



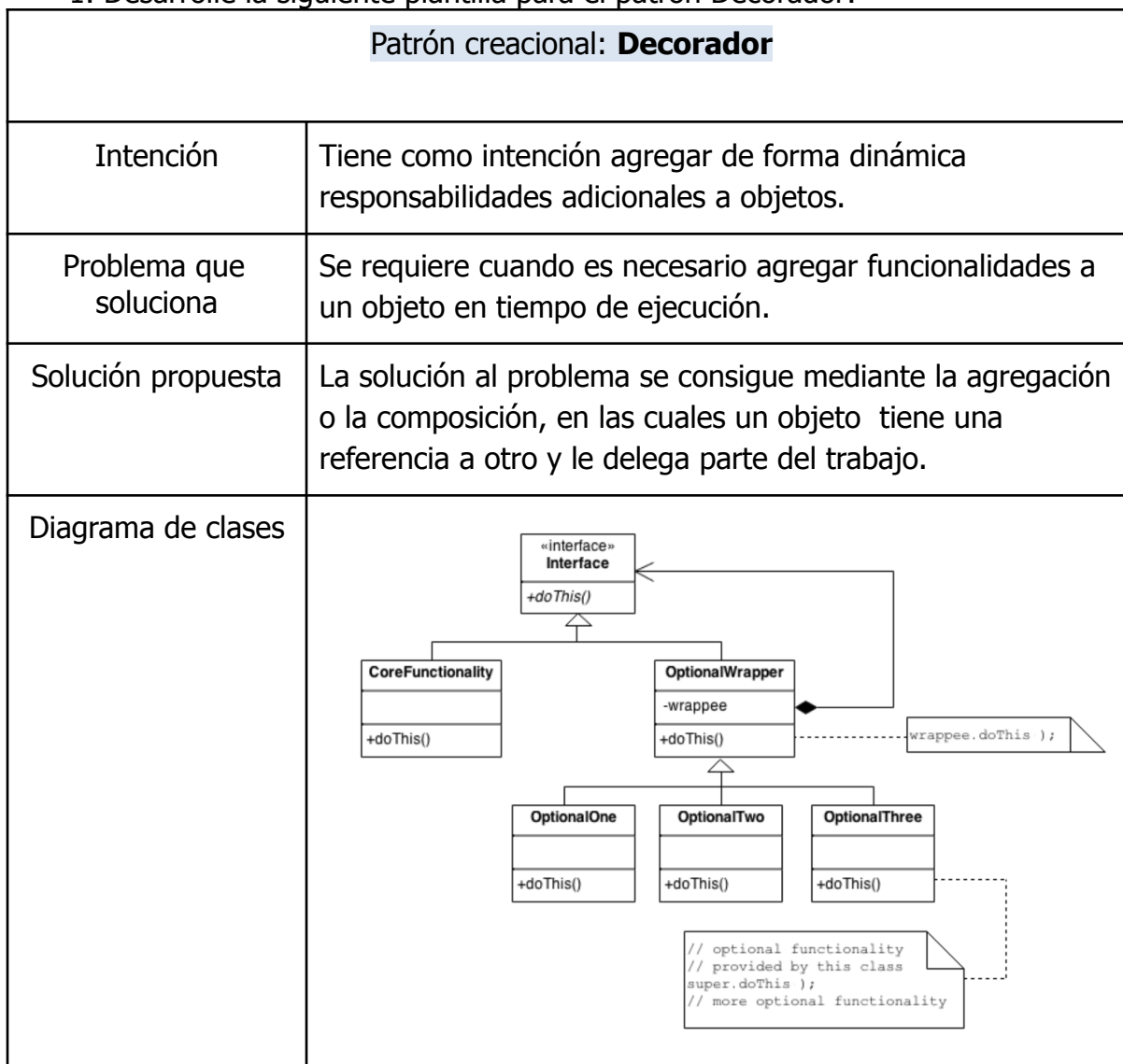
UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS

