LIGADURA DINÁMICA

Cuando se llama a un método sobreescrito mediante una referencia a una superclase, java determina qué versión del método ejecutar basándose en el tipo del objeto al que se referencia.

El funcionamiento descrito en el enunciado anterior recibe el nombre de *"despacho dinámico de métodos",* "ligadura dinámica", …. Y otros nombres. Esta técnica consiste en que java puede decidir que versión de método sobreescrito ejecutar en tiempo de ejecución, de ahí lo de "dinámico". Poder tomar esta decisión en tiempo de ejecución es muy importante porque es uno de los mecanismos con los que java soporta el *polimorfismo.*

La ligadura dinámica se consigue al utilizar conjuntamente dos mecanismos que ya conocemos:

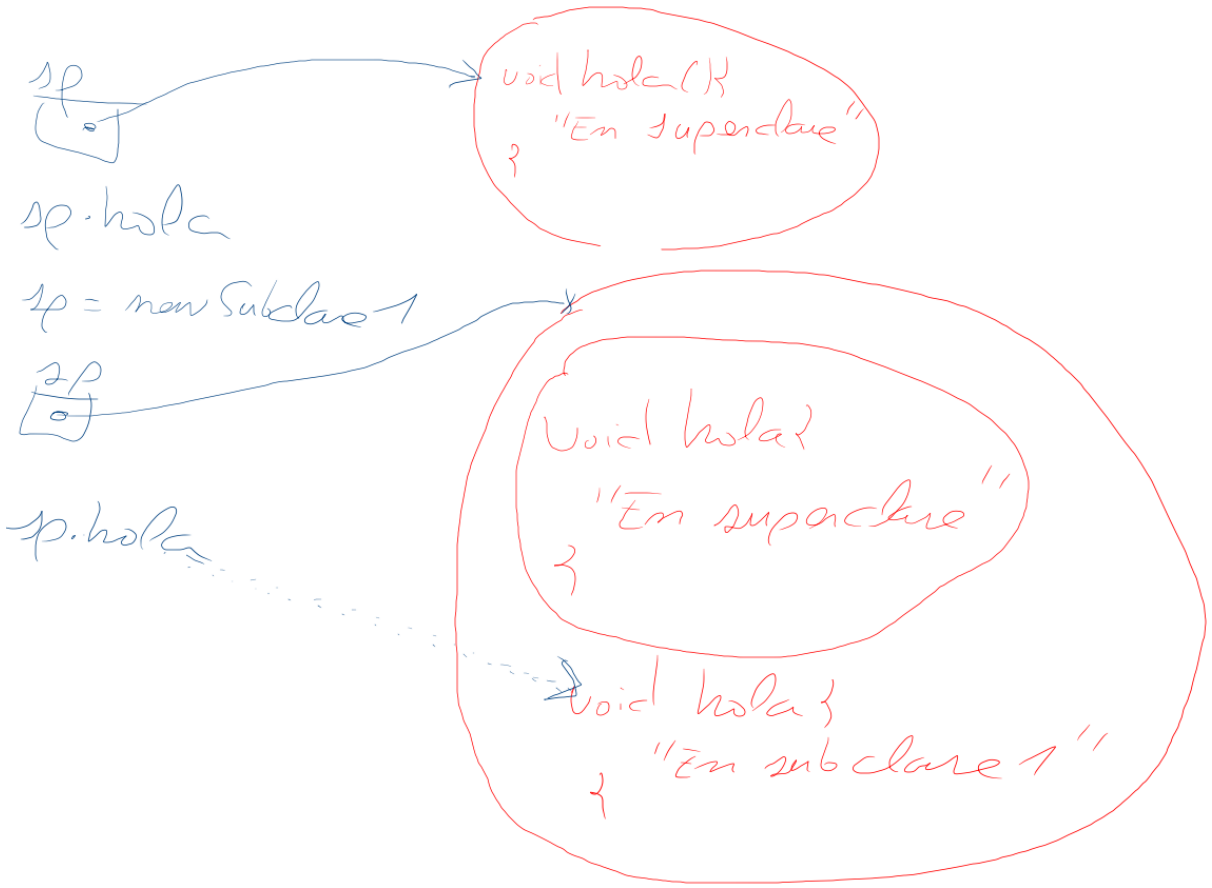
- Asignar a una referencia de superclase, un objeto de subclase
- Sobreescribir un método de la superclase en todas sus subclases.

