**PATRONES DE DISEÑO I**

**Introducción**:

**Dificultades/Retos de Diseño:**

* El diseño de un sistema/programa se lleva a cabo a través de prueba y error. Lo cual tiene un costo.
* Diseñar software para reusar en el futuro es difícil:
* Flexibilidad
* Modularidad

En consecuencia, esto hace difícil producir rápidamente un código eficiente.

**Patrones**:

* Al encontrar una solución eficiente a un problema, volvemos a utilizarla una y otra vez (**patrón**).
* Al conocer dicho patrón uno puede aplicarlo directamente sin la necesidad de redescubrirlo **(no más prueba y error**).
* El patrón podría aplicarse en cualquier entorno (**flexible**) y de manera sencilla (**modulado**). 🡪 ES COMO UNA SOLUCION QUE VOY A PODER UTILIZARLO EN CUALQUIER OTRA TAREA

**¿Qué son los Patrones de Diseño?**

* Un Patrón de Diseño es una **solución general** y **re-usable a** un problema que ocurre de manera recurrente en el diseño de software.
* No es un algoritmo/pseudocódigo, es una descripción de cómo resolver un determinado problema.
* En patrones de diseño orientado a objetos, únicamente se describen las relaciones entre las clases y objetos. No se especifica el objetivo final de las clases u objetos.

**Elementos**:

* **El nombre**
* Un identificador que, de una idea del problema, la solución y la consecuencia.
* **El problema**
* Un identificador que, de una idea del problema, la solución y la consecuencia.
* **La solución**
* Se describen de manera general las partes del patrón, y las relaciones entre estos.
* **La consecuencia**
* Se describen de manera general las partes del patrón, y las relaciones entre estos.

**Descripción:**

* **Nombre del patrón y categoría**
* **Propósito**:
* ¿Qué hace el patrón? ¿Qué soluciona?
* **Álias**:
* ¿Tiene algún otro nombre?
* **Motivación**:
* Un escenario demostrando el uso del patrón
* **Aplicabilidad**:
* ¿En qué escenarios? ¿Cómo identificar estos escenarios?
* **Estructura**:
* Una representación gráfica de las clases en el patrón
* **Participantes**:
* Los objetos y clases que lo conforman
* **Consecuencias**:
* Los resultados de usar el patrón
* **Implementación**:
* Sugerencias al implementar el patrón
* **Código**:
* Un ejemplo de cómo implementarlo en C++
* **Usos** **conocidos**:
* Aplicaciones en sistemas reales
* **Patrones** **relacionados**:
* ¿Qué otros patrones se relacionan con él?

**Categorías:**

* **De Creación (Creational)**

Son patrones de diseño que proveen de varios mecanismos de creación de objetos con el objetivo de incrementar su flexibilidad y reutilización.

* **De Estructura (Structural)**  
  Son patrones de diseño que facilitan la composición de clases y objetos para formar estructuras más grandes.
* **De Comportamiento (Behavioral)**  
  Se encargan de la asignación de responsabilidades entre objetos

