CLASES +

A la hora de diseñar un conjunto de clases para modelar el conjunto de información cuyo tratamiento se desea automatizar, es importante establecer apropiadamente las posibles relaciones que deben existir entre unas clases y otras.

⇒ Especialización
⇒ Generalización

Se pueden distinguir diversos tipos de relaciones entre clases:

✓ **Clientela**: Una clase utiliza objetos de otra clase.

✓ **Composición**: Algún atributo de una clase es objeto de otra.

✓ **Anidamiento**: Cuando se definen clases en el interior de otra.

✓ **Herencia**: Cuando una clase comparte características con otra, añadiendo alguna función específica

¿Herencia o Composición?

Cuando una clase está formada por objetos de otras clases utilizamos **Composición**, en cambio, cuando una clase cumple todas las características de otra utilizamos **Herencia**.

COMPOSICIÓN

El constructor de la clase contenedora debe invocar a los constructores de los objetos contenidos.

```
Pool ps=new Pool();   ← constructor
ps.volar();           ← método a invocar
Object nombre;        ← objeto de otra clase
```

HERENCIA

La clase de la que se hereda suele ser llamada **clase padre** o **superclase** y la que hereda **clase hija** o **subclase**.

```
public class Alumno extends Curso{
```
hereda →

Object()

La clase **Object()** representa la superclase que se encuentran en la cúspide de la jerarquía de herencia en Java.

Toda clase Java tiene un método **toString()** y **finalize()** heredado de Object().

Clases Abstractas

En algunos casos puede resultar útil disponer de clases que nunca serán instanciadas, sino que proporcionan un modelo a seguir por sus clases derivadas dentro de una jerarquía de herencia. Son las clases abstractas.

Public abstract class Vo{

Una clase abstracta sólo puede usarse para crear nuevas clases derivadas, pero no podría hacer un new de una clase asstracta.

Puede contener métodos asstractos (sin definir) y métodos no asstractos (totalmente definidos).

⚠ Un método abstracto implica que la clase a la que pertenece es abstracta.

⚠ Un método abstracto no puede ser privado y no podría implementarse.

⚠ No pueden ser estáticos pues los métodos estáticos no pueden ser redefinidos.

final

Una clase declarada como final no puede ser heredada, no puede tener clases derivadas.

Un método también puede ser declarado como final, ese método no podrá ser redefinido.

@romii278

Interfaces

Una interfaz consiste en una lista de declaraciones de métodos sin implementar, que caracterizan un determinado comportamiento.

Public abstract class Vo implements Xi{
↳ abstracta
↳ interfaz

Una clase puede adoptar distintos modelos de comportamiento establecidos en diferentes interfaces, una clase puede implementar varias interfaces.

Public interface Xi{

Las clases que quieran realizar una determinada acción cada vez que se produzca cierto evento en el sistema deben implementar la interfaz ActionListener con el método actionPerformed.

Aunque no está permitida la herencia múltiple para las clases, sí lo está para interfaces.

Polimorfismo

El polimorfismo ofrece la posibilidad de que toda referencia a un objeto de una superclase pueda tomar la forma de una referencia a un objeto de una de sus subclases, y puede llevarse a cabo en superclases e interfaces.

⇒ Ligadura estática = soon
⇒ Ligadura dinámica = later