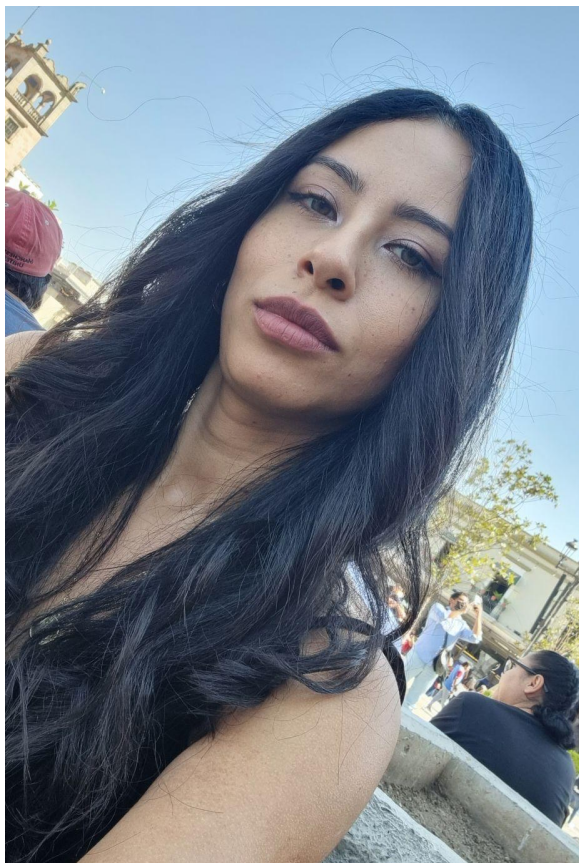




Patrones de diseño

Diana Fernández

@LizFdez



Diana Fernández

- Senior Software Engineer
- Mayormente enfocada en tecnologías backend como Java y AWS
- Leer, bailar y viajar

Notas Importantes



Identifícate en Zoom, usando tu nombre y apellido



Silencia tu micrófono durante el recorrido



Use el chat para preguntas durante las seccion de preguntas y respuestas



Haz preguntas sobre el tema presentado

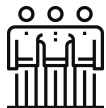


En caso de problemas de conexión apaga tu cámara

Código de conducta de la Academia



Sea respetuoso, no hay malas preguntas o ideas.



Sea acogedor y paciente



Ten cuidado con las palabras que eliges

Objetivos de la sesión

Al finalizar esta sesión, podrás:

- Comprender el propósito y el uso de los patrones de diseño, para elegir e implementar el patrón correcto según sea necesario.
- Analizar un código y determinar qué patrón de creación se ha aplicado.
- Saber que patrón de creación usar dado un problema actual.

Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



Abstract Factory



Builder



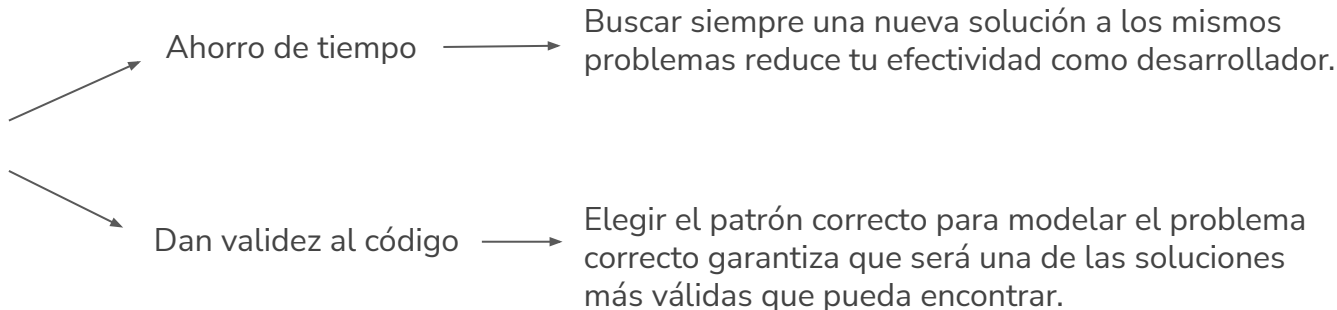


Patrones de diseño

DEFINICION

- Los patrones de diseño son **soluciones típicas a problemas comunes** en el diseño de software.
- Cada patrón es como un **modelo** que se puede **personalizar para resolver un problema de diseño** particular en su código.
- No es obligatorio implementar siempre patrones de diseño en su proyecto. Los patrones de diseño no están destinados al desarrollo de proyectos. Los patrones de diseño están destinados a la resolución de problemas comunes.

