

DISEÑO Y ARQUITECTURA DE SOFTWARE

PATRÓN DE DISEÑO DECORATOR

JESÚS ARMANDO GARCIA MUÑOZ



NOMBRE Y CLASIFICACIÓN

DECORATOR PATTERN



- SE CLASIFICA DENTRO DE LOS PATRONES DE DISEÑO ESTRUCTURAL.
- AGREGA RESPONSABILIDADES ADICIONALES A UN OBJETO DE MANERA DINÁMICA.
- PROPORCIONANDO UNA ALTERNATIVA FLEXIBLE A LA HERENCIA PARA EXTENDER FUNCIONALIDAD.
- PROGRAMACIÓN MODULAR

INTENCIÓN



- PROPORCIONA UNA FORMA FLEXIBLE DE INTRODUCIR O ELIMINAR FUNCIONALIDAD DE UN COMPONENTE SIN MODIFICAR SU APARIENCIA EXTERNA O SU FUNCIÓN.

OTROS NOMBRES



- **WRAPPER PATTERN** (OBJETOS LOS CUALES VAMOS A IR ENCAPSULANDO PARA AGREGAR FUNCIONALIDAD)
- HERENCIA PERO SIN SER HERENCIA COMO TAL NO ES HERENCIA TRADICIONAL SI NO IMPLEMENTADA DE CIERTO MODO

MOTIVACIÓN

- A VECES SE DESEA ADICIONAR RESPONSABILIDADES A UN OBJETO PERO NO A TODA LA CLASE.
- LAS RESPONSABILIDADES SE PUEDEN ADICIONAR POR MEDIO DE LOS MECANISMOS DE HERENCIA, PERO ESTE MECANISMO NO ES FLEXIBLE PORQUE LA RESPONSABILIDAD ES ADICIONADA ESTÁTICAMENTE.
- LA SOLUCIÓN FLEXIBLE ES LA DE RODEAR EL OBJETO CON OTRO OBJETO QUE ES EL QUE ADICIONA LA NUEVA RESPONSABILIDAD. ESTE NUEVO OBJETO ES EL DECORADOR

