Patrones de Creación

Unidad 4. Patrones de Diseño de Software

Patrones de Creación

- Se enfocan en la manera en que se crean objetos
- Se utilizan cuando se debe tomar una decisión sobre cómo instanciar una clase

Características

- Encapsulan el conocimiento de las clases que son utilizadas en el sistema
- Ocultan la manera en que los objetos son creados

Necesidad de los patrones

- Evitan crear o instanciar objetos de manera incorrecta
- Son útiles cuando hay objetos que deben ser creados en una sola ocasión

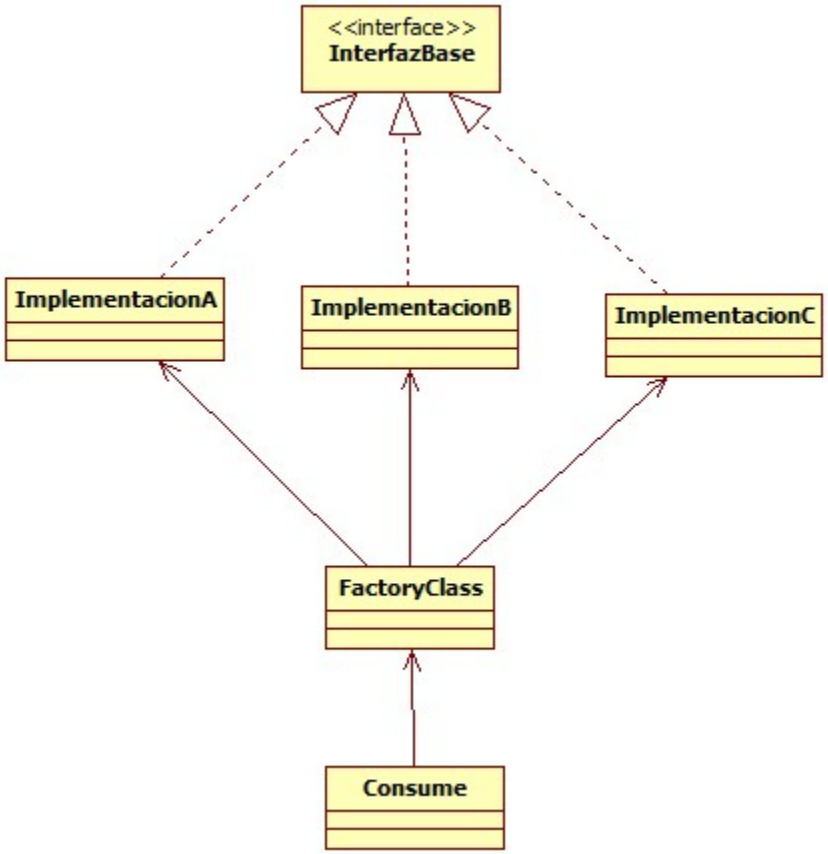
Patrones de Creación

- *Factory*
- *Builder*
- *Prototype*
- *Singleton*
- *Composite*
- *Proxy*

Patrón *Factory*

- Se utiliza cuando a partir de una super-clase, se quieren generar distintos tipos de sub-clases según la necesidad

