POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

MiW

Patrones de Diseño
2. Catálogo

Luis Fernández Muñoz
https://www.linkedin.com/in/luisfernandezmunyoz
setillofm@gmail.com

INDICE

1. Según Propósito

- **Patrones Creacionales** abstraen el proceso de creación de instancias.
  - Ayudan a hacer a un sistema independiente de cómo sus objetos son creados, compuestos o representados.
  - Se hacen más importantes según un sistema evoluciona al depender más de la composición de objetos que de la herencia de clases.
    - Cuando esto sucede, se pasa de codificar una serie de comportamientos fijos a definir en un conjunto más pequeño de comportamientos fundamentales que pueden componerse en otros más complejos.
    - Así, para crear objetos con un determinado comportamiento es necesario algo más que simplemente crear una instancia de una clase.

1. Según Propósito

- **Patrones Creacionales**
  - Los patrones de creación muestran cómo hacer un diseño flexible, no necesariamente más pequeño. En concreto, harán que sea más fácil cambiar las clases que definen los "productos". En este caso, el principal obstáculo para el cambio reside en fijar en el código las clases de las que se crean las instancias. Los patrones de creación proporcionan varias formas de eliminar las referencias explicitas a clases concretas en el código que necesita crear instancias de ellas.

1. Según Propósito

- **Patrones Creacionales**
  - Hay dos temas recurrentes en estos patrones.
    - Todos ellos encapsulan el conocimiento sobre las clases concretas que usa el sistema.
    - Todos ocultan cómo se crean y se asocian las instancias de las clases concretas que usa el sistema.
    - Todo lo que el sistema como tal conoce acerca de los objetos son sus interfaces, tal y como las definen sus clases abstractas. Por tanto, los patrones de creación dan mucha flexibilidad a qué es lo que se crea, quién lo crea y cuándo. Permiten configurar un sistema con objetos "producto" que varían mucho en estructura y funcionalidad. La configuración que puede ser estática (esto es, especificada en tiempo de compilación) o dinámica (en tiempo de ejecución).

1. Patrones Creacionales

- ***Abstract Factory*** (*Fábrica Abstracta*)
- ***Builder*** (*Creador*)
- ***Factory Method*** (*Método Factoría*)
- ***Prototype*** (*Prototipo*)
- ***Singleton*** (*Único*)

1. Patrones Creacionales

- *Un cirujano se responsabiliza de las intervenciones siguiendo un algoritmo y, dependiendo del lugar (p.e. quirófano, selva, …), la realizará con unos instrumentos u otros (p.e. bisturí vs navaja, pulsómetro vs contando pulsaciones, …).*
- *Si la responsabilidad del cirujano incluye la adquisición del instrumental en cualquier lugar, ésta se complica y tiende a crecer más y más con nuevos lugares mezclando dos asuntos innecesariamente.*
- *Si la responsabilidad del cirujano incluye la solicitud de instrumental abstracto (algo cortante, contador de pulsaciones, …) y otros se responsabilizan por separado de cómo se obtienen los instrumentos dependiendo lugar ninguna explicación será compleja ni tendiendo a crecer*

- ***Abstract Factory.*** Proveer una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas.

1. Patrones Creacionales

- *Antiguamente, el hombre construía los productos que él mismo producía (prosumidor, sabía hacer de "todo": p.e. pan, pantalones, …) y el intercambio no era fácil (yo quiero lo que tu tienes, tu quieres lo que él tiene y el quiere lo que yo tengo).*
- *Todo era muy complicado ("nadie sabe hacer de todo") e inflexible en sus colaboraciones (sin libre comercio, exportación, importación, …)*
- *Desde que el hombre dejó de ser prosumidor, delega la construcción de sus productos de consumo (p.e. en tiendas, en empresas, …) de tal forma que el mismo proceso de construcción (p.e. ahora, ir de compras) puede "crear" diferentes productos (p.e. pan integral o pan de boutique, pantalones largos o cortos, …) de forma sencilla y flexible (p.e. nuevos productos en tiendas, …).*

