Patrones de diseño, conceptos y cuando usarlos:

|  |  |  |  |
| --- | --- | --- | --- |
| Patrón | ¿Qué hace? | ¿Cuándo Usarlo? | Ejemplos |
| Singleton | Asegura que una clase tenga una única instancia | Cuando algo debe existir una sola vez, por ejemplo, un Login, donde el usuario solo debe estar logueado una sola vez |  |
| Factory Method | Permite a una clase delegar la instanciación de objetos a subclases | Cuando tengo una superclase con varias variantes que cambian según condiciones | Crear discos de distintos tipos:  (CrearCD(“DVD”, “Simple”)) |
| Abstrac Factory | Similar al Factory Method, pero para crear familias de objetos relacionados | Cuando necesito varios objetos coherentes entre sí | Crear conjunto:  (FabricaDVD.CrearSimple() o .CrearDoble()) |
| Composite | Permite tratar objetos individuales y compuestos de forma uniforme | Cuando tengo una estructura jerárquica, como un árbol o una recursiva | Un campo dividido en subcampos más chicos |
| Observer | Permite que objetos se subscriban a eventos de otro | Cuando varios objetos deben reaccionar a un cambio en otro | GUI que camba idioma cuando el usuario lo cambia |

1. Dibuje el diagrama asociado con el proceso de desarrollo de software proceso unificado (UP). Explique en que se basa y cuáles son sus fases.

Fases del proceso unificado:

Inicio 🡪 Elaboración 🡪 Construcción 🡪 Transición.

Este se basas en el desarrollo iterativo e incremental, y tiene su enfoque centrado en los casos de uso con su modelado visual en UML.

Fase de Inicio: esta contiene la visión, objetos y alcance del sistema. (Acá van los casos de uso)

Fase de Elaboración: contiene la arquitectura base y los casos de uso críticos.

Fase de Construcción: Desarrollo completo de la funcionalidad. (Acá pueden ir las Iteraciones)

Fase de Transición: Despliegue al usuario final.

1. Nombre y explique los 4 elementos de cada patrón.

Todo patrón de diseño incluye:  
Nombre 🡪 Identificador del patrón, por ejemplo, Singleton o composite.

Problema 🡪 Qué situación resuelve o en que contexto se aplica.

Solución 🡪 Estructura (Clases, relaciones, responsabilidades) que lo implementa.

Consecuencias 🡪 Ventajas, desventajas y efectos de aplicar el patrón.

1. Composite: ¿cuándo se utiliza? Realizar su diagrama de clases, realizar el diagrama de clases de un ejemplo practico distinto a los vistos en clase.

¿Cuándo se usa? El patrón composite se utiliza cuando se desea representar estructuras jerárquicas del tipo padre-todo, y trata de forma uniforme a los objetos individuales y a los grupos de objetos compuestos. Permite que el cliente trabaje con componentes simples y compuestos mediante la misma interfaz “Composite”.

Ejemplo:

