


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

Usando atributos

Resumen

- Introduciremos los *atributos*
- Añadiremos un *atributo* `name` a nuestra clase `Person`
- Lo inicializaremos en el método **constructor**

Descripción

En la primera tarea, comenzamos diciendo que una clase es un **tipo especial** de dato.

Hasta ahora, hemos usado nuestras *clases* para implementar métodos que consigan distintos **comportamientos** (lógica).

Pero la segunda parte **esencial** de las clases son los **atributos**.

Así, se dice que una clase almacena un **estado** (en sus atributos) y tiene un **comportamiento** (gracias a sus métodos).

¿Por ejemplo?

La idea es que una clase *represente* algo necesario en nuestra aplicación.

Imagina que nuestro programa trabaja con **nombres de personas**:

```
String name1 = "Lisa Simpson";
String name2 = "Homer J. Simpson";
String name3 = "María del Carmen Jiménez";
```

¿Qué pasa si necesitamos *distinguir* los nombres de los apellidos?

Si, **por ejemplo**, definiéramos una clase `Name` que almacenase **2** atributos distintos, el problema estaría solventado.

```
public class Name {
    String firstName;
    String surname;

    // Aquí la clase podría tener métodos
}
```

Como puedes observar, los **atributos** de una clase se declaran como "variables" corrientes y molientes, dentro de ella.

Lo especial es que **pertenecen** a esa clase. O mejor dicho, a la *instancia*.

¿Cómo se usan los atributos?

Se accede a ellos (para *recuperarlos* o *asignarlos*) mediante un punto (`.`), igual que los métodos.

Por ejemplo:

```
Name name1 = new Name();
name1.firstName = "Lisa";
name1.surname = "Simpson";

Name name2 = new Name();
name2.firstName = "Homer";
name2.surname = "J. Simpson";

Name name3 = new Name();
name3.firstName = "María del Carmen";
name3.surname = "Jiménez";
```

En la práctica los atributos suelen ser privados

El ejemplo anterior no se suele aplicar en la realidad.

Lo normal es que los **atributos** se *inicialicen* a través del **método constructor**.

Se suele desear que los **atributos** sean privados (`private`) para que no puedan alterarse desde fuera:

```
public class Name {  
    private String firstName;  
    private String surname;  
  
    // Método constructor  
    public Name(String firstName, String surname) {  
        this.firstName = firstName;  
        this.surname = surname;  
    }  
}
```

¿Comprendes este ejemplo?

¿Ves para qué se usa `this` ?

Hhhmmmmmm... Sí...

Quizá se vea más claro si reescribimos el código anterior de forma coherente.

Quedaría así:

```
Name name1 = new Name("Lisa", "Simpson");  
Name name2 = new Name("Homer", "J. Simpson");  
Name name3 = new Name("María del Carmen", "Jiménez");
```

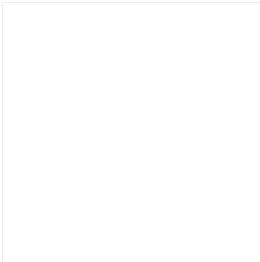
Estos ejemplos son **esenciales** y es importante comprender el uso del método constructor y de atributos privados en una clase Java.

Otro ejemplo más...

Supongamos que existe de una clase `Periodico` :

```
public class Periodico {  
    private String nombre;  
    private int dia;  
    private int mes;  
    private int año;  
  
    public Periodico(String nombre, int dia, int mes, int año) {  
        this.nombre = nombre;  
        this.dia = dia;  
        this.mes = mes;  
        this.año = año;  
    }  
}
```

Cada *instancia* sería como un periódico con distintos valores:



La tarea

Se pide que, en la clase `Person` :

- Añadas un atributo `private` llamado `name`
- En el método constructor, lo inicialices (ademas de hacer un *print*), usando `this.name = 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 [79e85fdf](#) 2 weeks ago