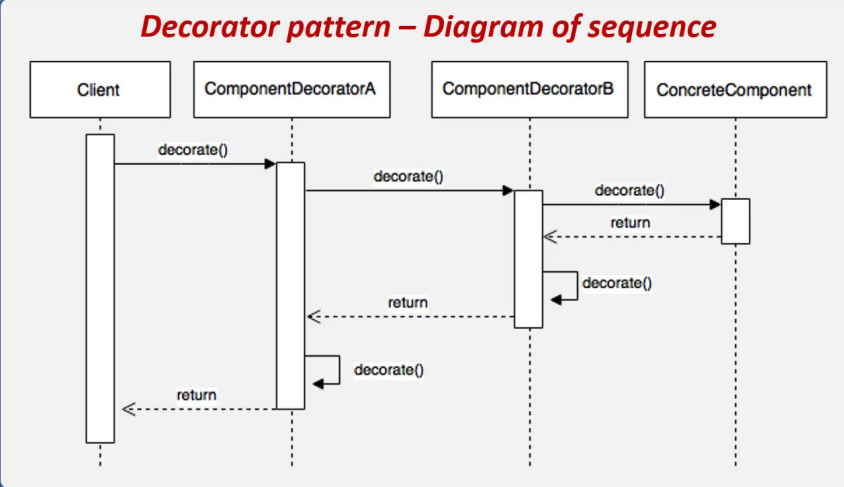
PRACTICA DE LABORATORIO No. 7
Definición del proyecto de trabajo

OBJETIVOS: Comprender e implementar el patrón estructural Decorator (Decorador).

ACTIVIDADES:

1. Desarrolle la siguiente plantilla para el patrón Decorador:



<p>Diagrama de secuencia</p>	<p>Decorator pattern – Diagram of sequence</p>  <pre> sequenceDiagram participant Client participant ComponentDecoratorA participant ComponentDecoratorB participant ConcreteComponent Client->>ComponentDecoratorA: decorate() activate ComponentDecoratorA ComponentDecoratorA->>ComponentDecoratorB: decorate() activate ComponentDecoratorB ComponentDecoratorB->>ConcreteComponent: decorate() activate ConcreteComponent ConcreteComponent-->>ComponentDecoratorB: return deactivate ConcreteComponent ComponentDecoratorB-->>ComponentDecoratorA: return deactivate ComponentDecoratorB ComponentDecoratorA->>ComponentDecoratorA: decorate() deactivate ComponentDecoratorA ComponentDecoratorA-->>Client: return deactivate ComponentDecoratorA </pre>
<p>Participantes</p>	<ul style="list-style-type: none"> • Interface (Component) • Concrete component • Base decorator • Concrete Decorators
<p>Aplicabilidad</p>	<p>Permite estructurar tu lógica de negocio en capas, crear un decorador para cada capa y componer objetos con varias combinaciones de esta lógica, durante el tiempo de ejecución. El código cliente puede tratar a todos estos objetos de la misma forma, ya que todos siguen una interfaz común.</p>
<p>Consecuencias</p>	<ul style="list-style-type: none"> - Puedes extender el comportamiento de un objeto sin crear una nueva subclase. - Puedes añadir o eliminar responsabilidades de un objeto durante el tiempo de ejecución. - Puedes combinar varios comportamientos envolviendo un objeto con varios decoradores. - Principio de responsabilidad única. Puedes dividir una clase monolítica que implementa muchas variantes posibles de comportamiento, en varias clases más pequeñas. - Resulta difícil eliminar un wrapper específico de la pila de wrappers. - Es difícil implementar un decorador de tal forma que su comportamiento no dependa del orden en la pila de decoradores. - El código de configuración inicial de las capas pueden tener un aspecto desagradable.

2. Desarrolle una implementación del patrón para la siguiente situación problemática:

La agencia de viajes "Limitless" ha decidido contratarlos a ustedes para el montaje de sus 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 muso 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 mas) 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.