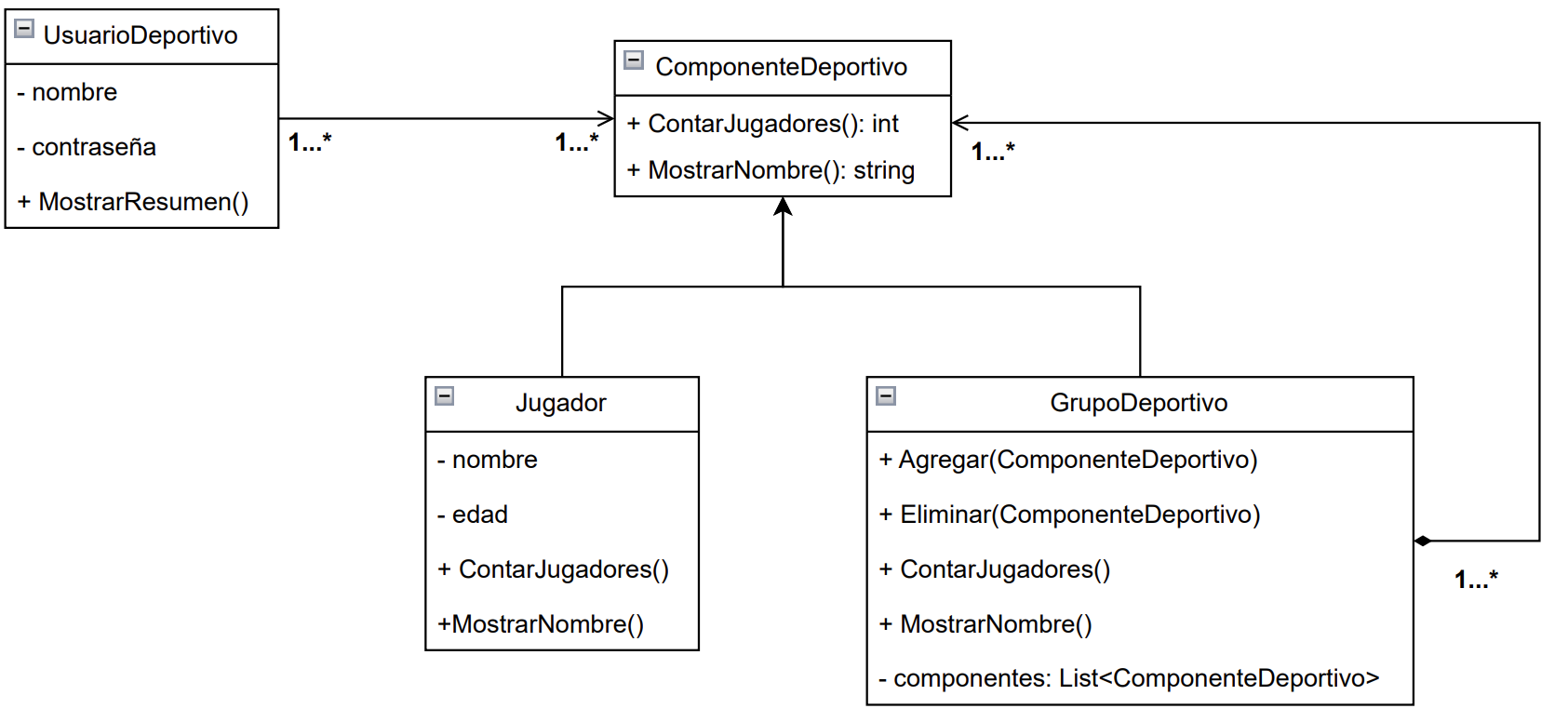
Hay jugadores (hojas) que pertenecen a equipos.

A su vez, se pueden crear Ligas (compuestos) que agrupan varios equipos o incluso otras ligas.

Cada entidad (jugador, equipo, liga) se puede recorrer de forma uniforme gracias al patrón Composite.

Además, hay una restricción: los jugadores no deben superar los 21 años.

Se puede aplicar una operación de ContarJugadores() en cualquier nivel del árbol.



1. Singleton: ¿cuándo se utiliza? Realizar un su diagrama de clases, explique porque y como lo aplico a su proyecto, realizar el diagrama de clases por capas de su proyecto.

