

Práctica 11: Quita los comentarios del método emitirSonido() ¿ da algún error el IDE ? Toma captura de pantalla. ¿Se puede combinar private y abstract? Cambia private a public ¿ el IDE da algún mensaje? Pon el cursor en la línea que declaras la clase Pajaro y pulsa: ALT + INTRO y selecciona la opción “implement all abstract methods” Haz lo propio con la clase Perro pero esta vez utiliza ALT + INSERT (toma captura de pantalla de las ventanas que salgan) . Quita el modificador abstract de donde se define la clase Animal pero deja ese modificador en el método emitirSonido(). Toma captura de pantalla del mensaje contextual del IDE. Pulsa ALT + INTRO toma captura de pantalla de la ventana que muestra y sigue su recomendación de hacer la clase abstracta

```
abstract class Animal {  
    protected String nombre;  
    protected int edad;  
    protected int peso;  
  
    public Animal() {  
        System.out.println("jajaja");  
    }  
    private abstract String emitirSonido();  
}  
  
class Perro extends Animal {
```

illegal combination of modifiers: abstract and private

(Alt-Enter shows hints)

Si intentamos crear un método abstracto y a la vez privado, el IDE no indica que es una combinación no válida. Esto tiene sentido puesto que los métodos abstractos están diseñados para informar que ese método no lo especifica la clase, sino las clases hijas, y no permitir la edición de los mismos no tiene sentido.

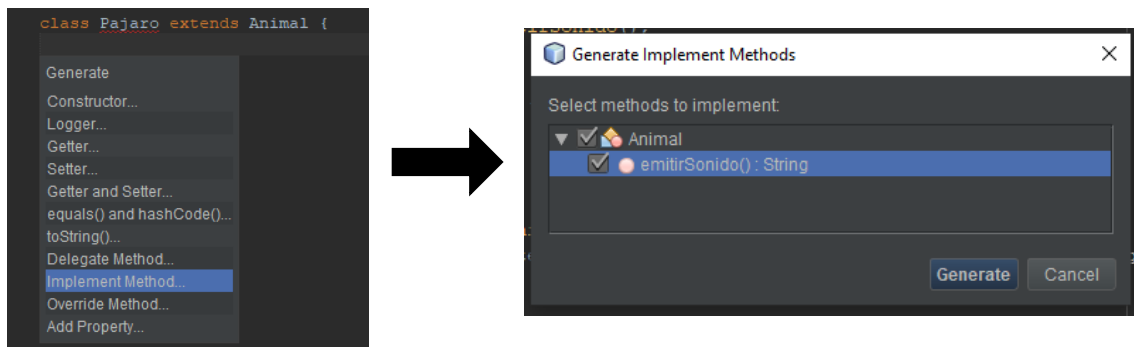
```
class Perro extends Animal {  
    double aia;  
}
```

- Implement all abstract methods
- Make class Perro abstract
- Create Test Class [JUnit in Test Packages]
- Create Test Class [Selenium in Test Packages]
- Create Test Class [TestNG in Test Packages]
- Create Subclass

Una vez cambiado el método a public, ambas clases hijas se nos quejaron de que no tienen implementados los métodos abstractos. Podemos implementarlos con ayuda del IDE con la combinación de botones ALT+ENTER.

```
class Perro extends Animal {  
    int dientes;  
  
    @Override  
    public String emitirSonido() {  
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  
    }  
}
```

También se puede implementar el método abstracto con la combinación de botones ALT+INSERT, en el apartado Implement Method... .



```
class Pajaro extends Animal {  
  
    @Override  
    public String emitirSonido() {  
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.  
    }  
  
    double ala;  
}
```

Los métodos abstractos solo se pueden generar en una clase abstracta, o en clases hijas de abstractas, si volvemos a la clase Animal no abstracta veremos que el IDE nos informa de esto.

```
Animal is not abstract and does not override abstract method emitirSonido() in Animal  
-----  
(Alt-Enter shows hints)  
  
class Animal {  
  
    protected String nombre;  
    protected int edad;  
    protected int peso;  
  
    public Animal() {  
        System.out.println("jajaja! soy un animal!!!!");  
    }  
    public abstract String emitirSonido();  
}
```

La clase Animal no es abstracta y no sobrescribe el método abstracto. La solución que nos ofrece el IDE es justamente convertir a la clase Animal en una clase abstracta.

```
class Animal {  
  
    public Animal() {  
        System.out.println("jajaja! soy un animal!!!!");  
    }  
    public abstract String emitirSonido();  
}
```