```java
public class SuperClase {
    public String nombre = "SuperClase"; 01
    public SuperClase(){…} 02
    public SuperClase(int i){…} 03
    public void Saludador(){return nombre}04
    public int Suma(){return 1 + 1;} 05
    protected void Secreto(){…} 06
}
```

```java
public class SubClase extends SuperClase {
    public String nombre = "SubClase"; 07
    public SubClase(){…} 08
    public SubClase(int i){…} 09
    public void Saludador(){return nombre} 10
    public int Suma(){return 2 + 2;}11
    public void Secreto(){super.Secreto()…} 12
    public int SuperSuma(){…}13
    public void SuperSaludador(){…}14
}
```

```java
public class SubSubClase extends SubClase{
    public SubSubClase() {…} 15
    public SubSubClase(int x) {…}  16
}
```

```java
SuperClase sup1 = new SuperClase(); // -> Llama a sus constructores 02
SuperClase sup2 = new SuperClase(0); // -> 03

SubClase sub1 = new SubClase(); // -> Notar que al no explicitar, llama al constructor del padre por omisión (sin argumentos)02 08
SubClase sub2 = new SubClase(0); // -> 02 09

SuperClase sub3 = new SubClase(); // -> Idem caso anterior. Pcipio de sustitución no altera instanciación 02 08
SuperClase sub4 = new SubClase(0); // -> 02 09

SubSubClase ssc1 = new SubSubClase(); // -> Idem caso anterior, se considera al 'padre del padre'  02 08 15
SubSubClase ssc2 = new SubSubClase(0); // -> 02 08 16

System.out.println("-> " + sup1.Suma()); // -> 05
System.out.println("~> " + sub1.Suma()); // -> Aplica la redefinición de método 11
System.out.println("~> " + sub3.Suma()); // -> Referencia no afecta redefinición de método 11

sup1.Saludador(); // -> 04
sub1.Saludador(); // -> Aplica redefinición 10
sub3.Saludador(); // -> Referencia no afecta redefinición de método, ni campos considerados (nombre = "SubClase") 10
```

```java
public class SuperClase {
    public String nombre = "SuperClase"; 01
    public SuperClase(){…} 02
    public SuperClase(int i){…} 03
    public void Saludador(){return nombre}04
    public int Suma(){return 1 + 1;} 05
    protected void Secreto(){…} 06
}
```

```java
public class SubClase extends SuperClase {
    public String nombre = "SubClase"; 07
    public SubClase(){…} 08
    public SubClase(int i){…} 09
    public void Saludador(){return nombre} 10
    public int Suma(){return 2 + 2;}11
    public void Secreto(){super.Secreto()…} 12
    public int SuperSuma(){…}13
    public void SuperSaludador(){…}14
}
```

```java
public class SubSubClase extends SubClase{
    public SubSubClase() {…} 15
    public SubSubClase(int x) {…}  16
}
```

```java
System.out.println(sup1.nombre);// -> considerará SuperClase 01
System.out.println(sub1.nombre);// -> considerará SubClase   07
System.out.println(sub3.nombre);// -> Impacto de la referencia, considerará SuperClase  .Campos se oculta, no se redefinen.01

System.out.println(sup1.SuperSuma()); // -> No esta definida en la clase SuperClase

System.out.println(sub1.SuperSuma()); // -> Si se puede, esta definida en la clase SubClase 13

System.out.println(sub3.SuperSuma()); // -> Impacto de la referencia, existe en la instancia no lo esta en la referencia (OJO pcipio de sustitución)

sup1.SuperSaludador(); // -> No esta definida en la clase SuperClase

sub1.SuperSaludador(); // -> Si se puede, esta definida en la clase SubClase 14

sub3.SuperSaludador(); // -> Impacto de la referencia, existe en la instancia no lo esta en la referencia (OJO pcipio de sustitución)


sup1.Secreto(); // -> No se puede fuera del paquete, dado que es protected

sub1.Secreto(); // -> Se puede dado que la subclase da acceso public 12

sub3.Secreto(); // -> Impacto de la referencia, se considera que no tiene acceso (protected). OJO picipio de sustitución.
```