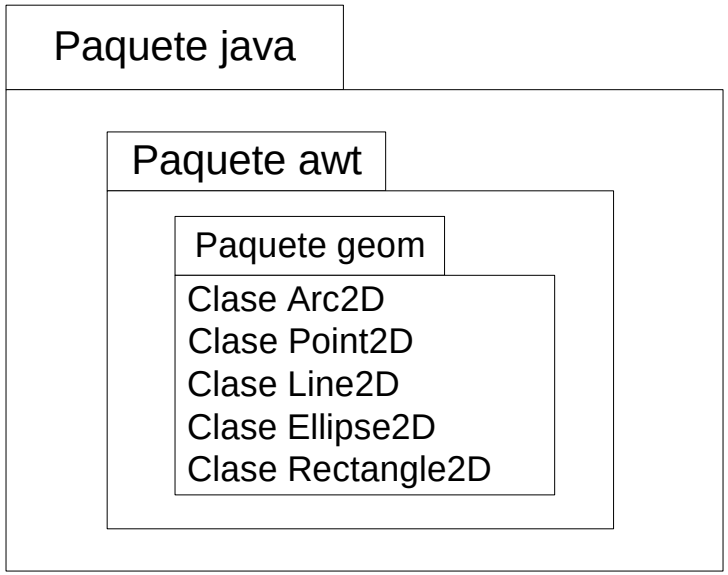

Figura 1: Algunas clases del paquete java.awt.geom

- Toda clase debe pertenecer a un paquete
  - En caso de no especificar explícitamente ningún paquete, existe un paquete por defecto (sin nombre) al que pasará a pertenecer la clase.
  - Todas las clases que hemos creado hasta ahora en la asignatura pertenecían a este paquete.

CÓMO CREAR UN PAQUETE

- Se usa la cláusula PACKAGE

```
package graphics;

class circle
{ ... }

class Rectangle
{ ... }
```

- Explicación:
  - Hemos creado un paquete llamado graphics
  - Todas las clases definidas a partir de la sentencia PACKAGE <NOMBRE_PAQUETE> pertenecerán a ese paquete
  - Las clases Circle y Rectangle pertenecen al paquete graphics

- Otra opción equivalente ➔ definirlas en ficheros diferentes
  - En un fichero Circle.java

```
package graphics;

class Circle
{ ... }
```

  - En un fichero Rectangle.java

```
package graphics;

class Rectangle
{ ... }
```

- Nombre Completo de una Clase
  - Incluye como prefijo el nombre del paquete al que pertenece:

```
graphics.Circle          graphics.Rectangle
```

  - Para ejecutarlas:

```
C:\java graphics.Circle
```

  - Para hacer referencia a ellas desde un programa:

```
graphics.Rectangle
```

- ✶ Correspondencia entre paquetes y directorios
  - ❑ Todas la clases de un paquete deberán estar en un directorio con el mismo nombre.
  - ❑ Si el nombre del paquete tiene varias palabras (paquetes anidados) cada una de ellas será un subdirectorio.
  - ❑ Ejemplos:

CLASES Y PAQUETES	DIRECTORIOS
Clases del paquete java.awt.geom	En el directorio ...\java\awt\geom\
Clase del paquete graphics	En el directorio …\graphics\
Clase graphics.Circle	En …\graphics\Circle.class
Clase graphics.Rectangle	En …\graphics\Rectangle.class

◆ CÓMO IMPORTAR UN PAQUETE

- ✶ Para acceder a las clases de otro paquete deberemos referirnos a ellas mediante su nombre completo.

```
package otroPaquete;

class MiClase
{
   ...
   graphics.Circle c = new graphics.Circle();
   graphics.Rectangle r = new graphics.Rectangle();
   java.util.Vector v = new java.util.Vector();
```

```
   ...
}
```

- Para evitar tener que indicar constantemente el paquete al que pertenece una clase, tenemos la palabra IMPORT

```
package otroPaquete;

import graphics.Circle;
import graphics.Rectangle;
import java.util.Vector;

class MiClase
{
   ...
   Circle c = new Circle();
   Rectangle r = new Rectangle();
   Vector v = new Vector();
   ...
}
```

- Palabra reservada IMPORT
  - Permite importar las clases de un paquete.
  - Debe estar al principio del fichero, antes de cualquier definición de clase y después de la sentencia package (si es que la hay).
  - También se pueden usar caracteres comodín (*) para importar todas las clases de un paquete.

```
package otroPaquete;

import graphics.*;
import java.util.*;
import java.awt.*;

class MiClase
{
   ...
   Circle c = new Circle();
   Rectangle r = new Rectangle();
   Vector v = new Vector();
   Windows w = new Windows();
   ...
}
```

