


Open · Opened 2 weeks ago by  **Rubén Montero**

Usando atributos, de verdad

Resumen

- Hablaremos de métodos `getters` y `setters`
- Añadiremos un método `getter` a nuestra clase `Person`
- Añadiremos un método `greeting()` que devuelva un `String` compuesto

Descripción

Hemos usado atributos, pero de momento sólo los hemos **escrito**.

La finalidad de almacenar un **estado** en las instancias de nuestras clases es que *sirva* para algo.

Si es `private`, ¿de qué sirve? ¿Cómo podemos *leerlo* fuera de la clase?

Un método `getter`

La respuesta es: Escribiendo un método `getter`. Suele tener el prefijo `get` en su nombre, seguido del nombre del atributo.

Por ejemplo, en `Person.java`:

```
public class Person {  
    private String name;  
  
    public Person(String name) {  
        System.out.println(name + " acaba de ser instanciado");  
        this.name = name;  
    }  
  
    public String getName() {  
        return this.name;  
    }  
}
```

Este ejemplo demuestra que podemos tener una clase `Person` con un atributo `name`. Este atributo *sólo se puede modificar una vez*, indicando el valor inicial al **construir** la clase.

Después, únicamente se puede *leer* invocando `getName()`.

Pero en serio

Todo esto, ¿para qué?

La fuerza de la programación orientada a objetos (POO) es la **encapsulación**.

Es decir, escribimos una clase que es **responsable** de algo, y **sólo** esa clase es la que tiene dicha responsabilidad.

Un ejemplo

Escribamos un nuevo método en `Person.java`:

- `public`
- Devuelve `String`
- Se llama `greeting`

...que funcione así:

```
public String greeting() {  
    return "Hola " + this.name;  
}
```

Este método ilustra el **encapsulamiento**.

Aquí, la clase `Person` es responsable de *cómo se saluda* a una persona.

Imagina que `Person.java` es parte de una aplicación que muestra una página web.

En la página web, hay diferentes sitios donde se saluda al usuario después de *loguearse* y se le dice:

Hola {usuario}

Los beneficios:

- Si el día de mañana se quiere cambiar el mensaje por **Bienvenido, {usuario}**, sólo hay que hacerlo en un sitio
- Si se encuentra un problema con el mensaje (e.g. Falta una coma), se sabe **dónde** arreglarlo

La tarea

Se pide que añadas los métodos `getName` y `greeting` como en los ejemplos anteriores, con la siguiente modificación:

- El método `greeting` devolverá `"Hola, " + this.name;`.

Por último

Una vez verifiques que el `test` funciona correctamente, la tarea ha sido completada.

Haz `commit` y `push` para subir los cambios al repositorio.



Rubén Montero [@ruben.montero](#) changed milestone to [%Sprint 1](#) 2 weeks ago



Ania Blanco [@ania.blanco](#) mentioned in commit [b6fa8620](#) 2 weeks ago