BENEFICIOS



- Se diferencian por su complejidad, nivel de detalle y resolución de problemas.
- Se pueden categorizar por su intención y dividirse en tres grupos:

Patrones de diseño

Patrones Creacionales

Proporciona formas de crear objetos.

Patrones Estructurales

Define cómo deben estructurarse las clases y los objetos.

Patrones de Comportamiento

Define cómo interactúan los objetos entre sí.

Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



Abstract Factory



Builder



Patrones Creacionales

DEFINICION

- Los patrones de creación tienen que ver con la **instanciación de clases o la creación de objetos**.
- Estos patrones enfatizan la encapsulación de la lógica de instanciación, **ocultando los detalles concretos de cada objeto** y permitiéndonos trabajar con abstracciones.

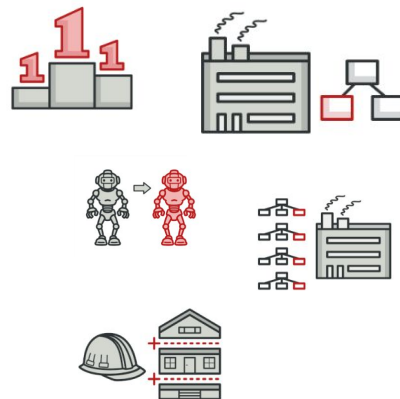
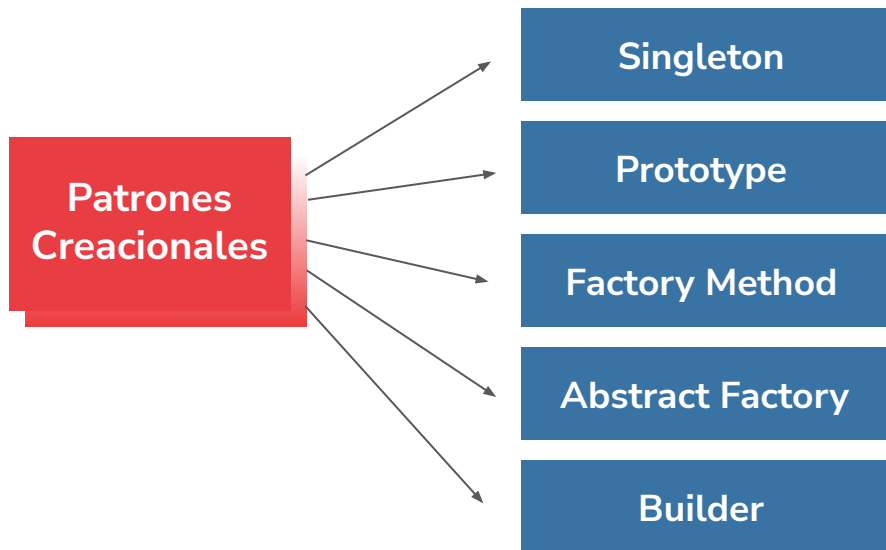


Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



Abstract Factory



Builder



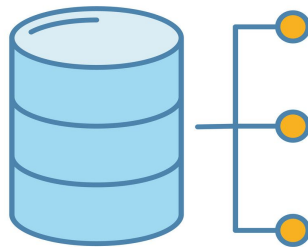
SINGLETON



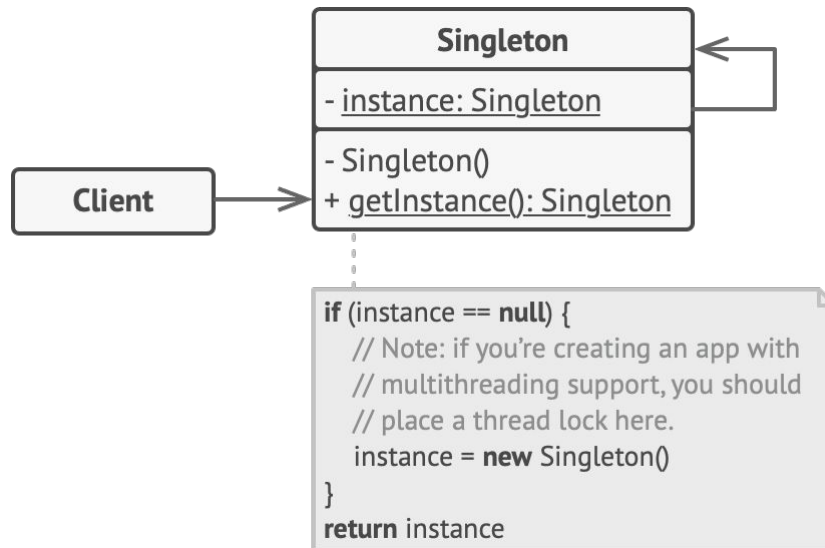
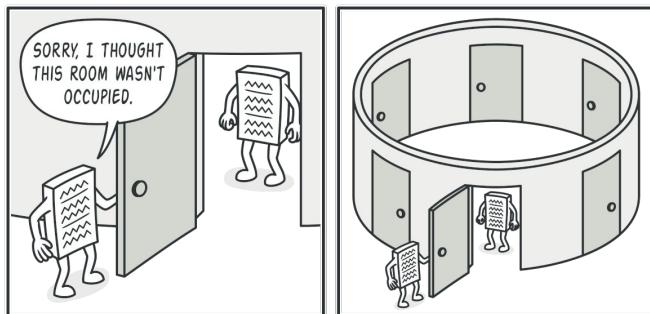
- Se aseguran de que una clase tenga ***solo una instancia***, mientras proporciona un ***punto de acceso global*** a esta instancia.

Ejemplo - Conexión a una Base de Datos

- Este comportamiento es imposible de implementar con un constructor regular ya que una llamada de constructor siempre debe devolver un nuevo objeto por diseño.



Shared Resource



¿Cuándo usar este patrón?

Utilice el patrón Singleton cuando una clase de su programa deba tener una única instancia disponible para todos los clientes; por ejemplo, un solo objeto de base de datos compartido por diferentes partes del programa.

Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



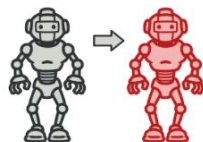
Abstract Factory



Builder

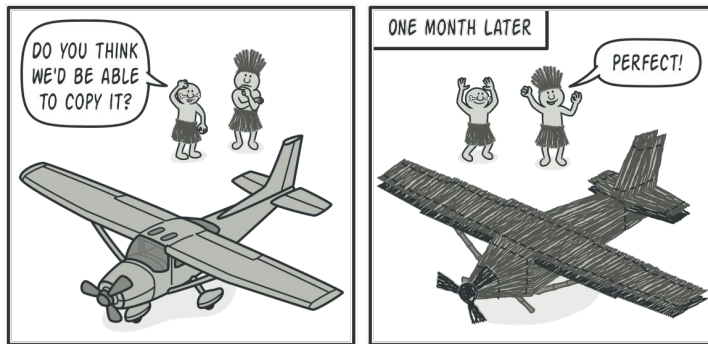


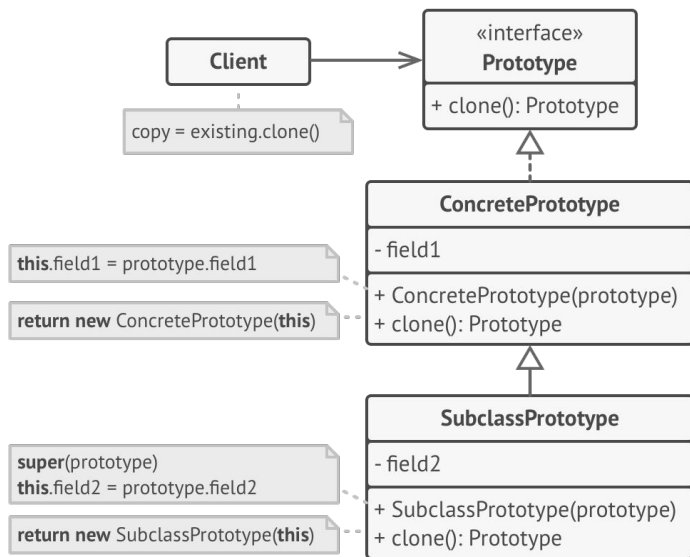
PROTOTYPE



- Le permite ***copiar (clonar) objetos existentes*** sin hacer que su código dependa de sus clases.

