Patrones de diseño

En ingeniería de software, un **patrón de diseño** es una solución repetible general para un problema común en el diseño de software. Un patrón de diseño no es un diseño terminado que pueda transformarse directamente en código. Es una descripción o plantilla sobre cómo resolver un problema que se puede utilizar en muchas situaciones diferentes.

Usos de patrones de diseño

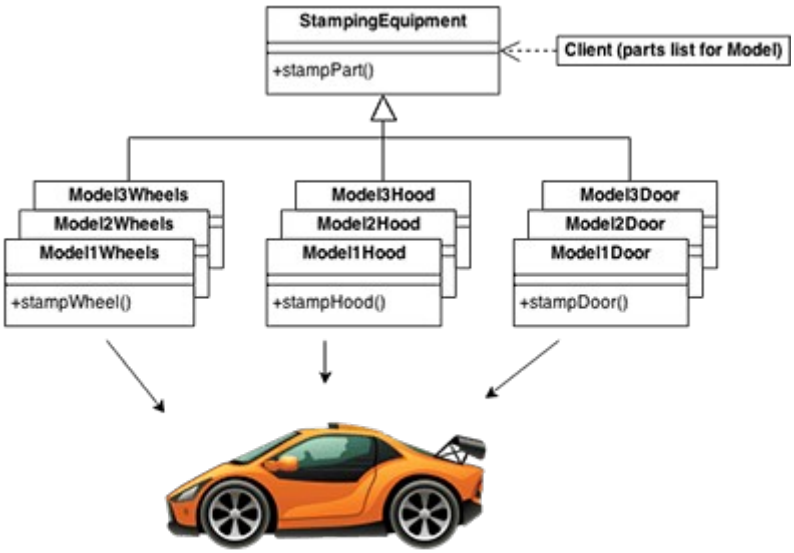
Los patrones de diseño pueden acelerar el proceso de desarrollo al proporcionar paradigmas de desarrollo probados y probados. El diseño efectivo del software requiere considerar problemas que pueden no ser visibles hasta más adelante en la implementación. La reutilización de patrones de diseño ayuda a evitar problemas sutiles que pueden causar problemas importantes y mejora la legibilidad del código para codificadores y arquitectos familiarizados con los patrones.

A menudo, las personas solo entienden cómo aplicar ciertas técnicas de diseño de software a ciertos problemas. Estas técnicas son difíciles de aplicar a una gama más amplia de problemas. Los patrones de diseño proporcionan soluciones generales, documentadas en un formato que no requiere detalles vinculados a un problema en particular.

Además, los patrones permiten a los desarrolladores comunicarse utilizando nombres conocidos y bien entendidos para las interacciones de software. Los patrones de diseño comunes se pueden mejorar con el tiempo, haciéndolos más robustos que los diseños ad-hoc.

Patrones de diseño creacional

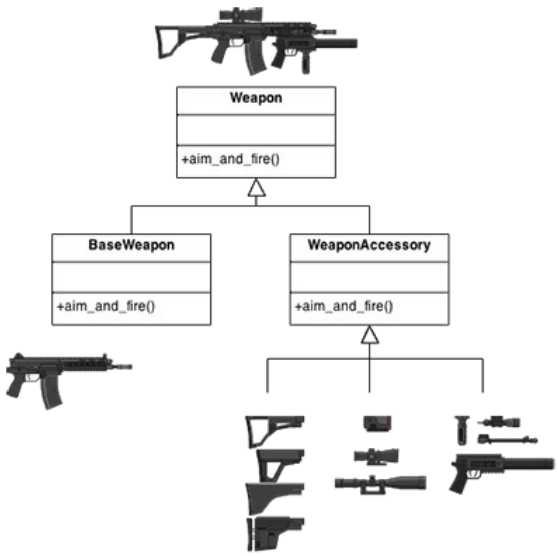
Estos patrones de diseño tienen que ver con la creación de instancias de clase. Este patrón se puede dividir en patrones de creación de clases y patrones de creación de objetos. Mientras que los patrones de creación de clases usan la herencia de manera efectiva en el proceso de creación de instancias, los patrones de creación de objetos usan la delegación de manera efectiva para hacer el trabajo.



- Fábrica abstracta
  Crea una instancia de varias familias de clases.
- Constructor
  Separa la construcción de objetos de su representación.
- Método de la fábrica
  Crea una instancia de varias clases derivadas
- Pool de objetos
  Evite la adquisición costosa y la liberación de recursos reciclando objetos que ya no están en uso
- Prototipo
  Una instancia completamente inicializada para ser copiada o clonada
- único
  Una clase de la que solo puede existir una única instancia

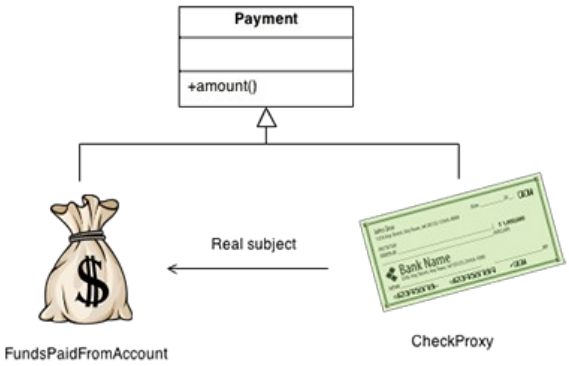
Patrones de diseño estructural.