- ***Builder.*** Separar la construcción de un objeto complejo de su representación de tal forma que el mismo proceso de construcción puede crear diferentes representaciones

1. Patrones Creacionales

- *La formación de un repartidor con toda su responsabilidad (recoger paquetes, estudiar rutas, entrega de paquetes, …) incluyendo el desplazamiento según los medios de transporte disponibles (p.e. bicicleta, moto, coche, …) se hace muy complicada y tiende a crecer según los nuevos disponibles .*
- *La formación de un Repartidor abstracto incluye toda su responsabilidad pero no se ciñe a un medio de transporte particular, lo menciona sin entrar en detalles. La formación específica de RepartidorEnBicicleta y RepartidorEnFurgoneta redefinen por separado la conducción dependiendo del medio de transporte concreto con el que se completará toda la formación del Repartidor abstracto*
- *Ninguna formación tiende a crecer y son más sencillas*

- ***Factory Method.*** Definir un interfaz para crear un objeto pero permitiendo a las subclases decidir que clase de instancia. El Método Factoría permite a una clase diferir la instanciación a las subclases.

1. Patrones Creacionales

- *Un poeta en un parque vende poemas famosos que escribe en el momento en que se lo piden porque se los sabe de memoria. Si cambian los gustos, tendrá que adaptarse y aprender nuevos poemas. Lo cual es complejo y tiende a crecer*
- *Otro poeta dispone de copias de poemas famosos y realiza fotocopias en el momento en que se lo piden sin sabérselos de memoria. Si cambian los gustos, tendrá que conseguir una copia del nuevo poema, incluso en otro idioma. Lo cual es muy sencillo y no tiende a crecer.*

- ***Prototype.*** Especificar las clases de objetos a crear usando una instancia prototípica y crear nuevos objetos por copia de estos prototipos

1. Patrones Creacionales

- *Si en una organización que disfruta de un personal de seguridad no ofrece un número de teléfono o cualquier mecanismo global, éste debería presentarse, si es nuevo en la organización, y avisar constantemente de su ruta para que cualquier pudiera dar a aviso por alguna incontingencia.*
- *Es el número de teléfono o mecanismo global el que permite la facilidad de localizarlo sin constantes avisos, incluso cambiando de personal sin afectar a nadie.*

- ***Singleton.*** Asegura que una clase tiene un sola instancia y provee un punto de acceso global a ésta.

1. Según Propósito

- **Patrones Estructurales** para tratar cómo se combinan las clases y los objetos para formar estructuras más grandes.
  - Existen similitudes entre los patrones estructurales, especialmente en sus participantes y colaboradores. Esto es debido a que los patrones estructurales se basan en un mismo pequeño conjunto de mecanismos del lenguaje para estructurar el código y los objetos:
    - herencia simple y herencia múltiple para los patrones basados en clases, y
    - composición de objetos (delegación) para los patrones de objetos.
  - Pero estas similitudes ocultan los diferentes propósitos de estos patrones.

2. Patrones Estructurales

- ***Adapter*** (*Adaptador*)
- ***Bridge*** (*Puente*)
- ***Composite*** (*Compuesto*)
- ***Decorator*** (*Decorador*)
- ***Facade*** (*Fachada*)
- ***Flyweight*** (*Peso Ligero*)
- ***Proxy*** (*Representante*)

2. Patrones Estructurales

- *Cuando se viaja al extranjero, existen problemas para enchufar los aparatos eléctricos (p.e. secador del pelo, maquinilla de afeitar, …) porque el numero, forma y/o disposición de las patillas de los enchufes no son iguales en todos los países, hay varios estándares.*
- *Un solución costosa y engorrosa sería comprar nada más llegar al destino todos los aparatos eléctricos en el país de destino y, a la vuelta, guardarlos en el trastero por si se vuelve a viajar a un país del mismo estándar.*
- *Otra solución barata y sencilla es comprar adaptadores de enchufes antes de viajar y guardarlos a mano para el próximo viaje a un país con el mismo estándar.*