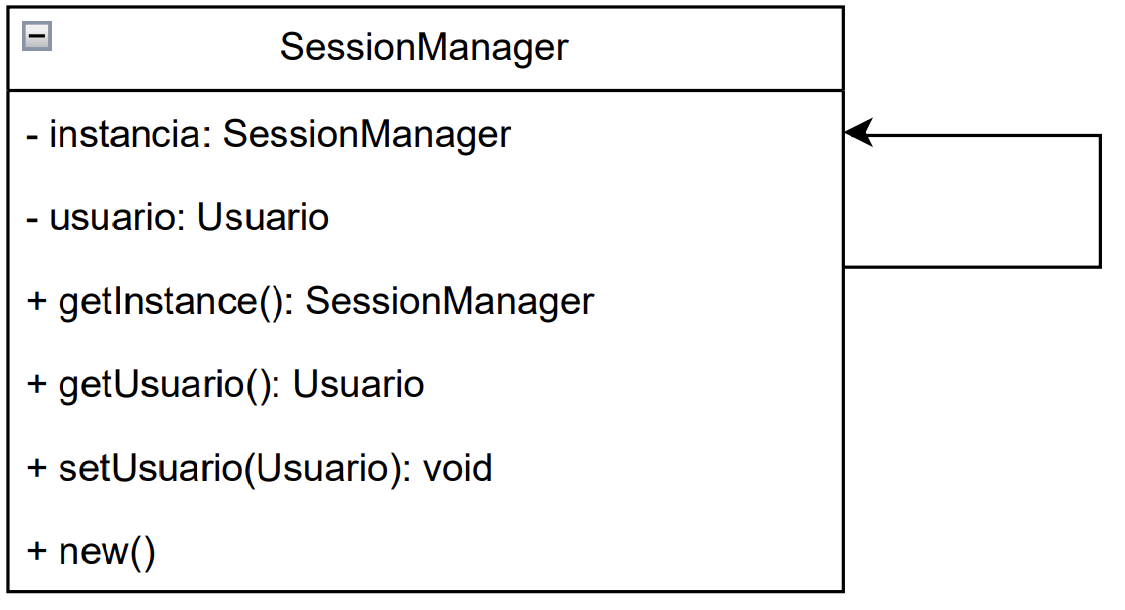
¿Cuándo se utiliza? El patrón Singleton se utiliza cuando debe existir una única instancia de una clase durante toda la ejecución del sistema, y se necesita un punto de acceso global a esa instancia.

¿Como lo aplique en mi proyecto? En mi proyecto utilice el patrón Singleton en la clase SessionManager para gestionar la sesión del usuario logueado:

Se guarda una única instancia del Usuario que accede al sistema.

Permite acceder a los datos del usuario desde formularios como GestionDeUsuarios, Reservas, MenuPrincipal.

Facilita la lógica de permisos según el tipo de usuario ya que habilita o deshabilita las opciones del menú.



1. Medida: Dar su definición y un ejemplo práctico.

Una medida es un valor numérico obtenido como resultado de una medición en el contexto del desarrollo de software. Representa una cantidad concreta que puede ser utilizada para generar métricas e indicadores posteriores.

