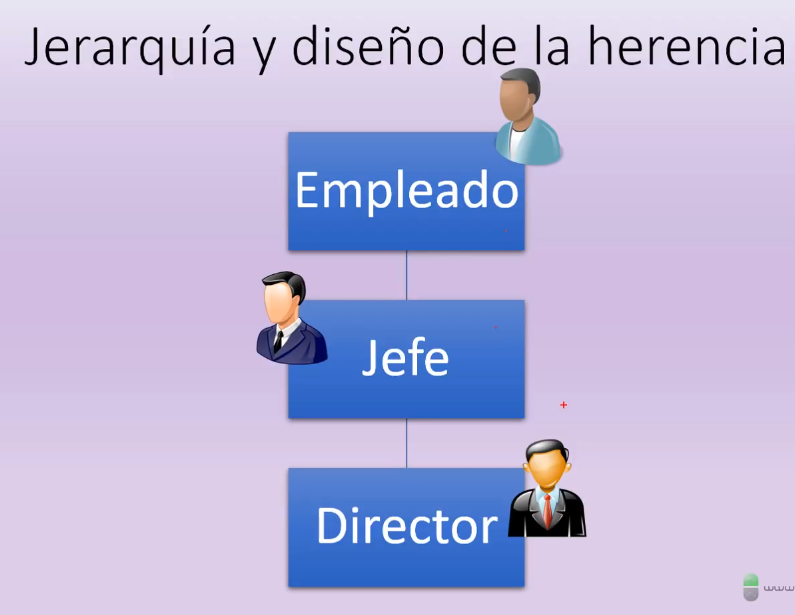
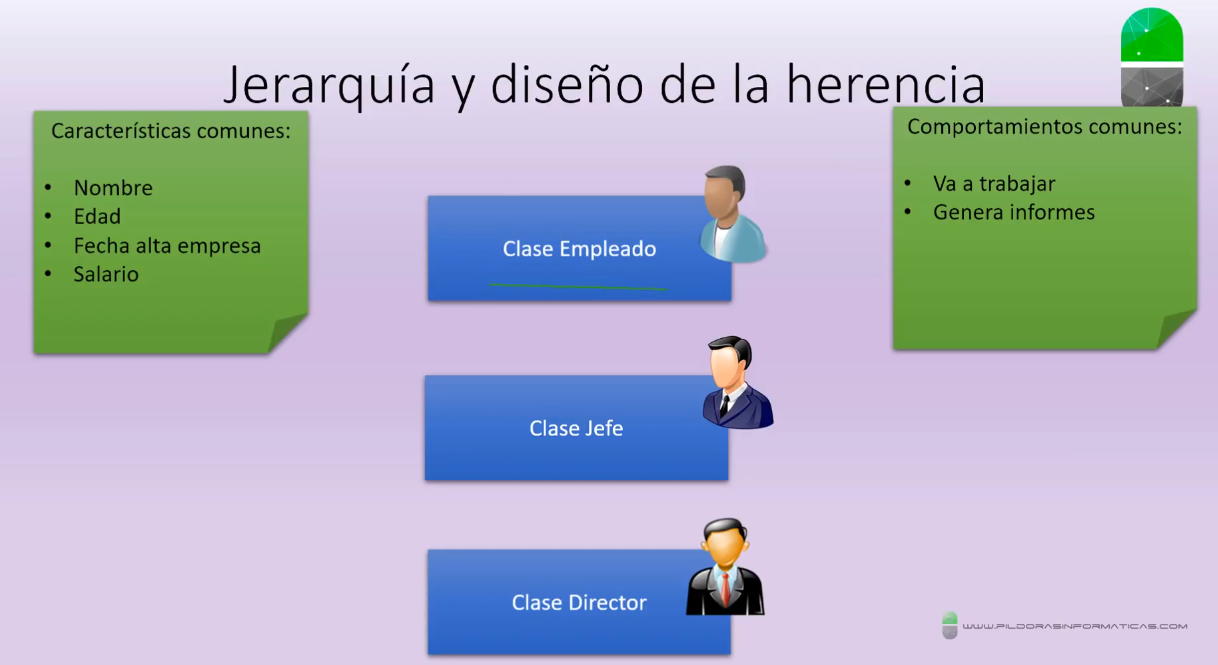