- ***Adapter.*** Convertir la interfaz de una clase en otra interfaz que esperan los clientes. Los Adaptadores permiten a las clases trabajar juntas que de otra forma no podrían por sus interfaces incompatibles

2. Patrones Estructurales

- *Cierta persona debe ser responsable de interpretar ciertos datos que guarda en cierto soporte (p.e. en una libreta o en una hoja de cálculo o de memoria o …). Esta persona debe saber escribir, utilizar el ordenador, técnicas de memorización, … lo cual es complejo, tiende a crecer (p.e. curricula vitae difíciles de encontrar) e inflexible (p.e. formación continua por nuevos soportes)*
- *Una alternativa sería que esta persona sólo sea responsable de interpretar dichos datos y que otras personas por separado sean responsables de guardar en diversos soportes específicos dependiendo de su especialidad. Así, cada uno está especializado en un única responsabilidad, lo cual no es complejo ni tiende a crecer y es flexible escogiendo una nueva pareja para el nuevo soporte*

- ***Bridge.*** Desacoplar una abstracción de su implementación de tal forma que los dos pueden variar independientemente

2. Patrones Estructurales

- *En muchos juegos de mesa (p.e. trivial pursuit, parchis, …), hay limitaciones en el tablero para el número máximo de jugadores. Cuando hay más personas que este máximo existe la posibilidad de formar equipo para que jueguen todos a la vez.*
- *Pero si no se buscan las fórmulas adecuadas puede ser un caos cuando el turno de un "jugador/equipo" lo realizan varias personas del equipo a la vez.*
- *La solución más habitual es nombrar un representante del equipo y, de cara al resto de jugadores, es con el único que interactúan. Internamente el representante se organizará transparentemente con el resto de personas del equipo para tomar decisiones.*

- ***Composite.*** Componer objetos en estructuras arbóreas para representar jerarquías todo-parte. Los Compuestos permiten a los clientes tratar uniformemente objetos individuales y objetos compuestos

2. Patrones Estructurales

- *Un aprendiz de un oficio puede llevar a cabo ciertas tareas pero sobre aspectos muy básicos seleccionadas/encargadas por el gran artesano. Puntualmente, para alguna tarea más delicada, el artesano explica la tarea a un oficial para que supervise según la está realizando el aprendiz sobre aspectos más avanzados.*
- *De esta manera, se consigue añadir nuevas responsabilidades al aprendiz (nuevos aspectos más avanzados) con la intermediación/supervisión del oficial.*

- ***Decorator.*** Añadir responsabilidades adicionales a un objeto dinámicamente. Los Decoradores proveen una alternativa flexible a la sub-clasificación para extender la funcionalidad

2. Patrones Estructurales

- *Una obra de reforma de un domicilio conlleva que el propietario realice diversas tareas (p.e. solicite licencias, crétditos, …) y, en particular, contrate a diversos especialistas (p.e. albañil, pintor, electricista, decorador, …) coordinando a todos (p.e. tiempos, retrasos, materiales, pre-condiciones, …). Lo cual es muy complejo y tiende a crecer con nuevos "materiales" (p.e. expertos en domótica, …)*
- *Una alternativa más sencilla y que no tiende a crecer es que el propietario se centre en sus tareas y contrate a un contratista para no interactuar con el resto de especialistas. Se reparten las responsabilidades*

- ***Facade.*** Proveer una interfaz unificada para un conjunto de interfaces en un subsistema. Define una interfaz de alto nivel que facilita el uso del subsistema.

