

Práctica de Laboratorio No. 6 Patrón Decorador

Harold Andrés Velásquez Tobar

Carlos Mario Vivas García

Mgtr. Ricardo Zambrano



Universidad
del Cauca

Ingeniería de Sistemas

18/10/2023

Universidad del Cauca

Popayán – Cauca

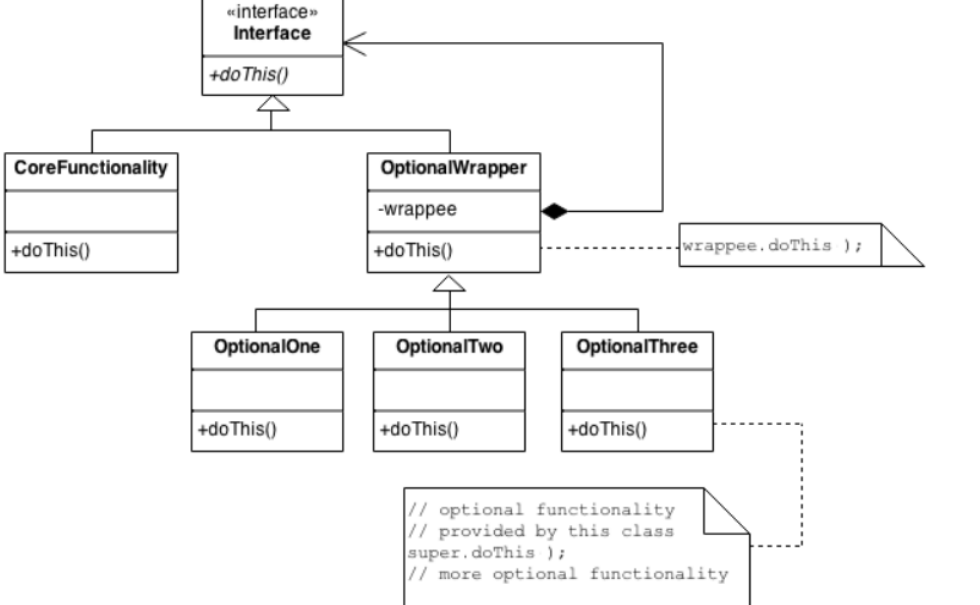
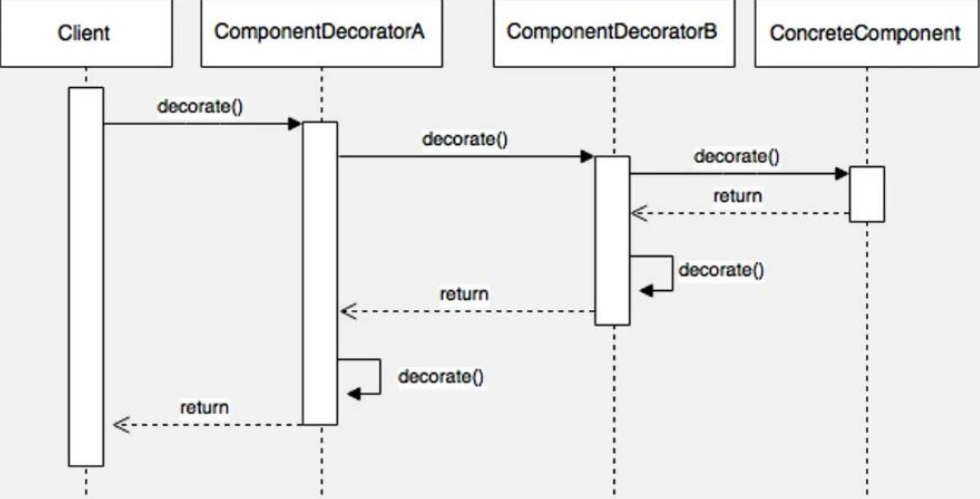
Tabla de Contenido

Plantilla Patrón Decorador	3
Implementación del Patrón Decorador.....	5
Referencias	8

