# RESUMEN TEORÍA PROGRAMACIÓN II

@Luuuoss

## HERENCIA Y POLIMORFISMO

Una clase hija es clase hija si y sólo si

1- Todos los objetos de la clase Hija son también objetos de la clase Padre.

2- El 100% de la definición del Padre (métodos)(atributos), debe ser, como mínimo parte de la definición de la clase hija.

- La herencia permite evitar duplicidad de código.
- Sólo puede haber 1 padre y todas las clases derivan de Object.
- Una clase hija puede acceder a métodos public, protected, y friendly si estuvieran en el mismo paquete.
- Permite la sobreescritura o especialización de métodos del Padre.

- UpCasting y DownCasting

Tenemos un objeto de una clase hija llamada "hijo" y un objeto de la clase padre llamado "padre".

- Es posible hacer padre = hijo, debido a la regla 1

Sin embargo, si se usa un método que está definido en la clase hija y no en la clase padre, se produce error de compilación.

- No se puede hacer hijo = padre directamente. Hay que hacer downcasting, osea: hijo = (ClaseHija) padre;

Si "padre" no contiene una instancia de ClaseHija en el momento del cast, se produce.

java.lang.ClassCastException;

- Sobreescritura de métodos

Las clases hijas pueden completar o cambiar el comportamiento de un método de la Clase Padre.

El método redefinido debe:

- Dar un resultado igual o mejor (POST igual o más fuerte)
- Exigir precondiciones iguales o más débiles.

El redefinido puede usar super para acceder al método padre o a cualquier otro del padre, o suyo.

@Lumoss

- **Sobrecarga de métodos**

  Métodos con el mismo nombre, y devuelven lo mismo.
  Puede variar el número, tipo y orden de los argumentos.

- **Sobreescritura**

  Métodos con mismo nombre pero distinta implementación en la clase Padre que en la clase Hija.

- **Polimorfismo**

  El polimorfismo se debe a la sobreescritura. Una misma llamada ejecuta sentencias distintas en función del tipo al que pertenezca el objeto.

  Esta "decisión" se produce gracias al enlace dinámico, el cual decide en tiempo de ejecución (cuando se activa el programa)

  La sobrecarga de métodos es distinta, utiliza enlace estático por lo que decide en tiempo de compilación.

- **Métodos y clases abstractas**

  Si tenemos un método, p.ej. área de un polígono y clases hijas cuadrado, triángulo
  La implementación del método es distinta en cada clase
  Para hacer un método abstracto:

  ```
  abstract double area();
  ```
  (nombre: area; tipo: double)

  Y en cada clase hija se hace un método con la misma cabecera (sin el abstract) distinto cuerpo.

- Si una clase contiene al menos 1 método abstracto, es una clase abstracta.
- Una clase abstracta puede ser clase padre pero no se pueden crear objetos de una clase abstracta.