¿Qué es un patrón de diseño y para qué sirve?

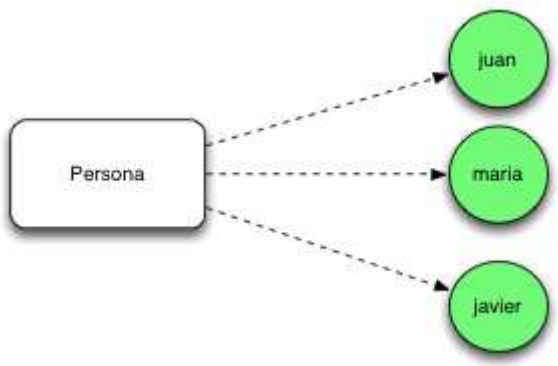
Los **patrones de diseño** son soluciones **para** problemas típicos y recurrentes que nos podemos encontrar a la hora de desarrollar una aplicación. Aunque nuestra aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos, operaciones entre sistemas etc.

Consulta esta dirección:

https://jesusramirezguerrero.com/2014/08/26/patrones-de-diseno-en-java/

Ejemplo de Java Singleton

Este patrón de diseño se encarga de que una clase determinada únicamente pueda tener un único objeto. Normalmente una clase puede instanciar todos los objetos que necesite.



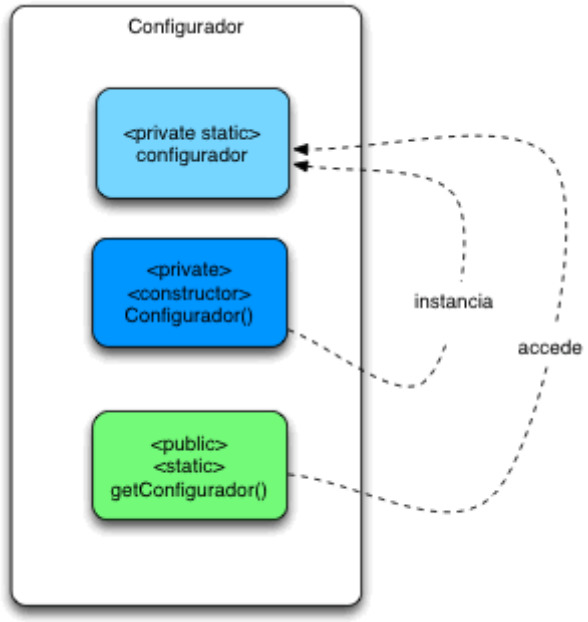
Sin embargo, una clase que siga el patrón Singleton tiene la peculiaridad de que solo puede instanciar un único objeto. Este tipo de clases son habituales en temas como configurar parámetros generales de la aplicación ya que una vez instanciado el objeto los valores se mantienen y son compartidos por toda la aplicación. Vamos a configurar una clase con el patrón Singleton, a esta clase la llamaremos Configurador.

Java Singleton

Una vez que tenemos claro cuál es el concepto de Configurador vamos a crearlo en código. En este caso nuestro configurador almacenará dos valores url, y base de datos que serán compartidos por el resto de Clases de la aplicación.

```
1
2
3 package com.arquitecturajava;
4
5 public class Configurador {
6
7 private String url;
8  private String baseDatos;
9  private static final Configurador miconfigurador;
10
11  public  static Configurador getConfigurador(String url,String baseDatos) {
12
13  if (miconfigurador==null) {
14
15  miconfigurador=new Configurador(url,baseDatos);
16  }
17  return miconfigurador;
18  }
19
20  private Configurador(String url,String baseDatos){
21
22  this.url=url;
23  this.baseDatos=baseDatos;
24
25  }
26
27 public String getUrl() {
28  return url;
29  }
30
31 public void setUrl(String url) {
32  this.url = url;
33  }
34
35 public String getBaseDatos() {
36  return baseDatos;
37  }
38
39 public void setBaseDatos(String baseDatos) {
40  this.baseDatos = baseDatos;
41  }
42 }
```

Para conseguir que una clase sea de tipo Singleton necesitamos en primer lugar que **su constructor sea privado.** De esa forma ningún programa será capaz de construir objetos de esta tipo. En segundo lugar necesitaremos disponer de una **variable estática privada** que almacene una referencia al objeto que vamos a crear a través del constructor. Por último, **un método estático publico que se encarga de instanciar el objeto la primera vez y almacenarlo en la variable estática.**



Una vez aclarado como funciona un Singleton es muy sencillo utilizarle desde un programa ya que basta con invocar al método estático. No creo ningún objeto desde la clase principal.

```
1
2 package com.arquitecturajava;
3
4 public class Principal {
5
6 public static void main(String[] args) {
7
8  Configurador c= Configurador.getConfigurador("miurl", "mibaseDatos");
9
10  System.out.println(c.getUrl());
11
12  System.out.println(c.getBaseDatos());
13
14 }
15  }
```