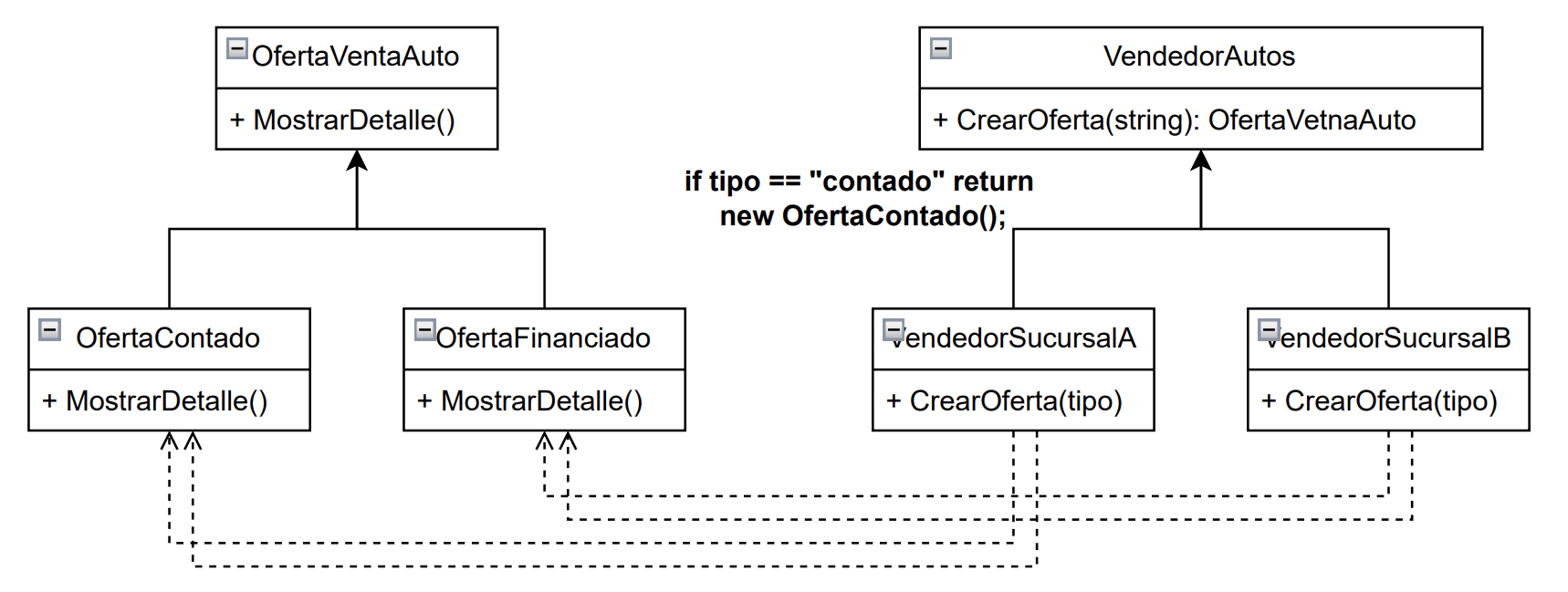
Un ejemplo de medición puede ser por ejemplo en el sistema StageLink, si contamos la cantidad de reservas activas realizadas por una banda en un mes y obtenemos el valor “12”, ese numero 12 es una medida, luego esta medida la podemos usar para calcular una métrica como el porcentaje de ocupación por banda en el mes para mejorar el sistema.  
PARTE PRACTICA:

Realizar una automatización de software de dos ofertas de ventas de autos.

Patrón, FACTORY METHOD:

Porque el sistema debe crear diferentes tipos de ofertas de forma flexible y escalable.

Cada tipo de oferta puede implicar distintos precios, condiciones o detalles, pero todas deben derivar de una misma estructura base. El patrón Factory Method permite delegar la creación de estas ofertas en subclases concretas.



Se eligió el Factory Method porque el sistema requiere crear ofertas de venta de autos de distintas modalidades (contado y financiado).

Cada tipo de oferta tiene comportamiento distinto, pero todas comparten una interfaz común (OfertaVentaAuto).

El patrón permite mantener el código flexible ante futuras extensiones (por ejemplo, agregar una OfertaCanje).

OTRO TEMA:  
1. ¿Qué es un proceso de software?

Un proceso de software es un conjunto estructurado de actividades, acciones y tareas necesarias para construir un software de alta calidad. Este define el marco de trabajo que guía el desarrollo desde el análisis inicial hasta el mantenimiento, integrando métodos y herramientas. En este se incluyen actividades como la comunicación, el modelado, la construcción, las pruebas y el despliegue.

Su objetivo principal es obtener software fiable, mantenible y funcional aplicando principios de ingeniería.

2. Las propiedades que hacen útil a un patrón de diseño son:

Que son reutilizables: es decir que pueden aplicarse a múltiples sistemas.

Que son generalizables: es decir que resuelven muchos problemas comunes no específicos.