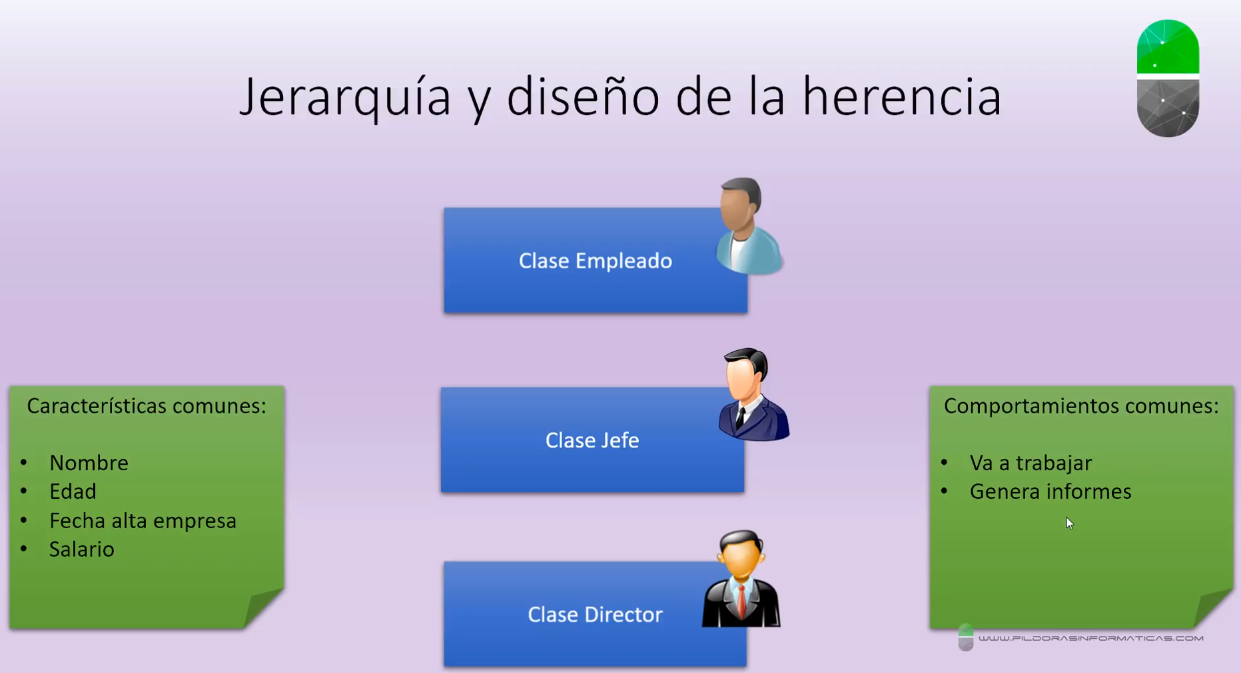
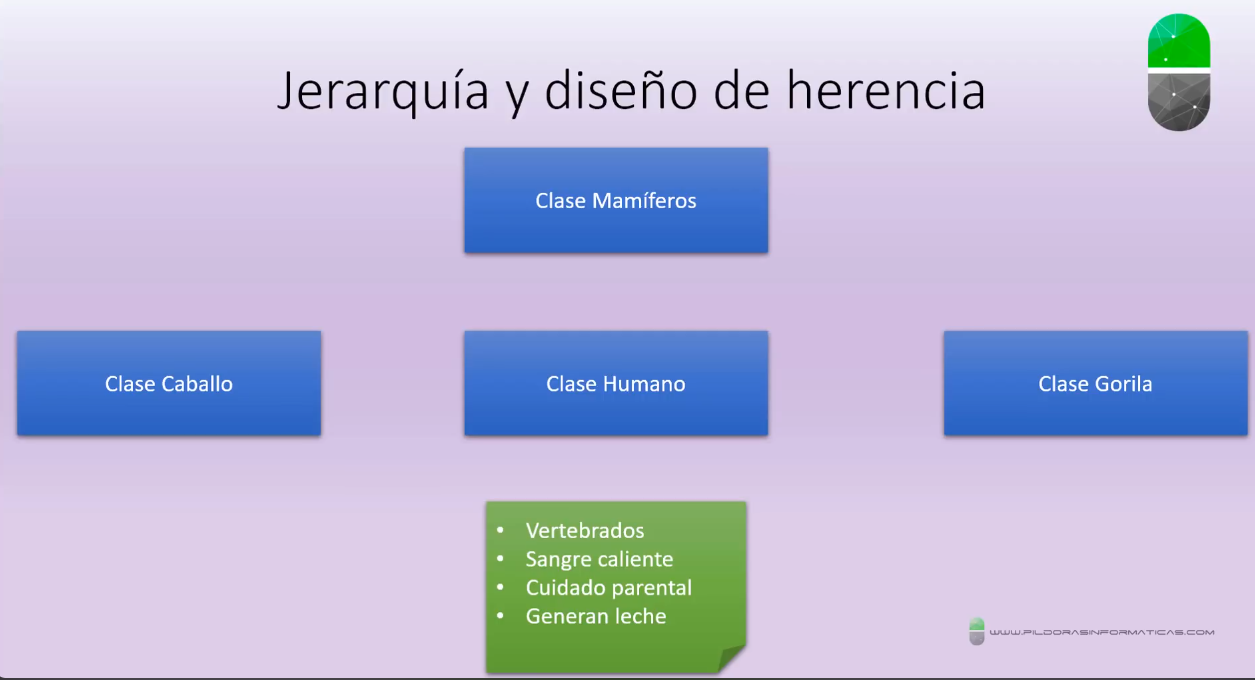


