Herencia

Relación de herencia

- Se basa en la existencia de relaciones de generalización/especialización entre clases.
- Las clases se disponen en una jerarquía, donde una clase hereda los atributos y métodos de las clases superiores en la jerarquía.
- Una clase puede tener sus propios atributos y métodos adicionales a lo heredado.
- Una clase puede modificar los atributos y métodos heredados.
- Las clases por encima en la jerarquía a una clase dada, se denominan superclases.
- Las clases por debajo en la jerarquía a una clase dada, se denominan subclases.
- Una clase puede ser superclase y subclase al mismo tiempo.
- Tipos de herencia:
    - Simple.
    - Múltiple (no soportada en Java)

La clase Object

En Java todas las clases heredan de otra clase:

- Si lo especificamos en el código con la keyword extends, nuestra clase heredará de la clase especificada.
- Si no lo especificamos en el código, el compilador hace que nuestra clase herede de la clase Object (raíz de la jerarquía de clases en Java).

- Esto significa que nuestras clases siempre van a contar con los atributos y métodos de la clase Object.

Castings

- El casting es una forma de realizar conversiones de tipos.
- Hay dos clases de casting:
    - UpCasting: conversión de un tipo en otro superior en la jerarquía de clases. No hace falta especificarlo.
    - DownCasting: conversión de un tipo en otro inferior en la jerarquía de clases.

- Se especifica precediendo al objeto a convertir con el nuevo tipo entre paréntesis.

Sobrescribir un método

- Sobrescribir un método significa que una subclase reimplementa un método heredado.
- Para sobrescribir un método hay que respetar totalmente la declaración del método:
    - El nombre ha de ser el mismo.
    - Los parámetros y tipo de retorno han de ser los mismos.
    - El modificador de acceso no puede ser más restrictivo.

- Al ejecutar un método, se busca su implementación de abajo hacia arriba en la jerarquía de clases.

Sobrescribir vs. Sobrecargar

- Sobrecargar un método es un concepto distinto a sobrescribir un método.
- La sobrecarga de un método significa tener varias implementaciones del mismo método con parámetros distintos:
    - El nombre ha de ser el mismo.
    - El tipo de retorno ha de ser el mismo.
    - Los parámetros tienen que ser distintos.
    - El modificador de acceso puede ser distinto.

- Habrá que tener muy en cuenta los parámetros que se envían y las conversiones por defecto para saber qué método se ejecuta.

Sobrecarga de métodos

- Se permite la sobrecarga de métodos cambiando también el tipo de retorno, pero siempre que:
- El método que se está sobrecargando sea de una clase padre (de la que heredamos directa o indirectamente).
- El nuevo tipo de retorno sea hijo del tipo de retorno del método original (es decir, que herede de él directa o indirectamente).
- Por tanto, no es válido para tipos primitivos.

El uso de la Herencia

- Debemos usar herencia cuando hay una clase de un tipo mas específico que una superclase.
  Es decir, se trata de una especialización. Lobo es mas específico que Canino. Luego tiene sentido que Lobo

herede de Canino.

- Debemos usar herencia cuando tengamos un comportamiento que se puede reutilizar entre varias otras clases del mismo tipo genérico. Las clases Cuadrado, Circulo y Triangulo tiene que calcular su área y perímetro luego tiene sentido poner esa funcionalidad en una clase genérica como Figura.
- No debemos usar herencia solo por el hecho de reutilizar código. Nunca debemos romper las dos primeras reglas. Podemos tener el comportamiento cerrar en Puerta. Pero aunque necesitemos ese mismo comportamiento en Coche no vamos a hacer que Coche herede de Puerta. En todo caso, coche tendrá un atributo del tipo Puerta.
- No debemos usar herencia cuando no se cumpla la regla: Es-un (Is-a). Refresco es una Bebida. La herencia puede tener sentido. Bebida es un Refresco. ¿? No encaja luego la herencia no tiene sentido.

super y this

- super y this son dos keywords de Java.
- super es una referencia al objeto actual pero apuntando al padre.
- super se utiliza para acceder desde un objeto a atributos y métodos (incluyendo constructores) del padre.
- Cuando el atributo o método al que accedemos no ha sido sobrescrito en la subclase, el uso de super es redundante.
- Los constructores de las subclases incluyen una llamada a super() si no existe un super o un this.
- this es una referencia al objeto actual. this se utiliza para acceder desde un objeto a atributos y métodos (incluyendo constructores) del propio objeto.
- Existen dos ocasiones en las que su uso no es

redundante:

- Acceso a un constructor desde otro constructor.
- Acceso a un atributo desde un método donde hay definida una variable local con el mismo nombre que el atributo.