2. Patrones Estructurales

- *Los libros de una biblioteca (supuestamente ejemplares únicos) son compartidos eficazmente por infinitos usuarios para leer, citar, resumir, …*
- *De esta manera se evitan los costes de que todos los usuarios tengan que comprar un libro para leerlo. Compartir ahorra.*

- ***Flyweight.*** Usa compartir para soportar eficientemente una gran número de objetos de grano fino

2. Patrones Estructurales

- *Cuando un jefe necesita abandonar la oficina se desatienden diversos aspectos en la toma de decisiones (p.e. reacción ante cierta contingencia, concesión de vacaciones, …) que requieren los empleados en su trabajo diario. Esto provoca una dependencia en la localización del jefe para que todo siga su curso.*
- *Una solución alternativa que independiza la localización del jefe es nombrar a un representante en su ausencia para que centralice la atención de las demandas de los empleados, las notifique al jefe (p.e. por teléfono o correo o …) y responda con las decisiones de éste sin tener que delegar en nadie.*
- *Puntualmente, se puede añadir a la responsabilidad del representante algunas acciones más allá de la simple notificación (p.e. filtrar ciertas demandas postergables, …)*

- ***Proxy.*** Proveer un sustituto o marcador de posición de un objeto para controlar el acceso a éste

1. Según Propósito

- **Patrones de Comportamiento** tienen que ver con algoritmos y con la asignación de responsabilidades a objetos.
  - Los patrones de comportamiento describen la comunicación entre clases y objetos.
  - Estos patrones describen el flujo de control complejo que es difícil de seguir en tiempo de ejecución, lo que nos permite olvidarnos del flujo de control para concentrarnos simplemente en el modo en que se interconectan los objetos
  - Una cuestión importante es cómo los objetos saben unos de otros. Cada uno podría mantener referencias explícitas al resto, pero eso incrementaría su acoplamiento. Llevado al límite, cada objeto conocería a todos los demás.
  - Otros patrones de comportamiento basados en objetos están relacionados con la encapsulación de comportamiento en un objeto, delegando las peticiones a dicho objeto.

1. Según Propósito

- **Patrones de Comportamiento**
  - Varios patrones describen aspectos de un programa que es probable que cambien. La mayoría de los patrones tienen dos tipos de objetos: el nuevo objeto que encapsula el aspecto y el objeto existente que usa el nuevo objeto creado. Normalmente, si no fuera por el patrón, la funcionalidad de los nuevos objetos sería una parte integral de los existentes.
  - Objetos como argumentos. Varios patrones introducen un objeto que siempre se usa como argumento. Otros patrones definen objetos que actúan como elementos mágicos que se pasan de un lado a otro y que más tarde pueden ser invocados.

1. Según Propósito

- **Patrones de Comportamiento**
  - Desacoplar emisores y receptores. Cuando los objetos que colaboran se refieren unos a otros explícitamente, se vuelven dependientes unos de otros, y eso puede tener un impacto adverso sobre la división en capas y la reutilización de un sistema.
  - Encapsular lo que varía. Encapsular aquello que puede variar es el tema de muchos patrones de comportamiento. Cuando un determinado aspecto de un programa cambia con frecuencia, estos patrones definen un objeto que encapsula dicho aspecto. De esa manera, otras partes del programa pueden colaborar con el objeto siempre que dependan de ese aspecto. Los patrones normalmente definen una clase abstracta que describe el objeto encapsulado, y el patrón toma su nombre de ese objeto.

3. Patrones de Comportamiento

- ***Chain of Responsability*** (*Cadena de Responsabilidad*)
- ***Command*** (*Comando*)
- ***Intepreter*** (*Intérprete*)
- ***Iterator*** (*Iterador*)
- ***Mediator*** (*Mediador*)
- ***Memento*** (*Recuerdo*)
- ***Observer*** (*Observador*)
- ***State*** (*Estado*)
- ***Strategy*** (*Estrategia*)
- ***Template Method*** (*Método Plantilla*)
- ***Visitor*** (*Visitante*)