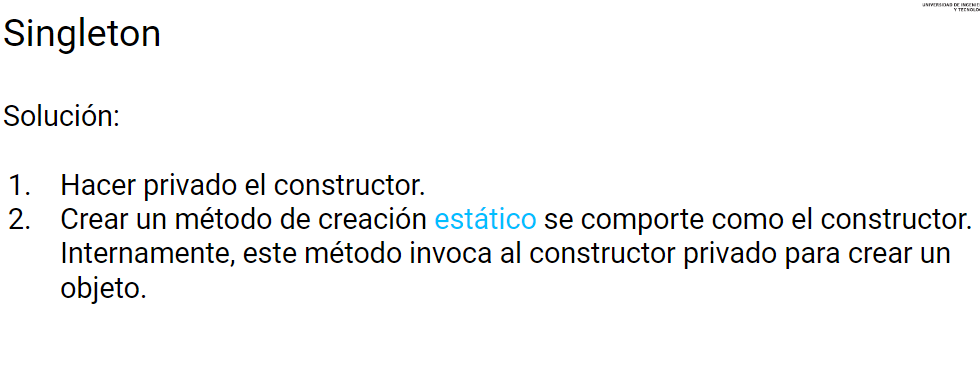
**Patrones de Diseño:**

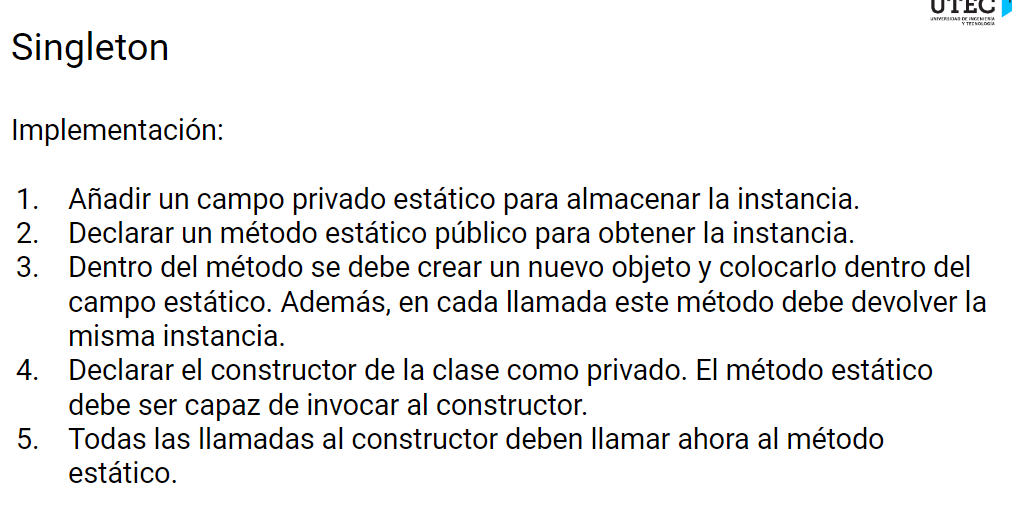
* **Singleton:  
  Propósito**
* Este patrón nos permite asegurar que una clase tenga solo una instancia. Al mismo tiempo este permite un acceso global a la instancia.

**Aplicabilidad**

* Controlar el acceso a un recurso compartido (base de datos)

**Patrones relacionados**

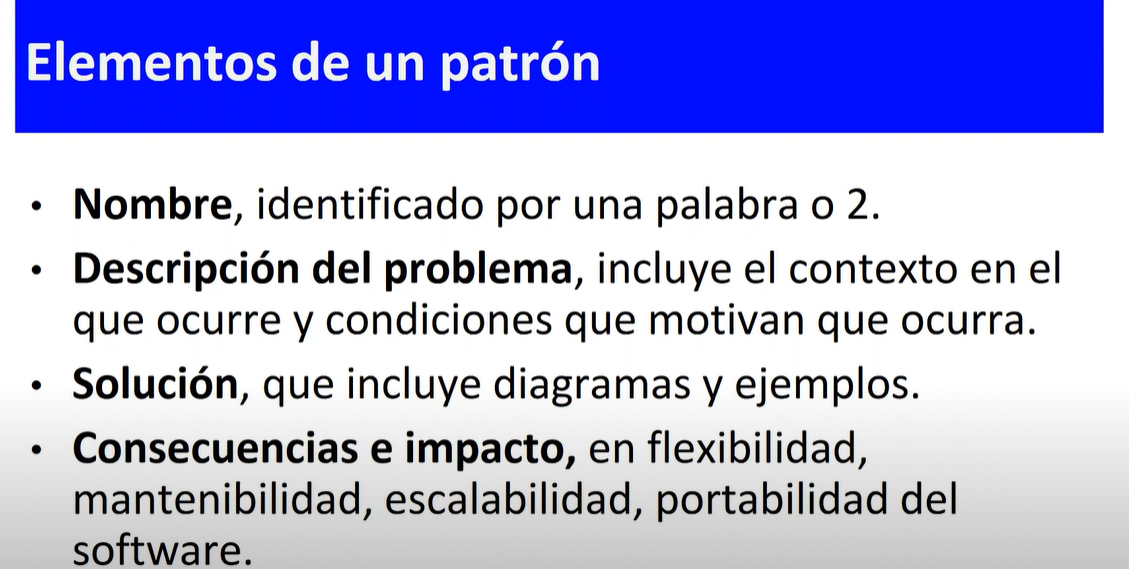
* Abstract Factory
* Builder
* Prototype



**Patrones de Diseño:**

Es una especie de plantilla, de forma que lo podamos reutilizar.   
Describe un problema que ocurre repetitivamente en el diseño y la solución del problema, de modo que pueda ser reusado en la solución de millones de casos.

Lo que se busca es una solución a problema que ocurren repetitivamente, pero una solución que pueda ser reutilizable

Para ello debe considera estos 4 elementos



Se clasifica en tres grupos:  
Un poco asociado a los componentes y clases, ya que también disponen de estos 3 elementos

Todos se pueden crear, nosotros creamos los objetos, tiene una estructura que esta definida normalmente por los atributos y tiene un comportamiento que esta normalmente definido por los métodos.

