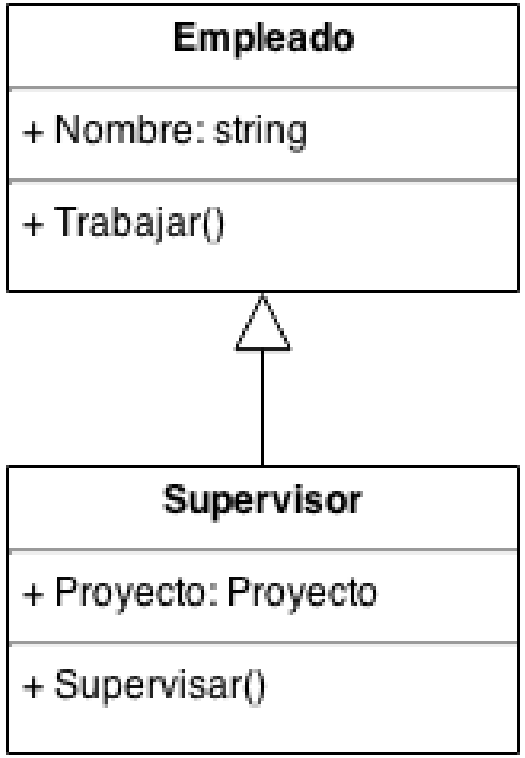
Herencia

Herencia

- La herencia permite a partir de una clase *padre* definir una subclase *hijo*
- La clase hija va a tener los atributos y métodos del padre además de definir los suyos propios
- La herencia potencia la reutilización de código, genera código y robusto y reduce el coste de mantenimiento



Herencia en Java

- La herencia en Java es simple. Una clase hijo puede tener sólo un padre
- Las propiedades y métodos privados no se heredan

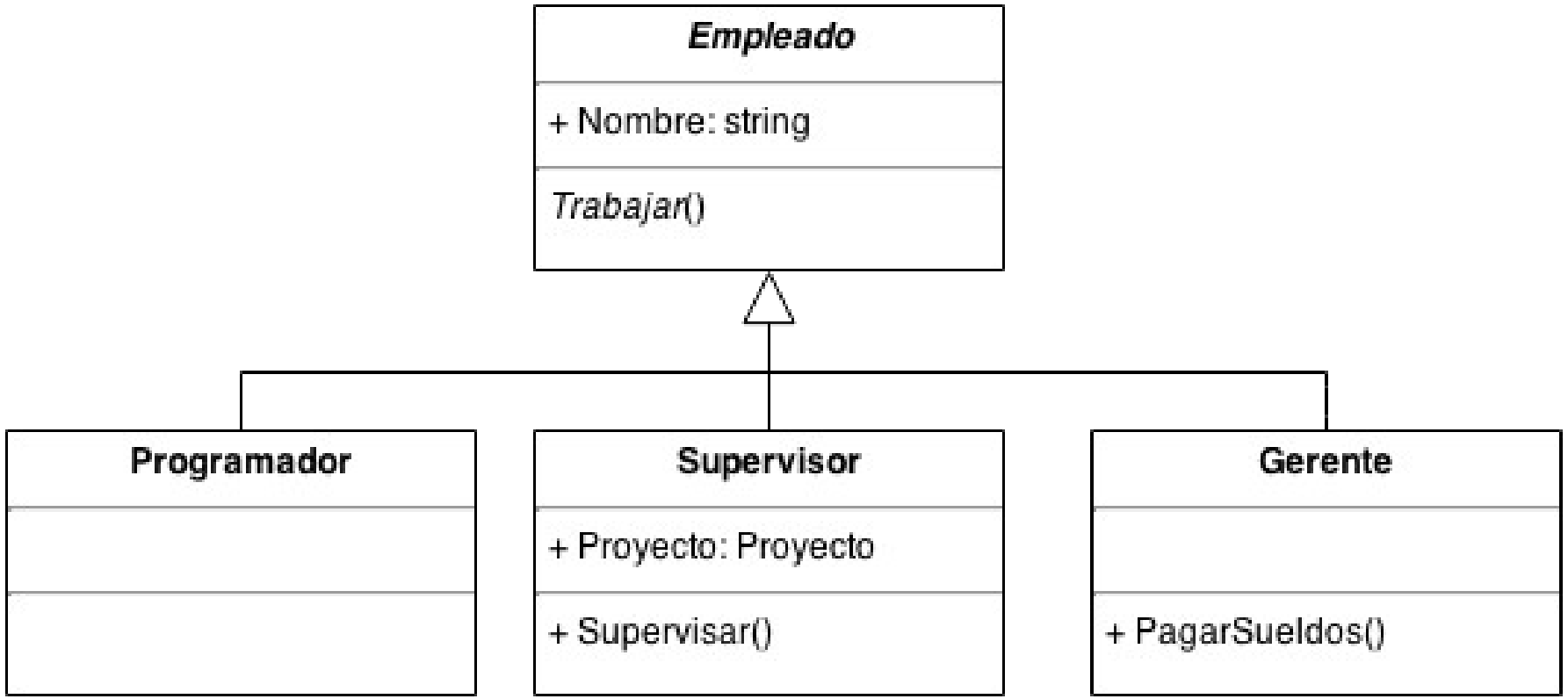
CAPTTON
IT CONSULTING

Herencia - Sintaxis

```
[modificadores] class NombrePadre{
  public NombrePadre(){
  }
}
[modificadores] class NombreClase extends NombrePadre{
  public NombreClase(){
     [super();]
  }
}
```

Modificador *Abstract*

- El modificador *Abstract* se utiliza para indicar que una clase, atributo, constructor o método posee una implementación incompleta
- Cuando se aplica a una clase, indicamos que dicha clase sólo se va a poder utilizar como base o padre para otras clases
- Los miembros que están marcados como abstractos o que se incluyen en una clase abstracta, **deben** ser implementados por las clases hijas de la clase padre abstracta


Empleado
+ Nombre: string
Trabajar()
Programador
Supervisor
+ Proyecto: Proyecto
+ Supervisar()
Gerente
+ PagarSueldos()

Modificador *Final*

- El modificador de clase *Final* se utiliza cuando queremos prohibir que otras clases hereden de ella
  - Es conceptualmente opuesto a *Abstract*
    - *Final* EVITA la herencia de una clase
    - *Abstract* REQUIERE la herencia de esa clase

- El modificador de métodos *Final* se utiliza cuando queremos que el método no pueda ser sobrescrito es clases derivadas.
- El modificador de campos *Final* se utiliza cuando queremos marcar que el campo solo puede ser inicializado una sola vez

CAPTTON
IT CONSULTING