Ejemplo - Copia de un avión





¿Cuándo usar este patrón?

- En escenarios donde la aplicación necesita crear **una cantidad de instancias de una clase**, que **tiene casi el mismo estado**.
- Crear un objeto es una **operación costosa** y sería más eficiente copiar un objeto.
- Se requieran **objetos que sean similares a los objetos existentes** y que en su mayoría tengan campos inmutables.

Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



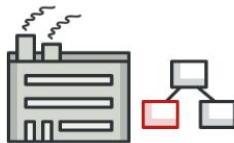
Abstract Factory



Builder



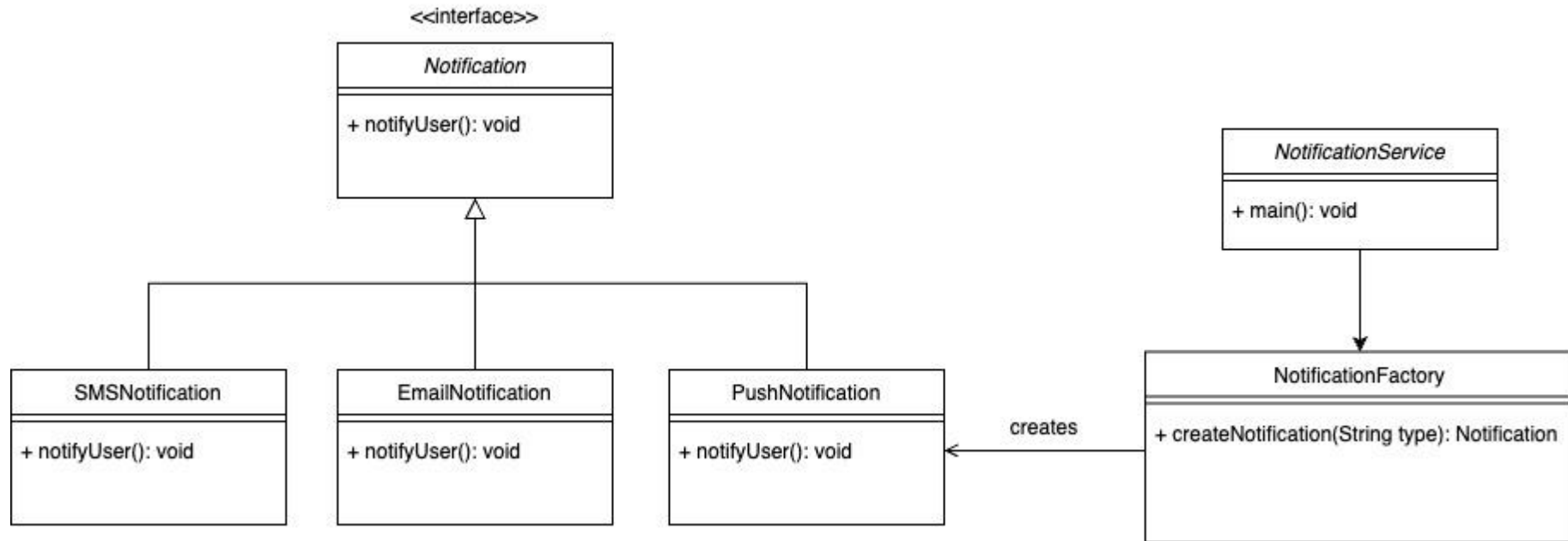
FACTORY METHOD



- Proporciona una interfaz (o una clase abstracta) para **crear objetos en una superclase** y **permite que las subclases decidan** qué clase instanciar.
- Es una de las mejores formas de crear un objeto donde la lógica de creación de objetos está oculta para el cliente.

Ejemplo - Notificación de Email





Email



SMS



Web Push



App Push

Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



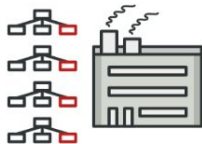
Abstract Factory



Builder












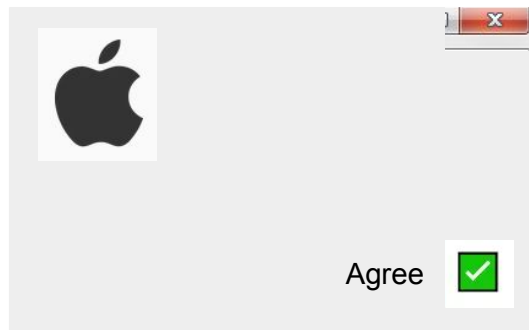
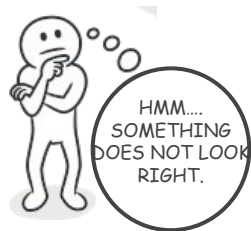
ABSTRACT FACTORY