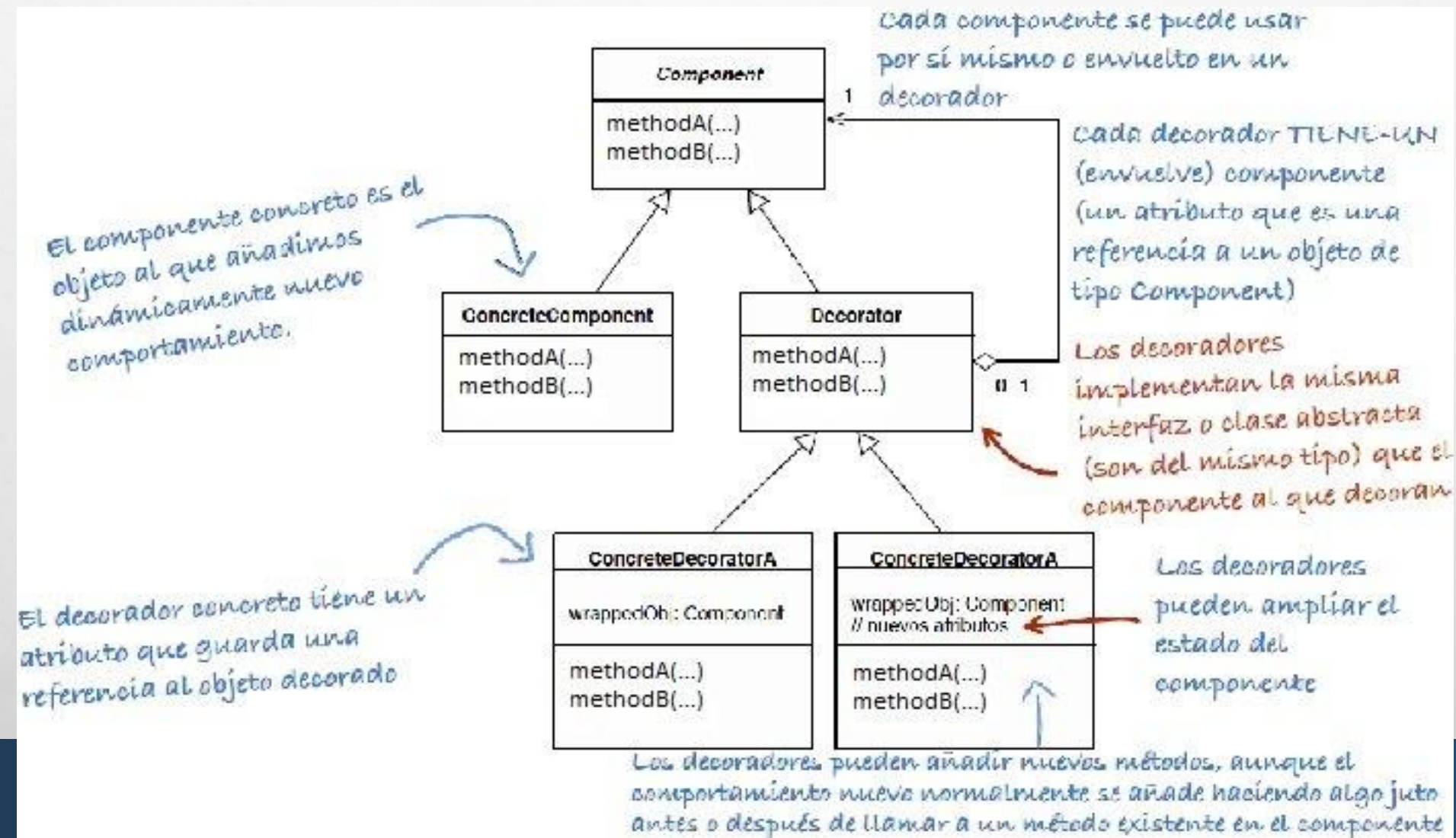
APLICACIÓN



SE DEBE USAR EL DECORADOR CUANDO:

- SE QUIERA AÑADIR RESPONSABILIDADES A OTROS OBJETOS DINÁMICAMENTE Y DE FORMA TRANSPARENTES.
- NO SE PUEDE HEREDAR O NO RESULTE PRACTICO.
- HAY UNA NECESIDAD DE EXTENDER LA FUNCIONALIDAD DE UNA CLASE.
- AÑADIR NUEVAS CARACTERÍSTICAS POR MEDIO DE EXTENSIÓN DE CLASE

ESTRUCTURA



PARTICIPANTES

- **COMPONENT:** DEFINE LA INTERFACE DE LOS OBJETOS A LOS QUE SE LE PUEDE ADICIONAR RESPONSABILIDADES DINÁMICAMENTE.
- **CONCRETECOMPONENT:** DEFINE EL OBJETO AL QUE SE LE PUEDE ADICIONAR UNA RESPONSABILIDAD.
- **DECORATOR:** MANTIENE UNA REFERENCIA AL OBJETO COMPONENT Y DEFINE UNA INTERFACE DE ACUERDO CON LA INTERFACE DE COMPONENT.
- **CONCRETEDECORATOR:** ADICIONA LA RESPONSABILIDAD AL COMPONENTE.

COLABORACIONES



- EL DECORADOR REDIRIGE LAS PETICIONES AL COMPONENTE ASOCIADO
- OPCIONALMENTE PUEDEN REALIZAR TAREAS ADICIONALES ANTES Y DESPUÉS DE REDIRIGIR LA PETICIÓN.

CONSECUENCIAS



- MÁS FLEXIBLE QUE LA HERENCIA. AL UTILIZAR ESTE PATRÓN, SE PUEDEN AÑADIR Y ELIMINAR RESPONSABILIDADES EN TIEMPO DE EJECUCIÓN. A
- EVITA LA UTILIZACIÓN DE LA HERENCIA CON MUCHAS CLASES Y TAMBIÉN, EN ALGUNOS CASOS, LA HERENCIA MÚLTIPLE.
- EVITA LA APARICIÓN DE CLASES CON MUCHAS RESPONSABILIDADES EN LAS CLASES SUPERIORES DE LA JERARQUÍA.
- GENERA GRAN CANTIDAD DE OBJETOS PEQUEÑOS
- PUEDE HABER PROBLEMAS CON LA IDENTIDAD DE LOS OBJETOS. UN DECORADOR SE COMPORTA COMO UN ENVOLTORIO TRANSPARENTE.
- PERO DESDE EL PUNTO DE VISTA DE LA IDENTIDAD DE OBJETOS, ESTOS NO SON IDÉNTICOS, POR LO TANTO NO DEBERÍAMOS APOYARNOS EN LA IDENTIDAD CUANDO ESTAMOS USANDO DECORADORES.

IMPLEMENTACIÓN



- EL PATRÓN DECORATOR SOLUCIONA ESTE PROBLEMA DE MANERA MUCHO MAS SENCILLA Y EXTENSIBLE.
- SE CREA A PARTIR DE *VENTANA* LA SUBCLASE ABSTRACTA ***VENTANADECORATOR*** Y, HEREDANDO DE ELLA, ***BORDEDECORATOR*** Y ***BOTONDEAYUDADECORATOR***.
- ***VENTANADECORATOR*** ENCAPSULA EL COMPORTAMIENTO DE *VENTANA* Y UTILIZA COMPOSICIÓN RECURSIVA PARA QUE SEA POSIBLE AÑADIR TANTAS "CAPAS" DE **DECORATORS** COMO SE DESEE.
- PODEMOS CREAR TANTOS **DECORATORS** COMO QUERAMOS HEREDANDO DE ***VENTANADECORATOR***.

EJEMPLOS DE CÓDIGO



USOS CONOCIDOS

- SE EMPLEA CON FRECUENCIA PARA DAR FUNCIONALIDADES A LOS TOOLKITS.



PATRONES RELACIONADOS



- **STRATEGY:** DECORADOR MODIFICA LA PIEL DEL OBJETO, MIENTRAS QUE ESTRATEGIA MODIFICA EL FUNCIONAMIENTO INTERNO.
- **ADAPTER:** SE DIFERENCIAN EN QUE ADAPTADOR MODIFICA LA INTERFAZ DEL OBJETO MIENTAS QUE EL DECORADOR NO LO REALIZA (AMBOS ENVUELVEN).

GRACIAS POR SU ATENCIÓN