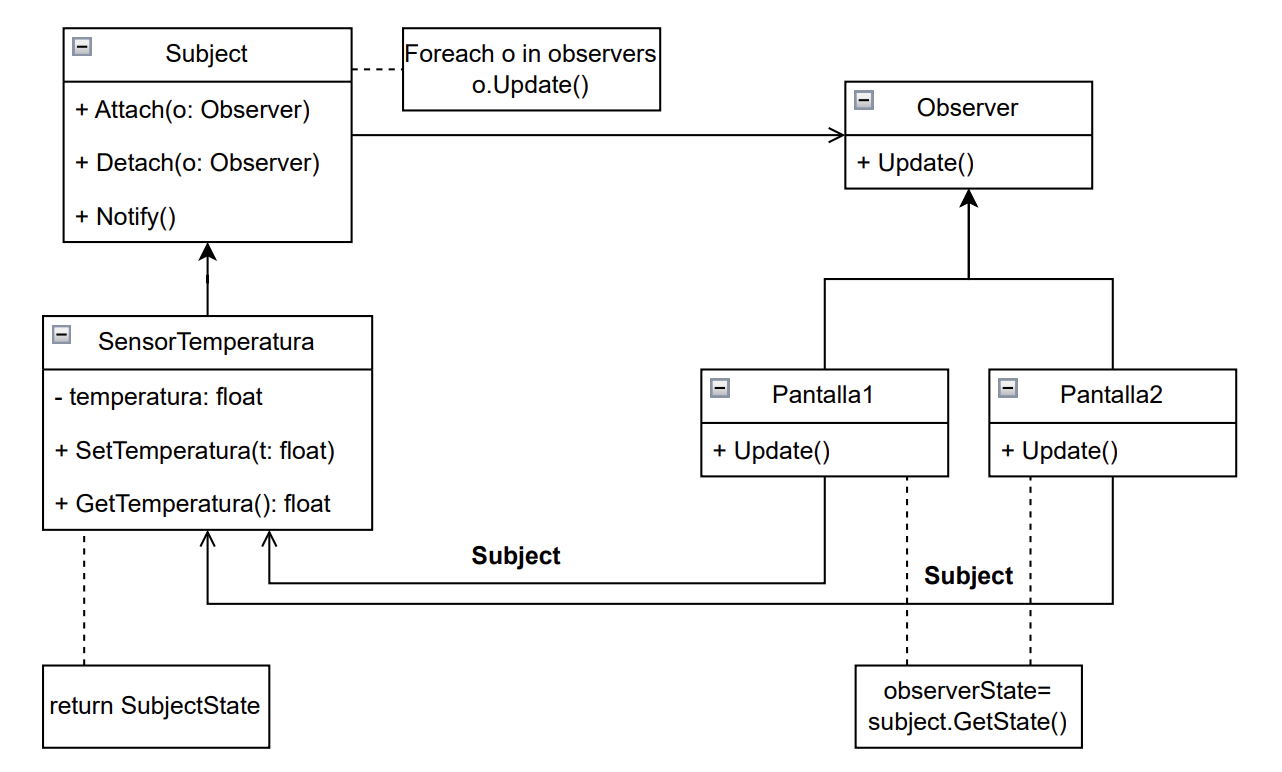
Que son flexibles: es decir que tienen adaptación y extensión.

Que son estandarizados: por eso se dice que facilitan la comunicación entre desarrolladores.

1. Observer: ¿cuándo se utiliza? Realizar su diagrama de clases, realizar el diagrama de clases de un ejemplo práctico distinto a los vistos en clase.

¿Cuándo se utiliza? Se utiliza cuando se desea notificar automáticamente a varios objetos (observadores) sobre un cambio de estado en otro objeto (sujeto), sin acoplarlos directamente. Es ideal para casos donde hay dependencias de 1 a muchos.

Un ejemplo de esto podría ser un sensor de temperatura, este SensorTemperatura (sujeto) registra la temperatura actual. Existen varias Pantallas (Observadores) en distintos lugares, que deben ser actualizadas automáticamente cada vez que cambia el valor de la temperatura.



1. Singleton igual al tema anterior.
2. Métricas, concepto de métricas y un ejemplo práctico.

Es una combinación de una o mas medidas que se interpreta en un contexto especifico, con el objetivo de evaluar un atributo del software, como su calidad, rendimiento, productividad o mantenibilidad.

La métrica permite comparar, tomar decisiones y mejorar procesos durante el desarrollo de software.

Un ejemplo seria en el sistema StageLink:

Cantidad promedio de reservas por banda en un mes.

Esta métrica surge de tomar como medida el total de reservas realizadas por todas las bandas y dividirlo por cantidad de bandas activas.

Sirve para evaluar la actividad del sistema, detectar bajas en el uso o tomar decisiones comerciales.