Trabajo Extraclase II: Patrones de Diseño

Harold Espinoza Matarrita

2019185140

Sebastián Moya Monge

2019077209

Instituto Tecnológico de Costa Rica

Ingeniería en Computadores

Estructuras de Datos II

Índice

Patrón de Diseño Observer

El patrón de diseño observer, consiste en una técnica de programación que se encarga de notificar a objetos que poseen cierta dependencia de un objeto en específico de que este a realizado un cambio. Cabe aclarar, si la situación lo amerita los objetos dependientes puede solicitar más información sobre el cambio realizado.

La implementación del patrón se explica en el siguiente diagrama:

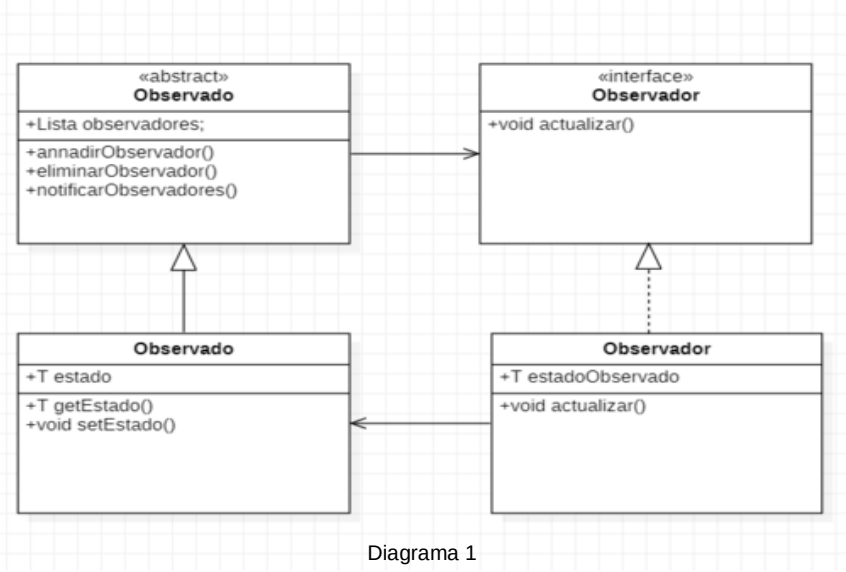


Diagrama 1

Como se muestra en el diagrama de clases todo objeto que es observado va a tener una relación de herencia con la clase "observado padre" la cual va a poseer una lista de observadores y tres métodos adicionales, los cuales cumplen con la función de añadir un observador a la lista, eliminar un observador de la lista y notificar de un cambio en el estado del observador. Los observadores, implementando el interfaz observador, reciben el mensaje del cambio y modifican sus características con el método actualizar de ser necesario.

Dicho patrón se puede usar en varios contextos, pero dos de los casos más característicos son los siguientes:

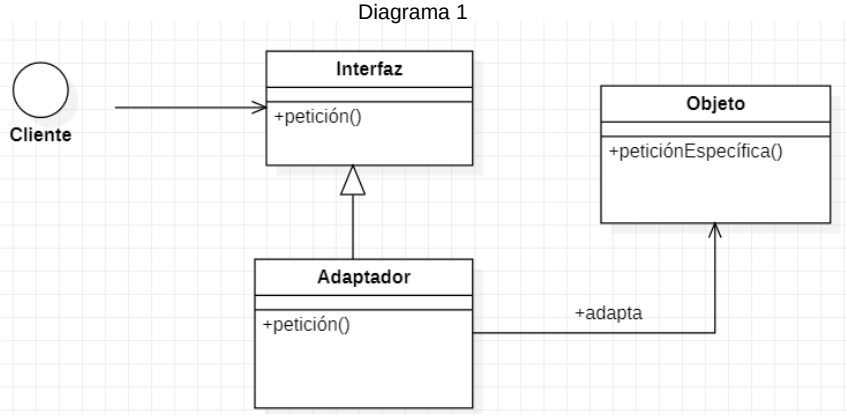
- El patrón es de mucha utilidad cuando múltiples objetos dependen de uno solo. Por ende, es muy recomendado en la programación orientada a objetos (POO).
- El patrón también es ideal en el caso de que múltiples objetos deben ser informados de un cambio de estado en un objeto en específico.

Patrón de Diseño Adapter

El patrón de diseño adapter es considerado como un patrón estructural que puede ser usado tanto en clases como en objetos. Este patrón cumple con el objetivo de permitir la cooperación de clases que en un principio son diferentes, esto para extender funcionalidades, todo esto mediante el acceso a funciones diferentes creando una manera en común para cada clase/objeto. La finalidad de esta manera de codificar es procurar ahorrar la mayor cantidad de código posible.

La implementación del patrón se explica en el siguiente diagrama:

Diagrama 1



En el diagrama 1 vemos la interacción de varios objetos dentro de un programa, los cuales son el cliente, el adaptador, la interfaz y el objeto a adaptar.

El objeto por adaptar es aquel al que se debe ajustar bajo una nueva interfaz, en este caso se planea adaptar el método de petición específica para poder ser llamada por el método petición. La interfaz es aquella con la que interactúa el cliente, el objeto debe ser adaptado para que pueda ser usado bajo las características de la interfaz. El adaptador es el que se encarga de que el objeto obedezca a la interfaz y pueda ser funcional con ella. Y finalmente el cliente es el que interactúa con la interfaz.

Dos de los principales contextos donde se debe usar este patrón son los siguientes:

- Principalmente este patrón debe de ser utilizado cuando se desea tener clases reutilizables que cooperen con otras clases no relacionadas.
- Por otra parte, hablando de objeto propiamente, se puede utilizar con al igual que las clases, varios objetos son incompatibles entre si, pero cumplen con funcionalidades similares, por lo cual se opta por este patrón.

Referencias

Hernandez, J. J. (28 de Agosto de 2013). *Blog Seas*. Obtenido de Blog Seas: https://www.seas.es/blog/informatica/patrones-de-diseno-en-java-patron-observer/

Max. (11 de Junio de 2011). *Mi granito de Java*. Obtenido de Mi granito de Java: http://migranitodejava.blogspot.com/2011/06/observer.html

Sanchez, M. (17 de Agosto de 2018). *Medium*. Obtenido de Medium: https://medium.com/all-you-need-is-clean-code/una-interfaz-para-controlarlos-a-todos-patr%C3%B3n-adapter-a9073f3460b