Plantilla Patrón Decorador

1. Desarrolle la siguiente plantilla para el patrón Decorador:

Patrón Creacional: Decorador	
Intención	La finalidad del patrón Decorador se desdobra en dos aspectos. En primer lugar, pretende permitir la inclusión de nuevas responsabilidades o funcionalidades en objetos preexistentes sin necesidad de alterar su código original. Esto se revela como un elemento fundamental para mantener la regla de "abierta a extensiones, cerrada a modificaciones". En segundo lugar, se propone lograr este propósito de manera adaptable, de modo que se puedan combinar y superponer múltiples decoradores para la creación de disposiciones personalizadas de objetos.
Problema que Soluciona	<p>El patrón Decorador afronta varios dilemas comunes en la concepción del software:</p> <ul style="list-style-type: none"> - Ampliación sin recurrir a la herencia: La herencia constituye un método para agregar atribuciones a las clases, aunque suele desembocar en una jerarquía de clases engorrosa y la generación de subclases que pueden tornarse difíciles de gestionar. El Decorador obvia la necesidad de instaurar múltiples subclases para cada combinación de funcionalidades. - Mantenimiento de las clases existentes: En las circunstancias en las que se carece de la facultad para alterar el código fuente de clases ya existentes, tales como bibliotecas o componentes de terceros, el Decorador concede un camino para extender su funcionalidad sin contravenir el principio de "no realizar modificaciones". - Versatilidad en la composición: El Decorador permite la configuración dinámica de objetos mediante la inclusión o eliminación de características mediante la combinación de decoradores. Esto adquiere particular valor en entornos donde los requerimientos puedan variar o donde se precisa una configuración altamente personalizada.
Solución Propuesta	<p>La resolución esbozada por el patrón Decorador implica el establecimiento de una estructura jerárquica de clases que comprende un componente raíz y decoradores que envuelven dicho componente. Cada decorador pone en práctica la misma interfaz que el componente principal y contiene una referencia al mencionado componente. Los decoradores añaden atribuciones adicionales y pueden ser apilados en múltiples capas. Cuando se invoca una operación en el objeto decorado, esta se propaga a través de los decoradores hasta el componente principal.</p> <p>El componente raíz define la interfaz que el cliente emplea para interactuar con el objeto, al tiempo que los decoradores agregan funcionalidad sin modificar esta interfaz.</p>

<p>Diagrama de Clases</p>	 <pre> classDiagram class Interface { <<interface>> +doThis() } class CoreFuncionality { +doThis() } class OptionalWrapper { -wrappee +doThis() } class OptionalOne { +doThis() } class OptionalTwo { +doThis() } class OptionalThree { +doThis() } Interface < -- CoreFuncionality Interface < -- OptionalWrapper OptionalWrapper < -- OptionalOne OptionalWrapper < -- OptionalTwo OptionalWrapper < -- OptionalThree OptionalWrapper o--> Interface : wrappee.doThis(); OptionalWrapper o--> OptionalWrapper : wrappee.doThis(); </pre> <p>The diagram shows an Interface with a <code>+doThis()</code> method. CoreFuncionality and OptionalWrapper implement this interface. OptionalWrapper has a <code>-wrappee</code> attribute and a <code>+doThis()</code> method that delegates the call to <code>wrappee.doThis();</code>. OptionalOne, OptionalTwo, and OptionalThree are subclasses of OptionalWrapper, each implementing <code>+doThis()</code>. A note for OptionalWrapper states: <code>// optional functionality // provided by this class super.doThis(); // more optional functionality</code>.</p>
<p>Diagrama de Secuencia</p>	 <pre> sequenceDiagram participant Client participant ComponentDecoratorA participant ComponentDecoratorB participant ConcreteComponent Client->>ComponentDecoratorA: decorate() ComponentDecoratorA->>ComponentDecoratorB: decorate() ComponentDecoratorB->>ConcreteComponent: decorate() ConcreteComponent-->>ComponentDecoratorB: return ComponentDecoratorB-->>ComponentDecoratorA: return ComponentDecoratorA->>ComponentDecoratorA: decorate() ComponentDecoratorA-->>Client: return </pre> <p>The sequence diagram illustrates the runtime behavior. The Client calls <code>decorate()</code> on ComponentDecoratorA. ComponentDecoratorA then calls <code>decorate()</code> on ComponentDecoratorB, which in turn calls <code>decorate()</code> on ConcreteComponent. ConcreteComponent returns to ComponentDecoratorB, which returns to ComponentDecoratorA. Finally, ComponentDecoratorA performs a self-call to <code>decorate()</code> before returning to the Client.</p>
<p>Participantes</p>	<ul style="list-style-type: none"> - Componente: Define el estándar común que todas las clases (componentes) deben incorporar. Los decoradores y el componente primario obedecen este estándar. - Concreto Componente: Representa la clase madre que será objeto de decoración. Pone en práctica la interfaz del componente y otorga la funcionalidad esencial. - Decorator: Se configura como la entidad abstracta que, además, cumple con la interfaz del componente. Comprende un miembro que mantiene un vínculo con el componente que se pretende adornar. - Concreto Decorador: Corresponde a las clases concretas que heredan del decorador. Integran funciones adicionales al componente principal. La apilación de varios decoradores es factible.
<p>Aplicabilidad</p>	<p>El patrón Decorador se aplica en una variedad de situaciones:</p> <ul style="list-style-type: none"> - Cuando se requiere añadir funcionalidad a objetos sin modificar su código fuente, de manera dinámica y adaptable.