Estos patrones de diseño tienen que ver con la composición de clases y objetos. Los patrones estructurales de creación de clases utilizan la herencia para componer interfaces. Los patrones de objetos estructurales definen formas de componer objetos para obtener una nueva funcionalidad.



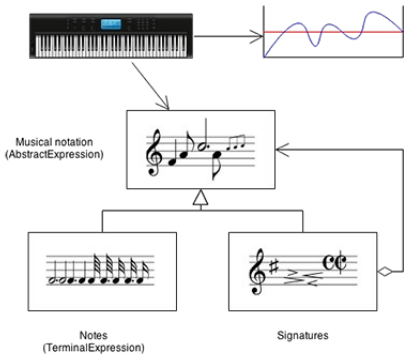
- Adaptador
  Emparejar interfaces de diferentes clases
- Puente
  Separa la interfaz de un objeto de su implementación
- Compuesto
  Una estructura de árbol de objetos simples y compuestos.
- Decorador
  Agregar responsabilidades a los objetos dinámicamente
- Fachada
  Una sola clase que representa un subsistema completo

- Peso mosca
  Una instancia detallada utilizada para compartir de manera eficiente
- Datos de clase privada
  Restringe el acceso de acceso / mutador
- Apoderado
  Un objeto que representa otro objeto.



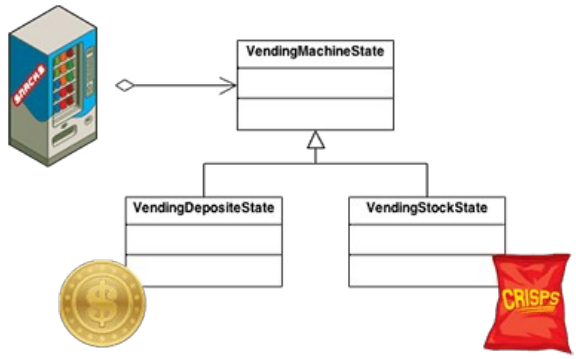
Patrones de diseño conductual

Estos patrones de diseño tienen que ver con la comunicación de objetos de Class. Los patrones de comportamiento son aquellos que están más específicamente relacionados con la comunicación entre objetos.



- Cadena de responsabilidad
  Una forma de pasar una solicitud entre una cadena de objetos
- Mando
  Encapsular una solicitud de comando como un objeto
- Interprete
  Una forma de incluir elementos del lenguaje en un programa
- Iterador
  Acceder secuencialmente a los elementos de una colección.
- Mediador
  Define la comunicación simplificada entre clases.
- Recuerdo
  Capture y restaure el estado interno de un objeto.

- Objeto nulo
  Diseñado para actuar como un valor predeterminado de un objeto
- Observador
  Una forma de notificar el cambio a varias clases
- Estado
  Alterar el comportamiento de un objeto cuando su estado cambia
- Estrategia
  Encapsula un algoritmo dentro de una clase
- Método de plantilla
  Diferir los pasos exactos de un algoritmo a una subclase
- Visitante
  Define una nueva operación para una clase sin cambio



Se dirige al problema equivocado

La necesidad de patrones resulta del uso de lenguajes informáticos o técnicas con capacidad de abstracción insuficiente. Bajo la factorización ideal, un concepto no debe copiarse, sino simplemente referenciarse. Pero si se hace referencia a algo en lugar de copiarse, entonces no hay un "patrón" para etiquetar y catalogar. Paul Graham escribe en el ensayo La venganza de los nerds .

Peter Norvig ofrece un argumento similar. Demuestra que 16 de los 23 patrones en el libro Design Patterns (que se centra principalmente en C ++) se simplifican o eliminan (a través del soporte directo del lenguaje) en Lisp o Dylan.

Carece de fundamentos formales

El estudio de los patrones de diseño ha sido excesivamente ad hoc, y algunos han argumentado que el concepto necesita una base más formal. En , la Banda de los Cuatro fue (con su plena cooperación) sometida a un juicio en el que fueron "acusados" de numerosos delitos contra la informática. Fueron "condenados" por ⅔ de los "jurados" que asistieron al juicio.

Conduce a soluciones ineficientes

La idea de un patrón de diseño es un intento de estandarizar las mejores prácticas ya aceptadas. En principio, esto puede parecer beneficioso, pero en la práctica a menudo resulta en una duplicación

innecesaria de código. Casi siempre es una solución más eficiente usar una implementación bien factorizada en lugar de un patrón de diseño "apenas lo suficientemente bueno".

No difiere significativamente de otras abstracciones

Algunos autores alegan que los patrones de diseño no difieren significativamente de otras formas de abstracción, y que el uso de nueva terminología (prestada de la comunidad de arquitectura) para describir los fenómenos existentes en el campo de la programación es innecesaria. El paradigma Modelo-Vista-Controlador se promociona como un ejemplo de un "patrón" que precede al concepto de "patrones de diseño" por varios años. Algunos sostienen que la contribución principal de la comunidad de Patrones de diseño (y el libro Gang of Four) fue el uso del lenguaje de patrones de Alexander como una forma de documentación; Una práctica que a menudo se ignora en la literatura.