Organización de las clases

Organización física: Ficheros

En Java, el código correspondiente a cualquier clase pública ha de estar definida en un fichero independiente con extensión `.java`.

El nombre del fichero ha de coincidir con el nombre de la clase.

En ocasiones, en un fichero se pueden incluir varias clases si sólo una de ellas es pública (esto es, las demás son únicamente clases auxiliares que utilizamos para implementar la funcionalidad correspondiente a la clase pública).

> *Ejemplo*
> Las clases que se utilizan para implementar manejadores de eventos en aplicaciones con interfaces gráficas de usuario.

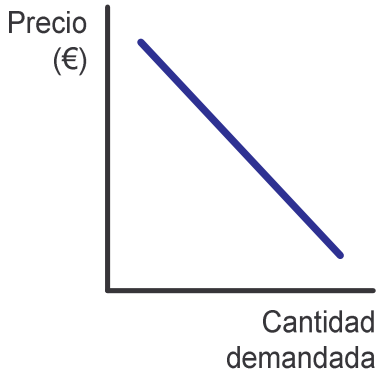
Una vez compilada, una clase, sea pública o no, da lugar a un fichero con extensión `.class` en el que se almacenan los bytecodes correspondientes al código de la clase.

Cuando ejecutemos una aplicación que utilice una clase particular, el fichero `.class` correspondiente a la clase debe ser accesible a partir del valor que tenga en ese momento la variable de entorno CLASSPATH:

> El fichero `.class` debe encontrarse en una de las carpetas/directorios incluidos en el CLASSPATH.
>
> Java también permite incluir ficheros comprimidos en el CLASSPATH, en formato ZIP, con extensión `.zip` o `.jar`, por lo que el fichero `.class` puede encontrarse dentro de uno de los ficheros de este tipo incluidos en el CLASSPATH.

Organización lógica: Paquetes

- Las clases en Java se agrupan en paquetes.
- Todas las clases compiladas en el mismo directorio (carpeta) se consideran pertenecientes a un mismo paquete.
- El paquete al que pertenece una clase se indica al comienzo del fichero en el que se define la clase con la palabra reservada `package`.
- El nombre del paquete ha de cumplir las mismas normas que cualquier otro identificador en Java.




```
package economics;

public class Demanda
{
  private double pendiente;
  private double precioMaximo;

  public Demanda (double precioMax, double pendiente)
  {
    this.pendiente = pendiente;
    this.precioMaximo = precioMaximo;
  }


  public double getPrecio (long cantidad)
  {
    return precioMaximo + cantidad*pendiente;
  }

  public long getCantidadDemandada (double precio)
  {
    return (long) ((precio-precioMaximo)/pendiente);
  }
}
```

- Cuando una clase pertenece a un paquete (p.ej. `economics`), el fichero `.java` ha de colocarse en un subdirectorio del directorio que aparezca en el CLASSPATH (p.ej. `personal/economics` si `personal` es lo que aparece en el CLASSPATH).

- Los paquetes se pueden organizar de forma jerárquica, de forma que `economics.markets` será un subpaquete del paquete `economics`

  Los ficheros `.java/.class` correspondientes deberán colocarse en el directorio `personal/economics/markets`

- Cuando usamos una clase que no está en el mismo paquete en el que nos encontramos, hemos de incluir una sentencia `import` al comienzo del fichero `.java` o utilizar el nombre completo de la clase a la que hacemos referencia en el código (esto es, `paquete.Clase`).

```
package economics.markets;

public class AnalisisEconomico
{
  private economics.Demanda  demanda;
  private economics.Costes   costes;
  private economics.Ingresos ingresos;
  …
}
```

o bien…

```
package economics.markets;

import economics.*;

public class AnalisisEconomico
{
  private Demanda  demanda;
  private Costes   costes;
  private Ingresos ingresos;
  …
}
```

Ejemplo

La biblioteca de clases estándar de Java
incluye cientos de clases organizadas en multitud de paquetes:

Paquete	Descripción
`java.lang`	Clases centrales de la plataforma Java (números, cadenas y objetos). No es necesario incluir la sentencia `import` cuando se usan clases de este paquete.
`java.awt`	Clases para crear interfaces gráficas y dibujar figuras e imágenes.
`java.applet`	Clases necesarias para crear applets.
`java.io`	Clases para realizar operaciones de entrada/salida (p.ej. uso de ficheros).
`java.util`	Utilidades varias: fechas, generadores de números aleatorios, vectores de tamaño dinámico, etcétera.
`java.net`	Para implementar aplicaciones distribuidas (que funcionen en redes de ordenadores).
`java.rmi`	Para crear aplicaciones distribuidas con mayor comodidad [*Remote Method Invocation*].
`java.sql`	Clases necesarias para acceder a bases de datos.
`javax.swing`	Para crear interfaces gráficas de usuario con componentes 100% escritos en Java.