	<ul style="list-style-type: none"> - Cuando se persigue mantener una interfaz uniforme para objetos, tanto decorados como no decorados, permitiendo la sustitución sencilla de objetos decorados por objetos que carecen de decoraciones. - En contextos en los que la jerarquía de subclases resulta en un entramado de clases complejo y complicado, lo que el Decorador contribuye a evitar. - Cuando se trabaja con bibliotecas o componentes de terceros que no admiten modificaciones directas, aunque se necesita ampliar sus capacidades.
Consecuencias	<p>Los efectos de aplicar el patrón Decorador incluyen:</p> <ul style="list-style-type: none"> - Ampliación de funcionalidad: Facilita la incorporación de funciones a objetos sin requerir alteraciones de su código. Respeta el principio SOLID de "abierto para extensión, cerrado para modificaciones". - Combinación flexible: Simplifica la formación de configuraciones específicas de objetos mediante la unión de decoradores. - Mantención de coherencia de interfaz: Asegura que los objetos adornados mantengan la misma interfaz que el componente principal, simplificando la sustitución de objetos con o sin decoraciones. - Potencial de complejidad: Puede dar lugar a una considerable cantidad de clases decoradoras si se abusa de ellas, lo que puede dificultar la comprensión del código. - Identificación del componente principal: En ocasiones, rastrear la jerarquía de decoradores y determinar el componente principal en objetos adornados complejos puede presentar complicaciones.

Implementación del Patrón Decorador

La agencia de viajes “Limitless” ha decidido contratarlos a ustedes para el montaje de su sistema de configuración de planes y paquetes turísticos a la isla de Hawaii en la Polinesia. El sistema se debe configurar a partir de un paquete básico. Cada cliente tendrá dicho plan como mínimo y podrá decorarlo de acuerdo con sus gustos, preferencias y presupuesto.

El paquete base (U\$ 7000 y cinco días) consta de:

- Tiquetes aéreos.
- Hotel en habitación estándar.
- Alimentación.
- Vuelta a la isla.
- Recepción con lei hawaiano y camiseta de Millonarios (es el equipo favorito de los nativos).

Este plan puede ser decorado con mini paquetes de actividades que tienen diferentes costos. Así, por ejemplo, para quienes gustan de la historia y la aviación se tiene un paquete llamado Pearl Harbor (U\$ 653 y 2 días adicionales) que permite visitar el museo del mismo nombre, el palacio Iolani, el museo Bishop y el USS Arizona Memorial.

A quienes les gusta la naturaleza se les ofrece el paquete Nature (U\$ 720 y 5 días más) que consta de visitas al Kualoa Ranch, el Maui Ocean Center y el Akaka Falls State Park.

Finalmente, si usted es deportista extremo podrá comprar el paquete Amazing Hawaii (U\$ 931 y 3 días), con el cual podrá escalar en el Waipio Valley, surfear en las playas Waikiki o Hanalei Bay y bucear en Hanauma Bay. Ahora, si su plan es rumbeo le recomendamos una discoteca para la noche final, pero queremos que conozca Hawaii, no que el licor le haga perder la oportunidad de conocer esta maravilla.