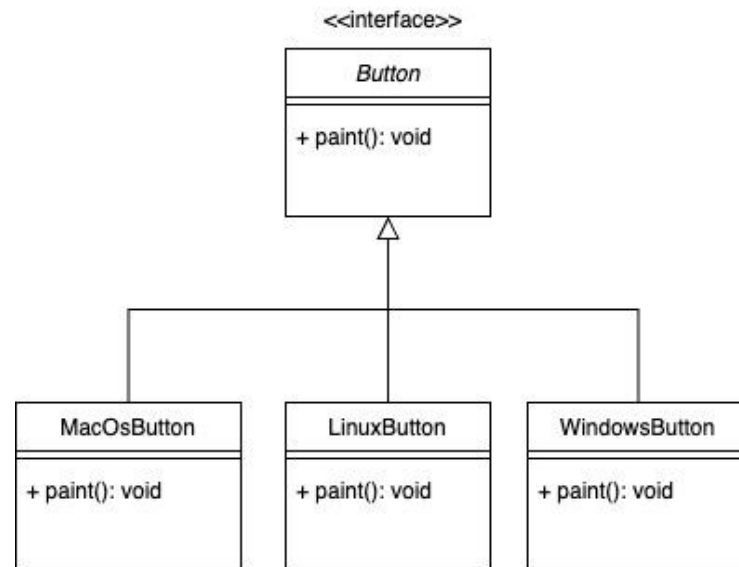
- Le permite ***producir familias de objetos relacionados*** sin especificar sus clases concretas.
- La fábrica abstracta también se denomina ***fábrica de fábricas***.

Ejemplo - Creación de una aplicación

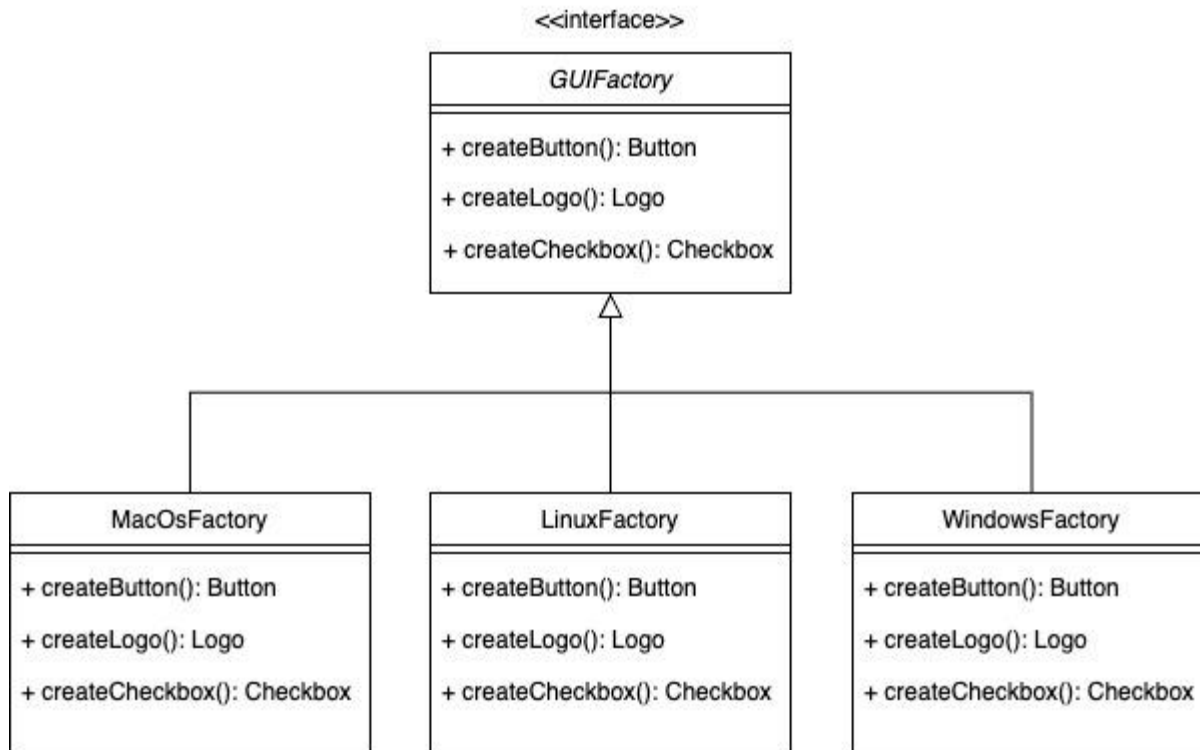
| | Logo | Button | Checkbox |
|---------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Windows |  |  |  |
| Mac |  |  |  |
| Linux |  |  |  |