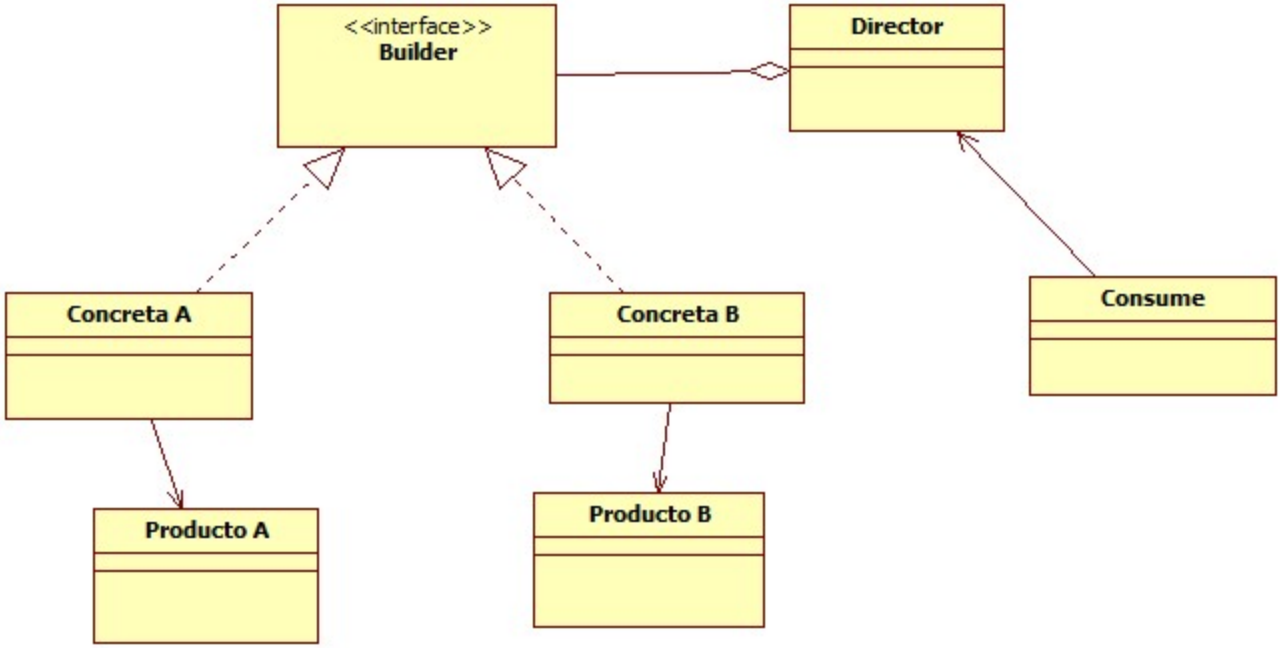
Diagrama del patrón *Factory*



Patrón *Builder*

- Se utiliza cuando se quiere construir un objeto complejo utilizando objetos simples
- Evita tener que usar una gran cantidad de parámetros en los constructores

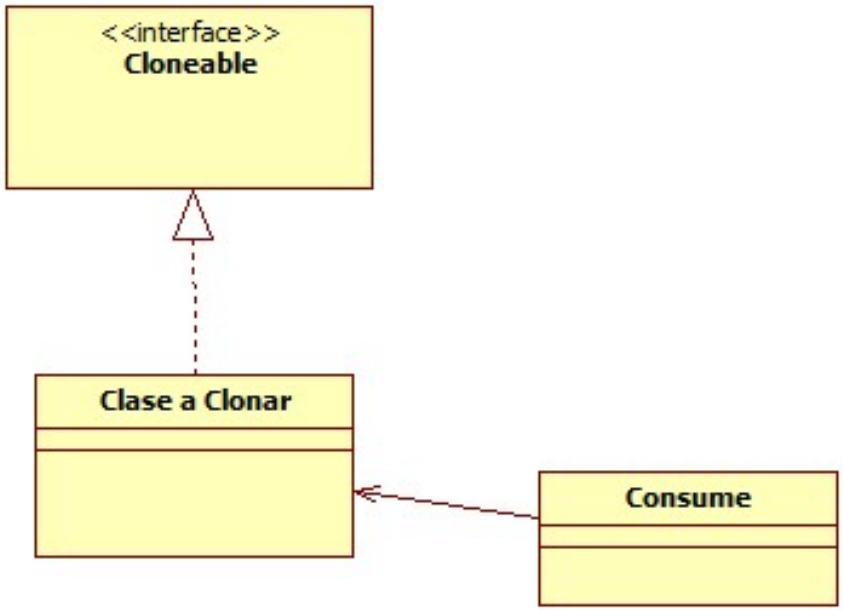
Diagrama de patrón *Builder*



Patrón *Prototype*

- Permite crear nuevos objetos a partir de objetos ya existentes
- Se utiliza cuando la creación de un objeto implica un costo muy alto y se quiere un objeto del mismo tipo