El sistema debe permitir realizar la cotización y duración de un plan por persona y por familia.

Solución:

Enlace Diagrama UML:

<https://drive.google.com/file/d/1IKmI0Jh-anMGMmA7P9bMVUtxFyGJrivl/view?usp=sharing>

Funcionamiento del Código:

```
Menu
1. Mostrar Paquete.
2. Decorar con paquete Amazing Hawaii.
3. Decorar con paquete Nature.
4. Decorar con paquete Pearl Harbor.
5. Cotizar.
6. Salir.
Selecciona una opción: a
Ingresa un número válido.
Presiona Enter para continuar...
```

Figura 1. Validaciones.

```
Paquete Base (U$7000.0 y 5 días) consta de:  
- Tiquetes aéreos.  
- Hotel en habitación estándar.  
- Alimentación.  
- Vuelta a la isla.  
- Recepción con lei hawaiano y camiseta de Millonarios.  
Presiona Enter para continuar...
```

Figura 2. Impresión por Consola del Paquete Base.

```
Paquete actualizado a Paquete Base, Pearl Harbor  
Presiona Enter para continuar...
```

Figura 3. Decoración del Paquete Base con Pearl Harbor.

```
Menu  
1. Mostrar Paquete.  
2. Decorar con paquete Amazing Hawaii.  
3. Decorar con paquete Nature.  
4. Decorar con paquete Pearl Harbor.  
5. Cotizar.  
6. Salir.  
Selecciona una opción: 4  
El paquete Pearl Harbor ya ha sido agregado.  
Presiona Enter para continuar...
```

Figura 4. Validación de que no se agregue un Decorado que ha sido agregado previamente.

```

Paquete Base, Pearl Harbor (U$7653.0 y 7 días) consta de:
- Tiquetes aéreos.
- Hotel en habitación estándar.
- Alimentación.
- Vuelta a la isla.
- Recepción con lei hawaiano y camiseta de Millonarios.
- Visita a Pearl Harbor.
- Museo Iolani.
- Museo Bishop y USS Arizona Memorial.

Presiona Enter para continuar...

```

Figura 5. Impresión por Consola del Paquete Base + Pearl Harbor.

```

Cotización de Paquete Base, Pearl Harbor
Ingrese la cantidad de personas (entre 1 y 10):

```

```

Cotización de Paquete Base, Pearl Harbor
Ingrese la cantidad de días (entre 5 y 20):

```

```

El Paquete Base, Pearl Harbor para 9 persona(s) y 7 dia(s) tiene un costo de U$17219.25

Presiona Enter para continuar...

```

Figura 6. Cotización.

Nota: para realizar la cotización, se partió del supuesto de que el paquete base, junto con sus decorados, estaba destinado para un grupo de 4 personas. A partir de esta premisa, se llevaron a cabo los cálculos necesarios. Es importante señalar que es posible cotizar tanto el paquete base solo, con un decorado adicional, o con todos los decorados incluidos.

Referencias

Decorador | Marco de Desarrollo de la Junta de Andalucía. (s. f.).
<https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/186>

Decorator. (s. f.). <https://reactiveprogramming.io/blog/es/patrones-de-diseno/decorator>

Design patterns and refactoring. (s. f.). https://sourcemaking.com/design_patterns

García, D. (2014, 9 marzo). Patrones Estructurales (III): patrón decorator. Let's code something up! <https://danielggarcia.wordpress.com/2014/03/10/patrones-estructurales-iii-patron-decorator/>

Jwmuun. (s. f.). Patrones de diseño: decorador. Microsoft Learn. <https://learn.microsoft.com/es-es/shows/visual-studio-toolbox/design-patterns-decorator>

Patrón de diseño decorador (Decorator). (s. f.). <https://www3.uji.es/~belfern/Docencia/Presentaciones/ProgramacionAvanzada/Tema2/decorador.html#1>

SL, P. E. C. (s. f.). Patrones de diseño (X): Patrones estructurales - decorator. Programación en Castellano. https://programacion.net/articulo/patrones_de_diseno_x_patrones_estructurales_decorator_1013