Entonces los patrones tratan de atacar estos 3 elementos:

* Creación:

Problemas al momento de tratar de crear un objeto, y cuales serian los patrones recomendables de utilizar para mejorar el proceso de creación

* Estructurales:

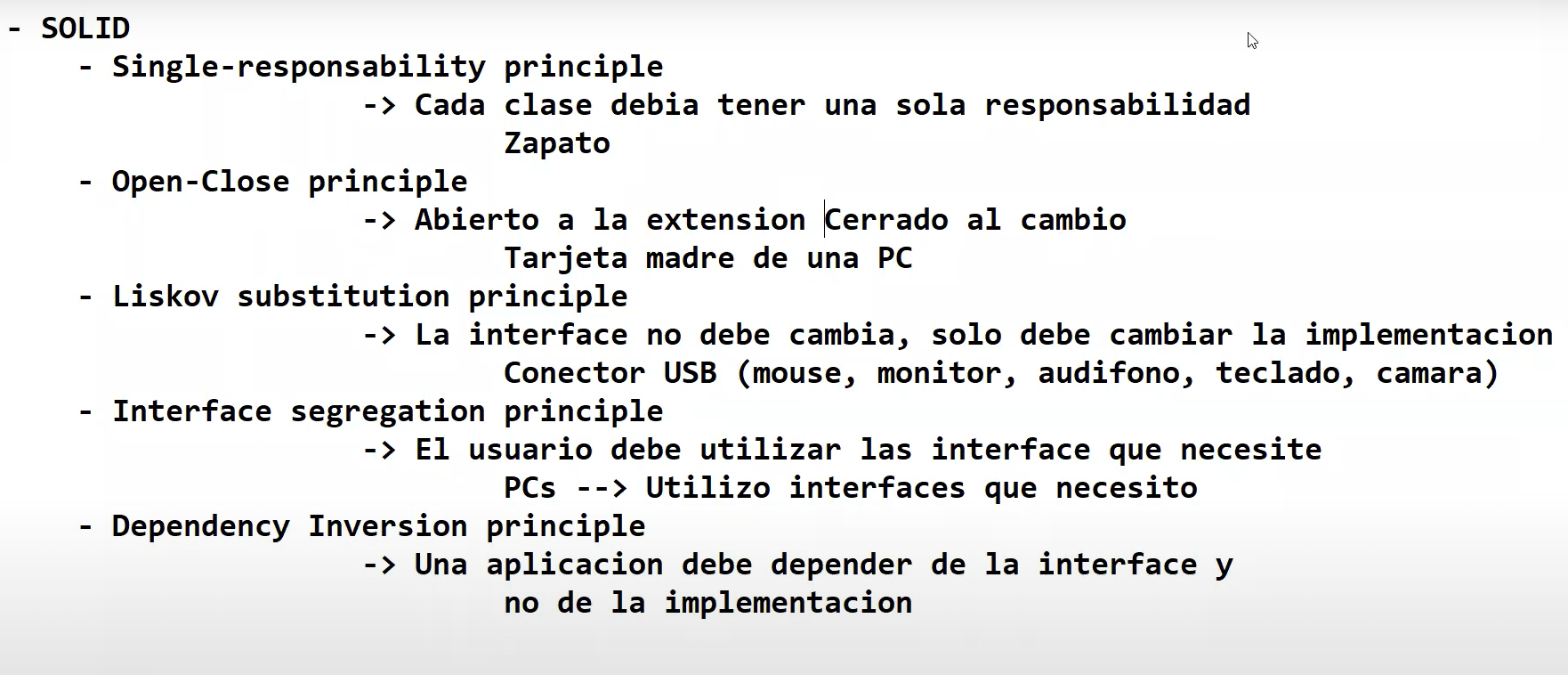
A veces puede ocurrir que una clase tiene ciertos atributos y ciertos métodos, y esa clase no se ajusta exactamente a la solución. Entonces se buscará acomodar la estructura para que se pueda adaptar a un problema determinado. Tendremos que buscar algún patrón estructural que nos permita realizar esta adaptación o ese cambio o ese ajuste del componente o la clase.

* Comportamiento:

El problema de comportamiento esta en la forma de como se llama a las acciones o los métodos de una clase, las clases o métodos tienen comportamientos, entonces hay comportamientos que posiblemente no estén incluidos dentro de la clase

Respecto a los patrones y a la reutilización, primero hay un principio de diseño que se llama:

**El principio KIS** (Keep it Simple) - (Mantenlo simple):

* Cuando ya las cosas se hacen complicadas, es que estamos mal, y ahí se mantiene el principio de KIS