Otra forma de pensarlo:

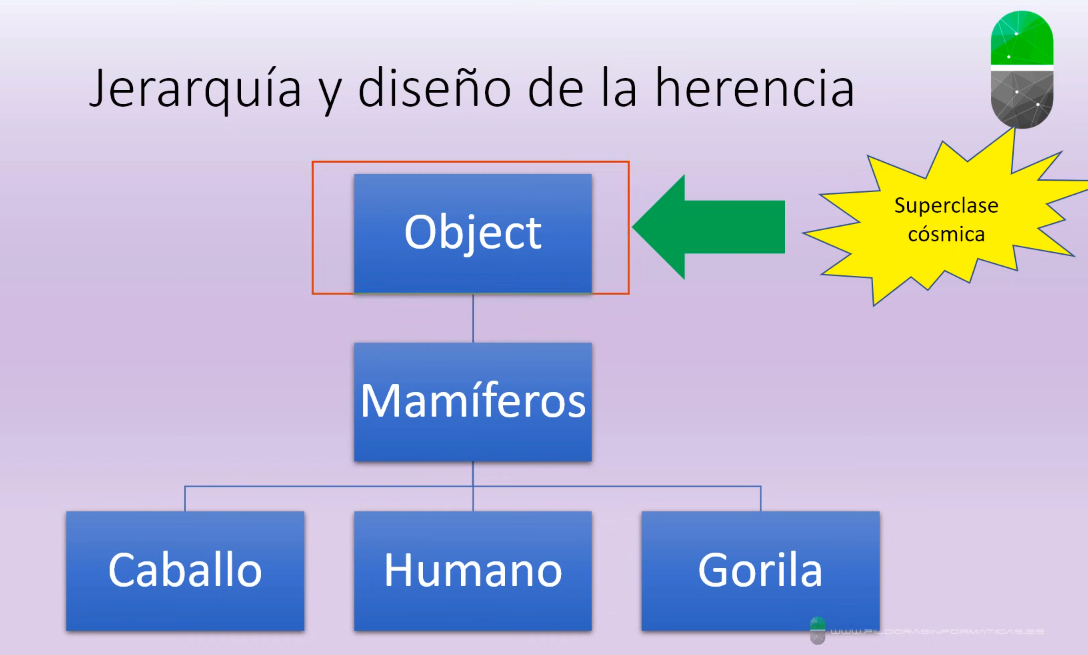
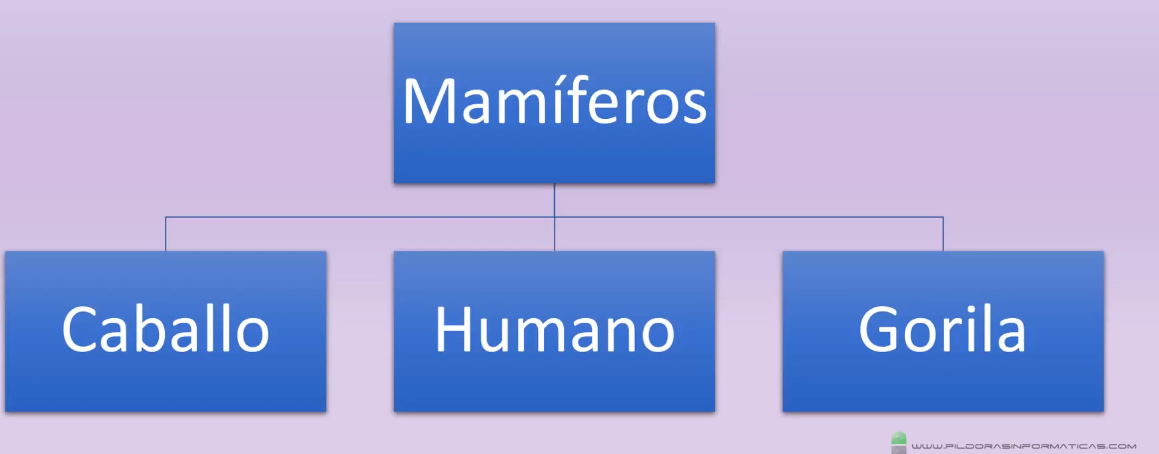