- Lo primero que sugiere el patrón Abstract Factory es **declarar explícitamente interfaces para cada producto distinto de la familia de productos** (por ejemplo, logo, button o checkbox). Luego puede hacer que todas las variantes de productos sigan esas interfaces.



- El siguiente paso es **declarar Abstract Factory**, una interfaz con una **lista de métodos de creación para todos los productos que forman parte de la familia de productos**. Estos métodos deben devolver tipos de productos abstractos representados por las interfaces que extrajimos anteriormente: Button, Logo, Checkbox, etc.



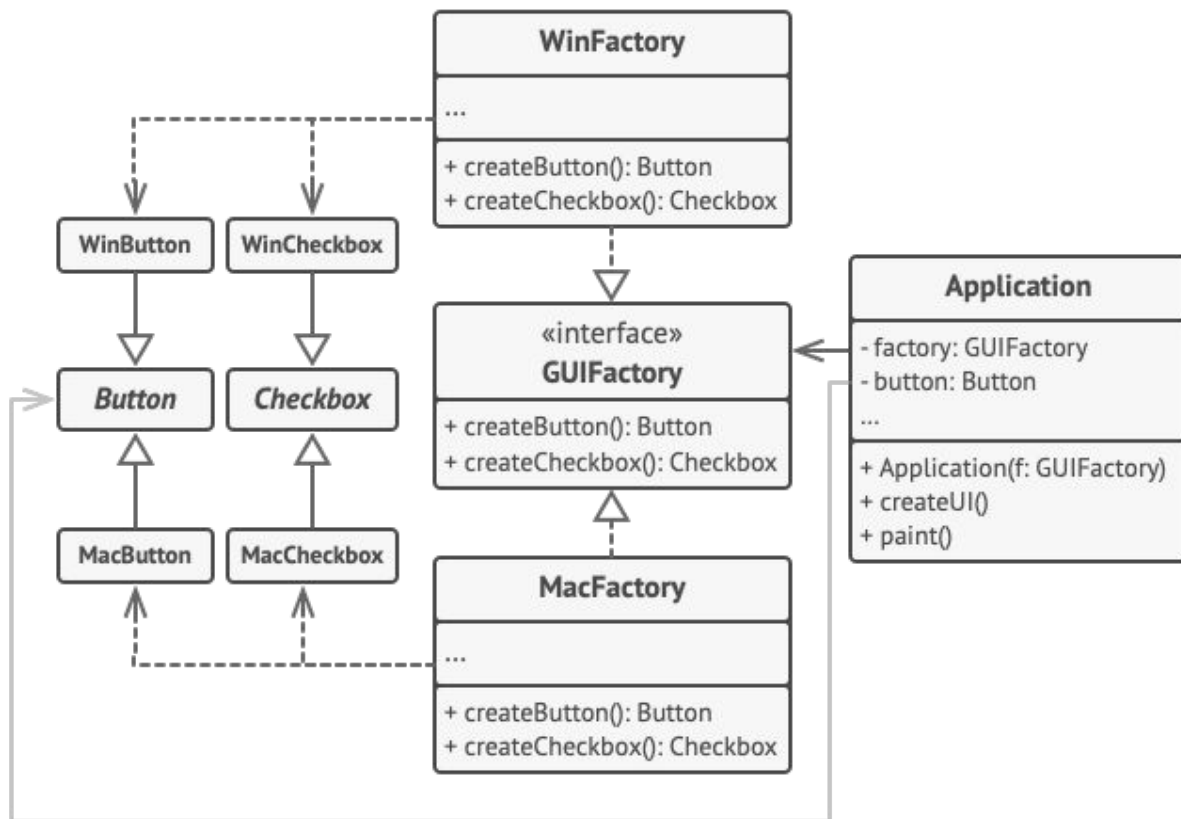


Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



Abstract Factory



