Patrones de diseño

Sesión 12:

PATRONES COMPORTAMIENTO: INTRODUCCIÓN A PATRONES DE COMPORTAMIENTO. PATRÓN STATE Y PATRÓN OBSERVER.

Universidad Tecnológica del Perú

Logro de la sesión

Al finalizar la sesión, el estudiante elabora su programación usando patrones de comportamiento: Introducción a patrones de comportamiento. Patrón State y Patrón Observer.

Universidad Tecnológica del Perú

Patrones de Diseño

		PROPÓSITO		
		CREACIONAL	ESTRUCTURAL	COMPORTAMIENTO
ALCANCE	CLASE	Factory Method	Adapter	Interpreter Template Method
	OBJETO	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight Observer State Strategy Visitor

Universidad Tecnológica del Perú

PATRON COMPORTAMIENTO

Los patrones de comportamiento tratan con algoritmos y la asignación de responsabilidades entre objetos.

https://refactoring.guru/es/design-patterns/behavioral-patterns

Universidad Tecnológica del Perú

Patrón State

El patrón de diseño State es un patrón de comportamiento que nos permite cambiar el comportamiento de un objeto en función del estado en el que se encuentre.
Este patrón nos permite desacoplar el comportamiento de un objeto de su implementación, lo que nos permite cambiar el comportamiento de un objeto en tiempo de ejecución de forma sencilla y mantenible.

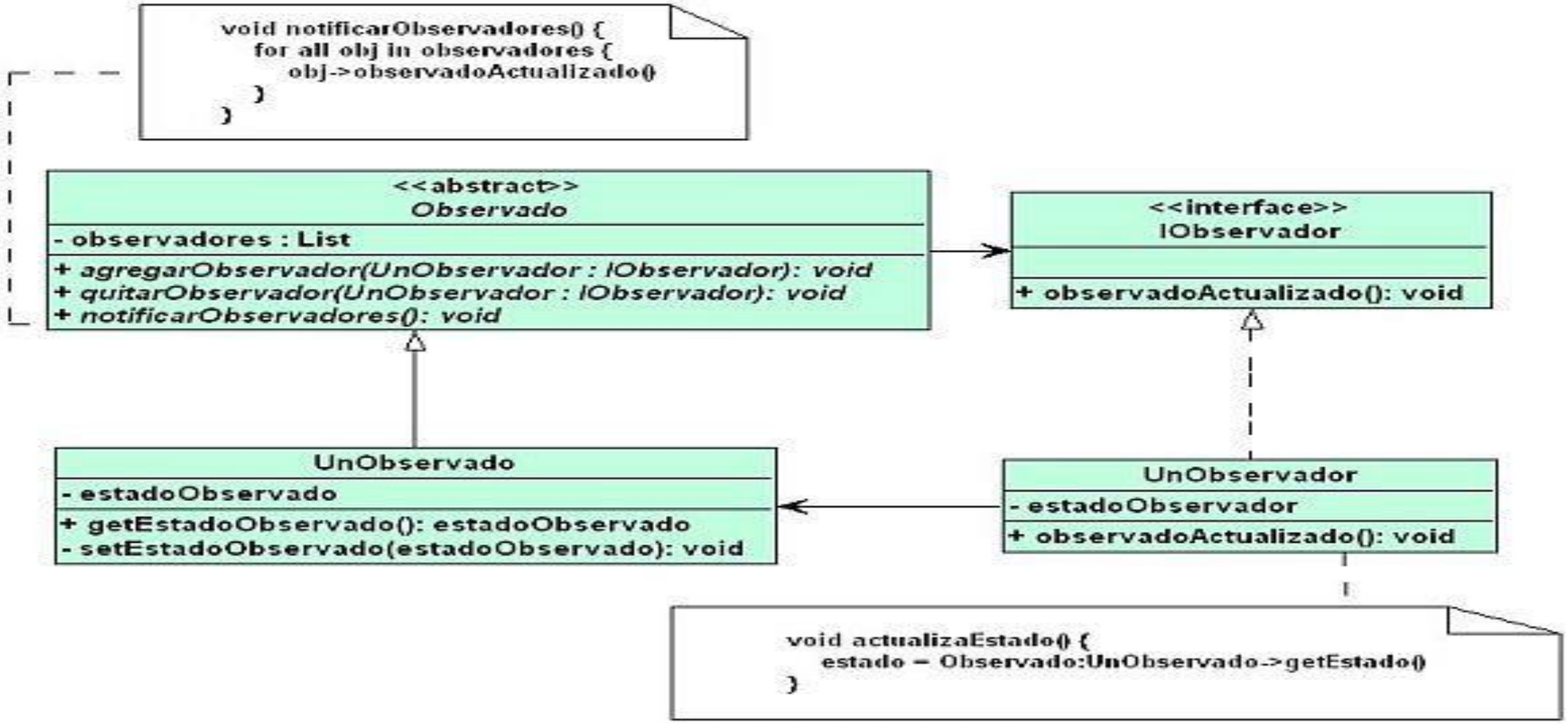
Enlaces:

https://devexpert.io/state-patrones-diseno/
https://refactoring.guru/es/design-patterns/state/java/example

Universidad Tecnológica del Perú

Patrón Observer

El patrón **Observer** puede ser utilizado cuando hay objetos que dependen de otro, necesitando ser notificados en caso de que se produzca algún cambio en él.



https://refactoring.guru/es/design-patterns/observer/java/example

Universidad Tecnológica del Perú

Explicación: Patrón Observer

Se utilizarán los métodos agregarObservador() y quitarObservador() de la clase abstracta UnObservado para registrar en una lista qué objetos de tipo UnObservador deberán ser notificados o dejar de serlo cuando se produzca algún cambio en él (en tal caso recorrerá dicha lista para enviar una notificación a cada uno de ellos).

El mensaje será enviado a UnObservador (que implementa la interface IObservador) utilizando su método observadoActualizado().

Veamos a continuación un ejemplo sencillo en lenguaje Java en el cual cuando se produce un cambio en el observado éste envía una notificación a los observadores, que simplemente mostrarán un mensaje al recibirla.

Siguiente enlace:

https://informaticapc.com/patrones-de-diseno/observer.php

Universidad Tecnológica del Perú

¿Consultas o dudas?

Universidad Tecnológica del Perú

Actividad

Resolver la actividad planteada en la plataforma.

Universidad Tecnológica del Perú

Cierre

¿Qué hemos aprendido hoy?

Elaboramos nuestras conclusiones sobre el tema tratado

Universidad Tecnológica del Perú

Universidad
Tecnológica
del Perú