- Las clases de paquete java.lang son importadas automáticamente por la JVM

VISIBILIDAD DE LAS CLASES

- Una clase puede ser declarada como:
  - De acceso PACKAGE (por defecto)
    - Será accesible solamente por las clases de su mismo paquete.
  - De acceso PUBLIC
    - Será accesible por cualquier otra clase siempre que ésta, si no está en su mismo paquete, la importe

- Ejemplo:

```
package contenedores;

public class Lista   //Acceso Public
{
   private Nodo raiz;
   public void add(Object dato){ ... }
   ...
}

class Nodo           //Acceso Package
{
   Object dato;
   Nodo siguiente;
}
```

- En cada fichero de código fuente sólo podrá haber una clase con acceso public

LA VARIABLE CLASSPATH

- Una aplicación Java local...
  - Se compone de un conjunto de clases que interactúan entre sí.
  - El código de estas clases puede encontrarse en cualquier lugar dentro del sistema de directorios de la máquina local.

- Ante esto, ¿Cómo localiza la JVM el código de las clases que intervienen en un programa?
  - Primero, la JVM añade a los nombres de todas las clases la extensión *class* para formar los nombres de los ficheros que contienen esas clases
  - Después, la JVM para localizar dichos ficheros usa la variable de entorno CLASSPATH
- La variable classpath contiene una serie de localizaciones donde la JVM debe buscar las clases Java compiladas
- Configuración de la variable classpath en Windows y DOS

```
set classpath = path1;path2;...
```

  - Los “path” deberían ser los directorios donde se encuentran los paquetes que contienen las clases a buscar.
  - Recetilla ➔ Siempre se debe indicar en el classpath el directorio anterior a dónde comienza la estructura de directorios correspondiente al paquete

◆ MECANISMO DE FUNCIONAMIENTO DEL CLASSPATH

- La JVM para localizar una clase en tiempo de ejecución:
  - Añade “.class” al nombre completo de la clase
  - Transforma los puntos en separadores de directorio “\”

- Añade la cadena resultante a cada una de las localizaciones del classpath.
- Si no se encuentra la clase en ninguna de ellas se produce un error de ejecución.

✶ Ejemplo:

- Tenemos la clase *ESIDE.ejemplos.ejemplo1*
- Tenemos el classpath definido como:

```
set classpath = c:\programas;c:\temp
```

- La JVM sabe que, según el nombre completo de la clase, debería encontrarla en un fichero que se encuentre en la dirección relativa *ESIDE\ejemplos\ejemplo1.class*
- Problema ➔ ¿A partir de qué directorio comienza a buscarla?
- Solución ➔ A partir de los directorios que están en el classpath
- Por tanto, lo buscará en:
  - c:\programas\ESIDE\ejemplos\ejemplo1.class
  - c:\temp\ESIDE\ejemplos\ejemplo1.class
- Si la clase no se encuentra en ninguno de esos dos sitios, se produciría un error de ejecución.

◆ EJEMPLOS DE CONFIGURACIÓN DEL CLASSPATH

- ✶ Tenemos la siguiente clase:

```
package universidad.eside;

public class Alumno
{
   ...
}
```

- ✶ En este momento ya sabemos que:
  - ❑ El fichero que contiene esta clase se llamará “Alumno.class”
  - ❑ Este fichero estará colocado dentro de la estructura de directorios “universidad\eside”
- ✶ Supongamos ahora que los directorios “universidad\eside” se encuentran dentro del directorio “c:\programacion”

```
C:\programacion\universidad\eside\Alumno.class
```

- ✶ ¿Cómo habría que configurar el classpath si quisiésemos usar en nuestro programa la clase Alumno?
- ✶ Solución:

```
set classpath = c:\programacion
```

◆ RESUMEN

- ✶ Concepto de paquete ➔ Mecanismo de agrupación de clases
- ✶ Cómo crear un paquete ➔ Cláusula PACKAGE
- ✶ Nombre completo de una clase ➔ paquete1.paquete2.Nombre_Clase
- ✶ Correspondencia entre paquetes y directorios
- ✶ Cómo importar las clases de un paquete ➔ Cláusula IMPORT
- ✶ Visibilidad de las clases ➔ PACKAGE y PUBLIC
- ✶ La variable CLASSPATH

◆ EJERCICIO PROPUESTO

- ✶ Apuntes Tema 3