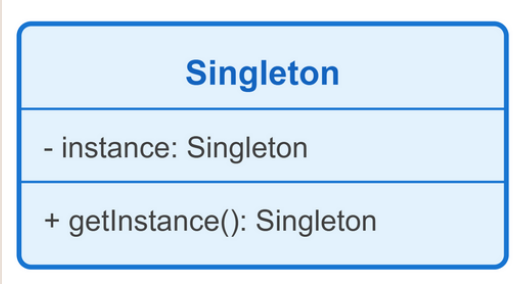
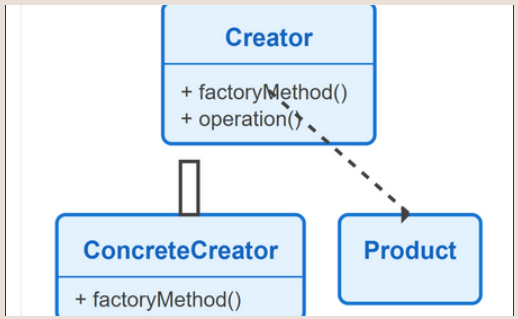
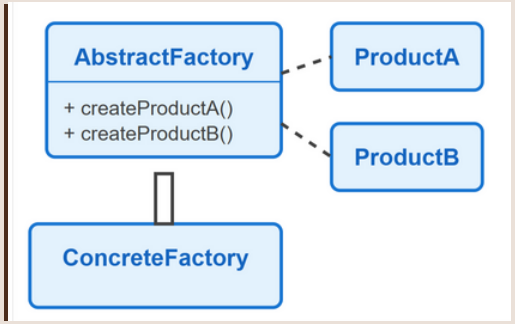
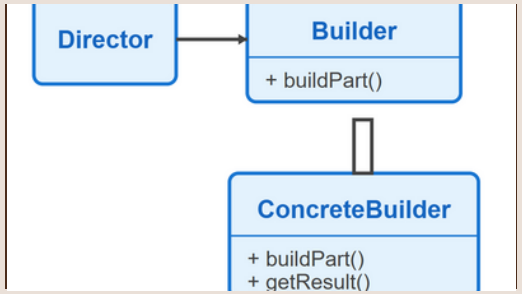
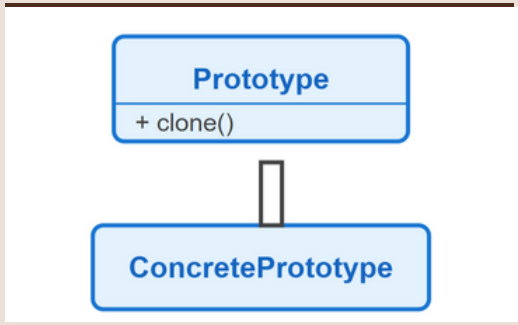
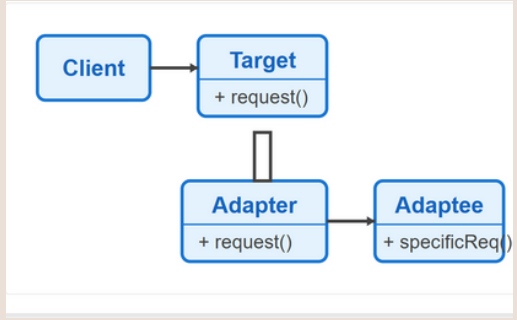
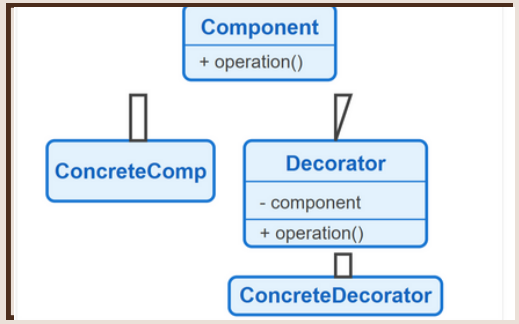
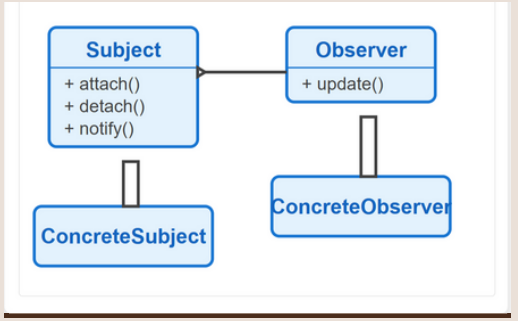
PATRONES DE DISEÑO

PATRONES	DESCRIPCIÓN	COMO USARLO	DIAGRAMA UML
SINGLETON	Garantiza que una clase tenga solo una instancia y proporciona un punto de acceso global a ella.	Cuando necesitas exactamente una instancia de una clase. -Para controlar el acceso a recursos compartidos -Cuando la instancia debe ser accesible desde cualquier punto de la aplicación.	
FACTORY METHOD	Define una interfaz para crear objetos, pero deja que las subclases decidan qué clase instanciar.	-Cuando no sabes de antemano el tipo exacto de objetos que necesitarás -Para delegar la creación de objetos a subclases -Cuando quieres proporcionar una biblioteca de clases sin exponer su lógica de instanciación	
ABSTRACT FACTORY	Proporciona una interfaz para crear familias de objetos relacionados sin especificar sus clases concretas.	-Cuando el sistema debe ser independiente de cómo se crean sus productos -Para trabajar con múltiples familias de productos relacionados -Cuando quieres garantizar que los productos de una familia se usen juntos	

BUILDER	Separa la construcción de un objeto complejo de su representación, permitiendo crear diferentes representaciones.	-Para crear objetos complejos paso a paso -Cuando el algoritmo de creación debe ser independiente de las partes que componen el objeto -Para construir objetos con muchos parámetros opcionales	
PROTOTYPE	Especifica los tipos de objetos a crear usando una instancia prototípica, creando nuevos objetos copiando este prototipo.	-Cuando la creación de objetos es costosa -Para evitar subclases de un objeto creador -Cuando las clases a instanciar se especifican en tiempo de ejecución	
ADAPTER	Convierte la interfaz de una clase en otra interfaz que los clientes esperan, permitiendo que clases incompatibles trabajen juntas.	-Para usar una clase existente cuya interfaz no coincide con la que necesitas -Para crear clases reutilizables que cooperen con clases sin interfaces compatibles -Para integrar bibliotecas de terceros en tu sistema	
DECORATOR	Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la herencia.	-Para añadir funcionalidades a objetos individuales sin afectar a otros -Cuando la extensión por herencia no es práctica -Para añadir o eliminar responsabilidades dinámicamente	

OBSERVER	Define una dependencia uno-a-muchos entre objetos, de manera que cuando un objeto cambia de estado, todos sus dependientes son notificados.	-Cuando un cambio en un objeto requiere cambiar otros objetos -Para implementar sistemas de eventos -Cuando un objeto debe notificar a otros sin conocer quiénes son	
STRATEGY	Define una familia de algoritmos, encapsula cada uno y los hace intercambiables. Permite que el algoritmo varíe independientemente.	-Cuando tienes múltiples algoritmos para una tarea específica -Para evitar declaraciones condicionales complejas -Cuando diferentes variantes de un algoritmo necesitan ser intercambiables