Otro ejemplo:



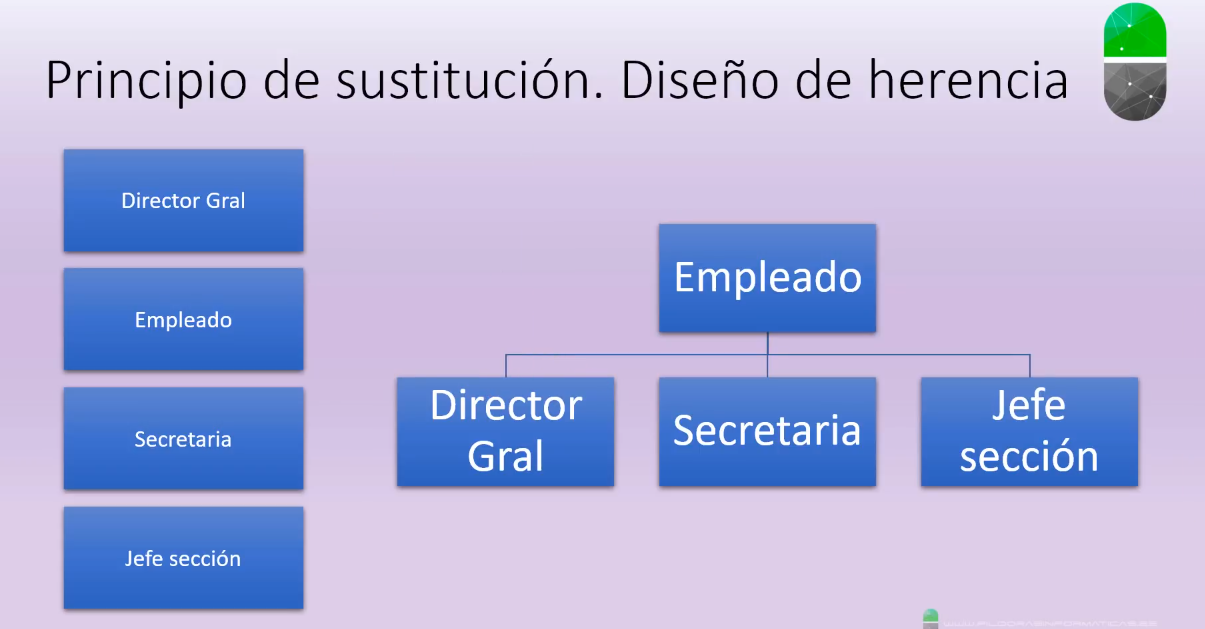
Quedaría así:



