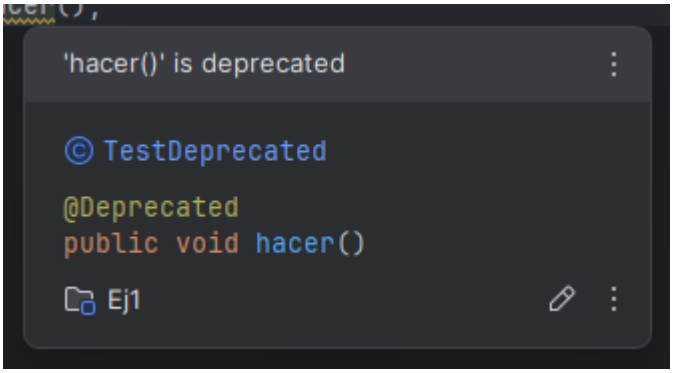
1.- Analice qué ocurre con la siguiente clase cuando se compila:

```
public class TestSobreescritura {
	@Override
	public String tostring() {
		return super.toString() + " Testeando: 'Override'";
	}
}
```

No compila porque le pusimos el override y no overritea nada

a) ¿Qué ocurre cuando se ejecuta TestAnotaciones?
Anda bien, la anotación solamente nos dice que no se recomienda pero ejecuta
b) ¿Qué ocurre si se elimina @SuppressWarnings({"deprecation"})? ¿el resultado de la ejecución es el mismo?
Si, pero me avisa que ya no se usa



c) ¿Cuál es la diferencia entre anotar el método testarYa() y anotar la clase TestAnotaciones?

En el método: solo **suprime el warning** por uso de APIs deprecated **dentro de** `testearYa()`. Es el alcance más preciso (mejor práctica).
En la clase: suprime **todos** los warnings de deprecación **en cualquier método de esa clase.** Es más cómodo pero **oculta** posibles usos obsoletos en otros lugares, lo que no suele ser deseable.

2.- Implementar una clase que mapee un objeto Bean a un archivo del filesystem y almacene en el archivo:
1. El nombre de la clase.
2. Los nombres de los atributos y el contenido de los mismos.
Las anotaciones que entiende son las siguientes:
● @Archivo(name="nombre.extension") -- Indica que la información se almacenará en el archivo nombre.extension. Si no se explicita un nombre se utiliza el nombre de la clase.
● @AlmacenarAtributo –– Denota que el nombre del atributo que está a continuación de la anotación se debe almacenar en el archivo.