Ejemplo:

```
class SuperClase{
    void hola(){
        System.out.println("hola() en superclase");
    }
}
class SubClase1 extends SuperClase{
    void hola(){
        System.out.println("hola() en subclase1");
    }
}
class SubClase2 extends SuperClase{
    void hola(){
        System.out.println("hola() en subclase2");
    }
}
class App{
    public static void main(String[] args) {
        SubClase1 sb1= new SubClase1();
        SubClase2 sb2= new SubClase2();
        SuperClase sp= new SuperClase();

        sp.hola();
        sp=sb1; //ahora la referencia de tipo SuperClase referencia a un objeto de tipo SubClase1
        sp.hola();
        sp=sb2;// y ahora a un objeto de tipo SubClase2
        sp.hola();
    }
}
```

Tienes que tener claro que sp puede acceder al método hola() de las subclases porque está sobreescrito. Un método no sobrescrito de una subclase es inaccesible desde una referencia superclase.



Ejemplo: Si anulamos (comentamos) el código de hola() en superclase con lo cual ya no hay sobreescritura y observa los errores que aparecen en el código.

```
class SuperClase{
    /*void hola(){  //AL COMENTAR ESTO NO HAY SOBRESCRITURA EN SUBLCASES
        System.out.println("hola() en superclase");
    }*/
}
class SubClase1 extends SuperClase{
    void hola(){
        System.out.println("hola() en subclase1");
    }
}
class SubClase2 extends SuperClase{
    void hola(){
        System.out.println("hola() en subclase2");
    }
}
class App{
    public static void main(String[] args) {
        SubClase1 sb1= new SubClase1();
        SubClase2 sb2= new SubClase2();
        SuperClase sp= new SuperClase();
```

```
        sp.hola();
        sp=sb1; //ahora la referencia de tipo SuperClase referencia a un objeto de tipo SubClase1
        sp.hola();
        sp=sb2;// y ahora a un objeto de tipo SubClase2
        sp.hola();
    }
}
```

Ejercicio U5_B6_E1: El sueldo base de un empleado es 1000. El sueldo de un empleado técnico son 1000 adicionales sobre la base y el de un jefe 2000 adicionales sobre la base. Completa la jerarquía para que tenga sentido el siguiente main().

```
class Empleado {
    public static int base = 1000;
    int salario() {
        return base;
    }
}
//añadir código para que funcione el main

class App {
    public static void main(String[] args) {
        Empleado empleado;
        empleado = new Tecnico();
        System.out.println(empleado+" tiene sueldo "+empleado.salario());
        empleado = new Jefe();
        System.out.println(empleado+" tiene sueldo "+empleado.salario());
    }
}
run:
Tecnico@1b2c6ec2 tiene sueldo 2000
Jefe@70177ecd tiene sueldo 3000
```

Ejercicio U5_B6_E2: Añadiendo el atributo nombre a Empleado y sobrescribiendo toString() en Empleado y con el siguiente main

```
class App {
    public static void main(String[] args) {
        Empleado empleado;
        empleado = new Tecnico("Técnico X");
        System.out.println(empleado);
        empleado = new Jefe("Gran jefe");
        System.out.println(empleado);
    }
}
```

Consigue esta salida

```
Técnico X tiene sueldo 2000
Gran jefe tiene sueldo 3000
```

Ejercicio U5_B6_E3: Ahora sobrescribiendo toString() en las subclases conseguir la siguiente salida con el mismo main anterior.

```
El empleado técnico llamado "Técnico X" tiene sueldo 2000
El empleado jefe llamado "Gran jefe" tiene sueldo 3000
```

Ejercicio U5_B6B_E4: Supongamos que el método salario() en la superclase Empleado no tiene sentido ya que jamás se ejecuta su código. Escribe de nuevo la clase Empleado con un salario() abstracto y comprueba que los ejemplos anteriores funcionen igual con la nueva clase empleado.