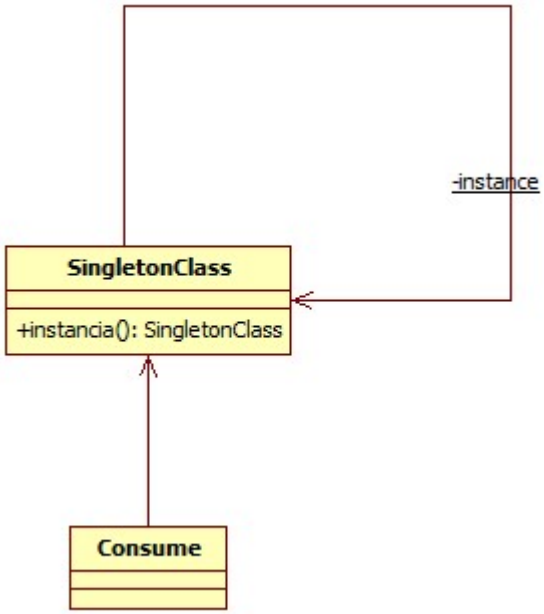
Diagrama del patrón *Prototype*



Patrón *Singleton*

- Se encarga de la creación de objetos cuando se desea que solo exista una instancia en tiempo de ejecución

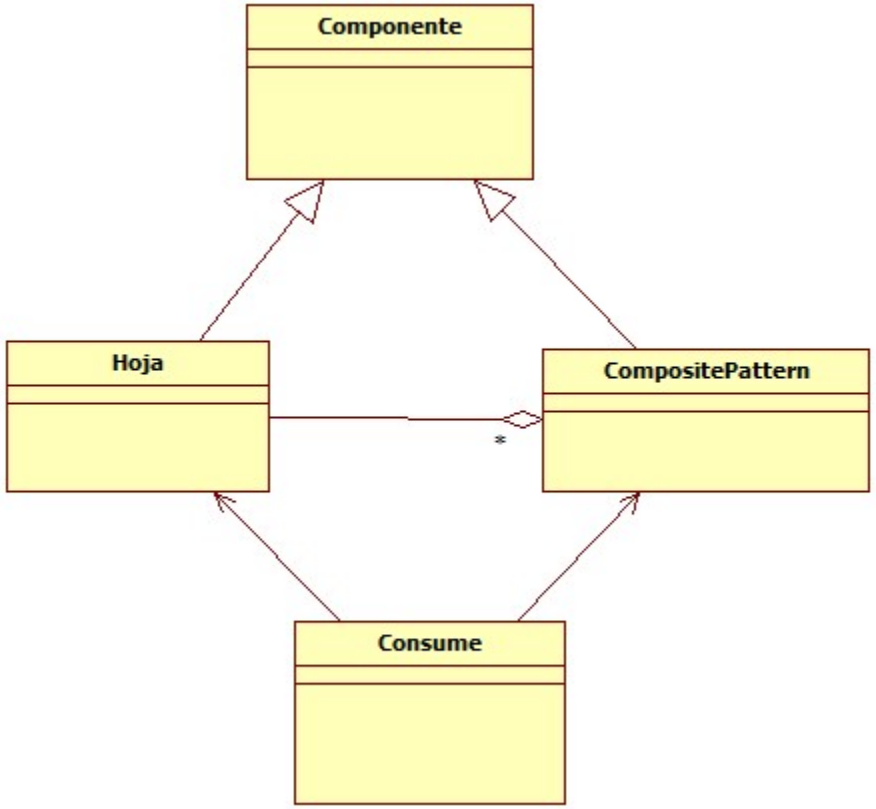
Diagrama del patrón *Singleton*



Patrón *Composite*

- Se utiliza cuando se quiere crear un objeto que está compuesto por elementos de su misma clase

Diagrama del patrón *Composite*



Patrón *Proxy*

- Ofrece una forma de acceder a la funcionalidad de un determinado objeto
- Una clase contiene la funcionalidad de otra clase

Diagrama del patrón *Proxy*