3. Patrones de Comportamiento

- *Un profesor está en contacto directo con los alumnos pero no puede tomar decisiones ante todas las demandas posibles de éstos (p.e. cambio de plan de estudios, denunciar actitudes fraudulentas del propio profesor, …). Pero tampoco es deseable que el alumno tenga que conocer y discernir qué persona de la cadena de mando es la apropiada para cada demanda (p.e. coordinador de la asignatura, jefe de departamento, director de escuela, vicerrector, …)*
- *Una solución mas sencilla para el alumno es que cuando tiene una demanda se la comunica al profesor de la asignatura, si no puede/sabe resolverla, éste la eleva al coordinador de la asignatura, si no puede/sabe resolverla, éste la eleva al director del departamento, …*

- ***Chain of Responsibility.*** Evitar el acoplamiento del emisor de una petición a su receptor, dando a más de un objeto la oportunidad de manejar la petición. Encadenar los objetos que reciben la petición y pasar la petición a lo largo de la cadena hasta un objeto que la maneje.

3. Patrones de Comportamiento

- *Una posible organización en un restaurante sería que cuando un camarero anota una comanda de un cliente, se dirija a la cocina, busque/encargue a un cocinero adecuado (p.e. sabe cocinar el plato, dispone de tiempo, de fogones, …) y le consulta si está disponible cada plato para servir al cliente. O sea, una caos potencial donde todo camarero interactúa con todo cocinero.*
- *Una alternativa es que los camareros dejen la comanda en un lugar acordado al que acuden los cocineros para escoger la siguiente según sus condiciones y, tras la preparación, dejan los platos para servir a los clientes. Con dos equipos totalmente independientes y mucho menos trabajo para el cocinero.*

- ***Command.*** Encapsular una petición como un objeto, permitiendo de ese modo la parametrización de clientes con diferentes peticiones, en cola o en registro, y apoyar las operaciones que se pueden deshacer

3. Patrones de Comportamiento

- *Demandar a un anciano que sepa manejar un aparato moderno (p.e. móvil, televisión digital, …) puede ser una tarea muy compleja por su propia naturaleza y contexto (millones de millones de conexiones físicas de neuronas a deshacer y rehacer).*
- *Una solución que alivia toda la complejidad es que cuando el anciano quiere manipular algún aparato moderno, llama a su nieto para que "interprete" sus deseos: el anciano dicta (p.e. quiero hablar con Pepe, quiero ver el partido, …) y el nieto opera (p.e. marca el teléfono, maneja los 3 mandos a distancia, …).*

- ***Interpreter.*** Dado un lenguaje, definir una representación de su gramática junto con un intérprete que utiliza la representación para interpretar sentencias del lenguaje.

3. Patrones de Comportamiento

- *En un centro comercial con todos los productos expuestos en los estantes sin dependientes en el local, son los clientes los que tienen que conocer/saber encontrar la localización de los productos. Por ese motivo, cuando cambian la localización de los productos (p.e. obras, campaña de navidad, …) los usuarios no encuentran los productos y quedan afectados.*
- *En un comercio que no sea autoservicio, el dependiente accede "secuencialmente" a los productos de un almacén sin exponer sus estanterías o estructuras de almacenamiento. De esta manera, no se afecta al cliente cuando hay algún cambio de localización de los productos en el almacén, solo afecta al dependiente.*

- ***Iterator.*** Proveer una forma de acceso secuencial a los elementos de un objeto agregado sin exponer su representación subyacente

3. Patrones de Comportamiento

- *En un partido de futbol de una liga profesional hay mucho más que dos equipos de jugadores y un árbitro; se contempla el público, los cuerpos de seguridad, el delegado de campo, …*
- *Para evitar que todos estos agentes interactúen con todos los otros (p.e. un policía ordenando retirarse a un defensa por seguridad, un delegado de campo solicitando una prorroga para el comienzo a todos los jugadores, …) en un gran caos de sincronización, se establece la figura del árbitro al que todos acuden y éste actúa sobre todos según considere oportuno.*

- ***Mediator.*** Definir un objeto que encapsula cómo un conjunto de objetos interactúan. El Mediador promueve bajo acoplamiento evitando que los objetos se refieran explícitamente a cada uno de los otros y permitir variar sus interacciones independientemente.

