**Tema: Patrones de diseño**

[**Singleton**](https://migranitodejava.blogspot.com/2011/05/singleton.html)

La idea del patrón Singleton es proveer un mecanismo para limitar el número de instancias de una clase. Por lo tanto el mismo objeto es siempre compartido por distintas partes del código. Puede ser visto como una solución más elegante para una variable global porque los datos son abstraídos por detrás de la interfaz que publica la clase singleton. Dicho de otra manera, esta patrón busca garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.  
  
Usaremos este patrón cuando:  
♦ Debe haber exactamente una instancia de una clase y deba ser accesible a los clientes desde un punto de acceso conocido.  
♦ Se requiere de un acceso estandarizado y conocido públicamente.  
  
Sus usos más comunes son clases que representan objetos unívocos. Por ejemplo, si hay un servidor que necesita ser representado mediante un objeto, este debería ser único, es decir, debería existir una sola instancia y el resto de las clases deberían de comunicarse con el mismo servidor. Un Calendario, por ejemplo, también es único para todos. No debe utilizarse cuando una clase esta representando a un objeto que no es único, por ejemplo, la clase Persona no debería ser Singleton, ya que representa a una persona real y cada persona tiene su propio nombre, edad, domicilio, DNI, etc.  
  
  
Diagrama UML

[A screenshot of a computer program

Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjB2PcvHsmNkkE6IEWvmcP9UnPzSqmzIw6G4YJuL1YTbEFXli6HeWhLkT9lCHmM1BYPcSDxh_ZFyQY67NNMtC9LXBAV18iCWhkyuhdCN-Km7quNrna50WxKQuoefpyA9VuPWJ6dlH68wmw/s1600/image00.jpg)

En el diagrama, la clase que es Singleton define una instancia para que los clientes puedan accederla. Esta instancia es accedida mediante un método de clase.  
Los clientes (quienes quieren acceder a la clase Singleton) acceden a la única instancia mediante un método llamado getInstance().  
  
Ejemplo.  
  
Como dije anteriormente, este patrón es ideal para aquellas clases que representan objetos únicos. Por ejemplo, un instituto educativo es un objeto único. No deberíamos crear muchas instancias de esta clase ya que al hacer esto estaríamos diciendo que hay varios institutos educativos. Caso contrario serían los alumnos que asisten a dicho instituto. Debería haber un objeto por cada uno de los alumnos, ya que todos ellos tienen propiedades distintivas, desde el nombre hasta el documento de identidad. Pero todos los alumnos deberían comunicarse con el mismo instituto.  
  
Entonces, haremos que el instituto aplique el patrón Singleton:

A screenshot of a computer

Description automatically generated

Para llamar al instituto debemos hacer lo siguiente:A computer screen shot of a computer code

Description automatically generated