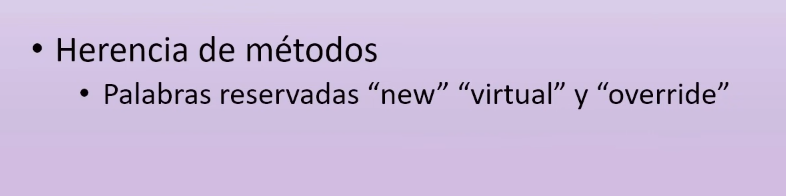


Al hacer nosotros el constructor de la clase padre, tendremos que programar manualmente la instrucción :base(), que sirve para heredar el constructor de la super clase (nosotros no lo vemos)





A *es siempre un* B 🡪 rta == si => A hereda de B



virtual:

* Permite que un método de la clase base pueda ser sobrescrito en las clases derivadas.
* Se usa en la clase base para indicar que el método es "polimórfico".

override:

* Se usa en una clase derivada para sobrescribir un método marcado como virtual en la clase base.
* El método sobrescrito debe tener el mismo nombre, tipo de retorno y parámetros.

new:

* Se usa en una clase derivada para ocultar un método de la clase base con el mismo nombre.
* El método de la clase base no se sobrescribe, simplemente se oculta en la clase derivada.

**Cuándo usar cada uno:**

* **virtual y override:**
  + Usar cuando quieres habilitar el polimorfismo y permitir que una clase derivada modifique el comportamiento de un método base.
* **new:**
  + Usar cuando quieres redefinir un método en una clase derivada, pero no necesitas reemplazar el comportamiento del método base en instancias referenciadas como clase base.

// Polimorfism - es la capacidad que tienen los objetos en programacion de comportarse de diferentes formas o a tener diferente forma dependiendo del contexto

// Poli... : Muchas

// ...morfimo : formas

**Modificador de acceso Protected**

En C#, la palabra clave **protected** es un modificador de acceso que permite que un miembro (método, propiedad, campo, etc.) sea **accesible solo desde:**

1. La **clase en la que se define**.
2. Las **clases derivadas** de esa clase base.

En C#, la palabra clave **protected** es un modificador de acceso que permite que un miembro (método, propiedad, campo, etc.) sea **accesible solo desde:**

1. La **clase en la que se define**.
2. Las **clases derivadas** de esa clase base.

**Características principales de protected:**

1. **No es accesible desde fuera de la clase o sus derivadas.**
   * No puedes acceder a un miembro protected usando un objeto de la clase.
2. **Es útil para herencia:**
   * Permite que las clases derivadas accedan a miembros que no deben ser públicos.
3. **Privado, pero con acceso adicional a derivadas:**
   * Es similar a private, pero permite a las clases hijas usar esos miembros.