3. Patrones de Comportamiento

- *En la elaboración de un documento, cada vez que se guarda se pierde la posibilidad de "volver" a una versión anterior del documento.*
- *Por ese motivo, muchas personas recurren al "guardar como" para almacenar sucesivas versiones para tener un histórico de los distintos estados de la elaboración del documento y poder "volver" a cualquier estado anterior*

- ***Memento.*** Sin violar la encapsulación, captura y externaliza el estado interno de un objeto de tal forma que el objeto puede ser restaurado a sus estados anteriores.

3. Patrones de Comportamiento

- *Cuando en una sucursal bancaria se produce un atraco hay que avisar de inmediato a los cuerpos de seguridad públicos y privados particulares del banco para reforzar a los primeros.*
- *Una alternativa es dotar a los empleados de los mecanismos particulares para avisar a cada cuerpo de seguridad (p.e. números de teléfono o páginas web o …). Lo cual obliga a gestionar esos mecanismos cada vez que haya un cambio (p.e. nuevos cuerpos, …)*
- *Una alternativa más eficiente sería pulsar un botón que dispara automáticamente el aviso a todos los cuerpos registrados en el automatismo. Esto independiza a los empleados de intervenir en la gestión de esos mecanismos, solo pulsa y ya se avisará quien esté registrado*

- ***Observer.*** Definir una dependencia uno-a-muchos entre objetos de tal forma que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizados automáticamente

3. Patrones de Comportamiento

- *La explicación del comportamiento de una persona depende de su estado de ánimo (p.e. cansado, bloqueado, proactivo, …) en cada una de las tareas a comprender (p.e. dormir, estudiar, trabajar, …). Esto complica la comprensión de su comportamiento. Según surgen nuevos estados de ánimo (p.e. enamorado, enfermo, …) se complica cada vez más la comprensión de cada tarea en cada estado.*
- *Una alternativa más sencilla sería la explicación por separado de todas las tareas para cada estado de ánimo particular. Con el advenimiento de nuevos estados no se complica ninguna explicación para estados anteriores o futuros.*

- ***State.*** Permitir a un objeto alterar su comportamiento cuando cambia su estado interno. El objeto parecerá que cambió de clase

3. Patrones de Comportamiento

- *Cuando alguien comienza a aprender a ligar establece a fuego una técnica de acercamiento inicial (p.e. forzar casualidad, afición compartida, persona en común, te invito a algo, …) que considera que es su mejor opción (p.e. donde se siente más seguro, donde cree que cosechará más existos, …). Pero normalmente no está capacitado para improvisar con nuevas técnicas.*
- *La diferencia con los expertos en ligar es que no establecen una política rígida sino que pueden escoger entre una u otra dependiendo del contexto o incluso incorporar a lo largo de su vida nuevas técnicas que no conocía. Esto les da la flexibilidad para lograr una tasa de éxito mayor independientemente del contexto y del tiempo.*

- ***Strategy.*** Definir una familia de algoritmos, encapsular cada uno y hacerlos intercambiables. La Estrategia permite variar independientemente el algoritmo desde los clientes que lo usan

3. Patrones de Comportamiento

- *La responsabilidad de un profesor se concreta en un algoritmo que contempla la preparación de asignaturas, su impartición y evaluación. Para ello dispone de diversos recursos didácticos (p.e. libros, pizarras, fotocopias, …) disponibles según el contexto (p.e. profesor del MIT o voluntariado en una selva perdida o …). Lo cual provoca que la formación de un profesor sea compleja y tienda a crecer con nuevos recursos didácticos (p.e. juegos didácticos ágiles, internet, …).*
- *Una alternativa es separar la formación de un profesor genérico con el algoritmo de sus tareas sin ceñirse a unos recursos didácticos particulares y, por otro lado, distintas formaciones especializadas en ciertos recursos didácticos. Se simplifica la responsabilidad de un profesor particular sin afectarle con nuevos recursos que no utilizará.*

- ***Template Method.*** Definir el esqueleto de un algoritmo en una operación defiriendo algunos pasos a las subclases. El Método Plantilla permite a las subclases redefinir ciertos pasos de un algoritmo sin cambiar la estructura del algoritmo.