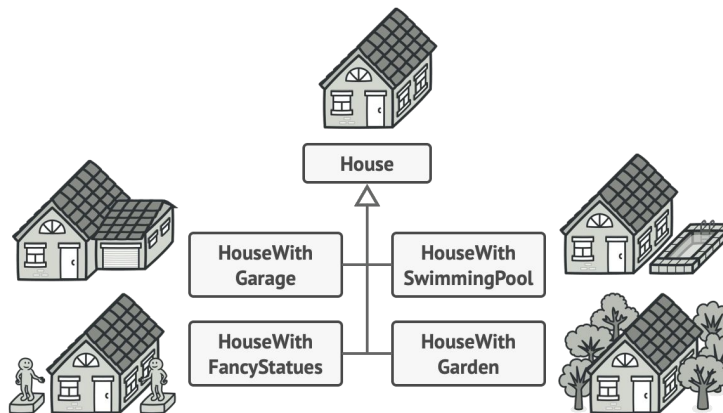
Builder



BUILDER



- Le permite **construir objetos complejos** paso a paso. El patrón le permite producir **diferentes tipos y representaciones de un objeto** utilizando el mismo código de construcción.
- El patrón Builder **oculta la complejidad** de la creación de objetos.
- Una clase delega la creación de objetos a un objeto Builder en lugar de crear los objetos directamente.



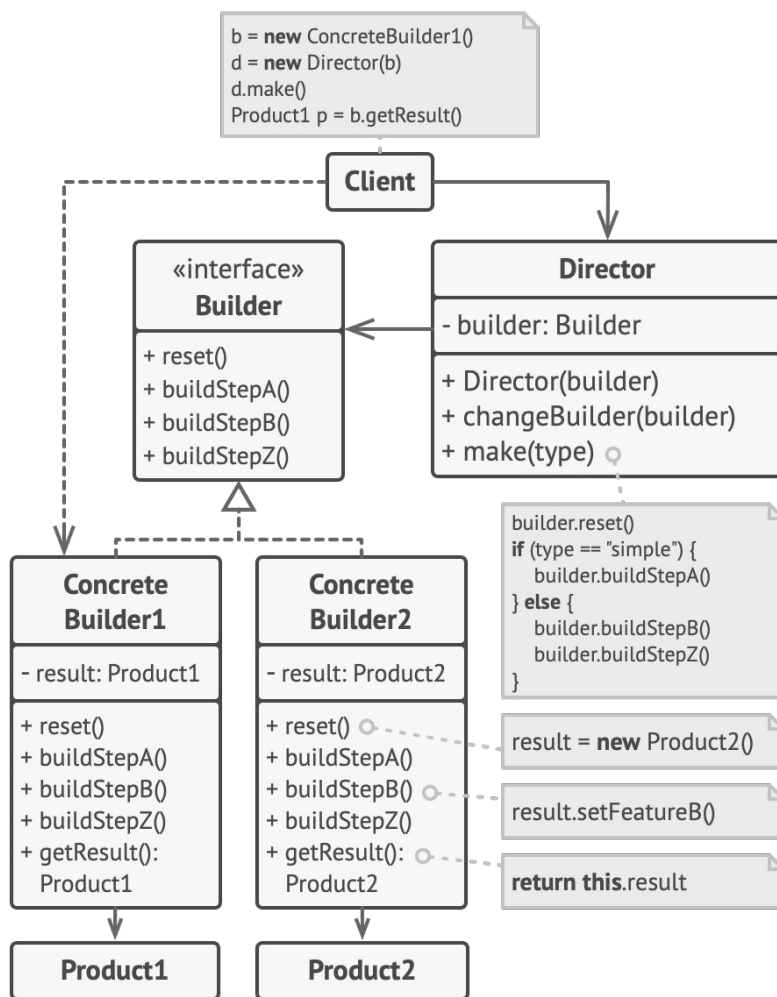


Tabla de Contenido

Patrones de Diseño



Patrones Creacionales



Singleton



Prototype



Factory Method



Abstract Factory



Builder



Q&A



Feedback Form

Let us know your feedback!

<https://forms.gle/WKtc8wZeSxWnjGo8A>





TM

Thank you