3. Patrones de Comportamiento

- *Una posible organización dentro de un hospital sería que un mismo empleado fuera responsable de todas las tareas a realizar (p.e. transportar, curar heridas, diagnosticar, intervención quirurjica, …) con los órganos de un paciente (p.e. piernas, bazo, menisco, …). Esto complicaría muchísimo la responsabilidad de dichos empleados que saben de todo y tendiendo a crecer con nuevas técnicas en cualquier area.*
- *Una alternativa sencilla y sin tendencia desmesurada a crecer es que cada tarea la realicen personas especializadas diferentes (p.e. celador, enfermero, médico, cirujano, …) visitando en el momento oportuno los órganos del paciente.*

- ***Visitor.*** Representar un operación para ser realizada sobre los elementos de una estructura de objetos. Los Visitadores permiten definir una nueva operación sin cambiar las clases de los elementos sobre los que opera

2. Según Ámbito

- **Patrones de Clases** se ocupan de las relaciones entre clases y sus subclases. Estas relaciones se establecen a través de la herencia, de modo que son relaciones estáticas – fijadas en tiempo de compilación.
  - Casi todos los patrones usan la herencia de un modo u otro, así que los únicos patrones etiquetados como patrones de clases son aquellos que se centran en las relaciones de herencia entre clases.
- **Patrones de Objetos** tratan con las relaciones entre objetos, que pueden cambiarse en tiempo de ejecución y son más dinámicas.
  - La mayoría de los patrones tienen un ámbito de objeto.

2. Según Ámbito

- Patrones creacionales:
  - Un patrón creacional de **clases** usa la herencia para variar la clase de la instancia a crear.
  - Un patrón creacional de **objetos** delegará la instanciación a otro objeto.
- Patrones de comportamiento:
  - Los **patrones de comportamiento de clases** usan la herencia para distribuir el comportamiento entre clases.
  - Los **patrones de comportamiento de objetos** usan la composición de objetos en vez de la herencia. Algunos describen cómo cooperan un grupo de objetos parejos para realizar una tarea que ningún objeto individual puede llevar a cabo por sí solo.

2. Según Ámbito

- Patrones estructurales:
  - Los **patrones estructurales de clases** hacen uso de la herencia para componer interfaces o implementaciones. A modo de ejemplo sencillo, pensemos en cómo la herencia múltiple mezcla dos o más clases en una solo. El resultado es una clase que combina las propiedades de sus clases padre. Este patrón es particularmente útil para lograr que funcionen juntas bibliotecas de clases desarrolladas de forma independiente.
  - En vez de combinar interfaces o implementaciones, **los patrones estructurales de objetos** describen la forma de componer objetos para obtener nueva funcionalidad. La flexibilidad añadida de la composición de objeto viene dada por la capacidad de cambiar la composición en tiempo de ejecución, lo que es imposible con la composición de clases estática.

2. Según Ámbito

		Propósito		
		Creacional	**Estructural**	**De comportamiento**
Ámbito	**Clase**	*Factory Method*	*Adapter*	*Interpreter* *Template Method*
	Objeto	*Abstract Factory* *Builder* *Prototype* *Singleton*	*Adapter* *Bridge* *Composite* *Decorator* *Facade* *Flyweight* *Proxy*	*Chain of Responsability* *Command* *Iterator* *Mediator* *Memento* *Observer* *State* *Strategy* *Visitor*