Autores:
Delgado Díaz Itzel Asuzena, Martinez Mendoza Miguel Angel, Rojas Espinoza Luis Angel

¿En qué caso se podrían usar los patrones de diseño Observer y Strategy?

Observer .- Imaginemos un programa bastante complejo donde tengamos muchos módulos que se encargan de hacer su tarea en la aplicación, pensemos que estos módulos son inconexos entre ellos, evidente mente tendremos que encontrar una manera en que se comuniquen entre ellos y conozcan los estados de al menos uno ellos, podríamos suponer que necesitaremos una especie de gestor que los conecte, este gestor se encargara de recibir las notificaciones de estado de todos nuestros módulos y a su vez notificarlo a todos los demás, justamente acabamos de hacer una breve descripción del patrón Observer, donde el gestor o nexo pasaría a llamase sujeto y los módulos observadores, ejemplos mas en concreto de esto podrían ser, sistemas de suscripción online donde el sujeto seria el servidor y los observadores los usuarios suscritos a este servicio.
Strategy .- Este algoritmo es aplicable cuando muchas clases relacionadas difieren solo en su comportamiento, la estrategia permiten configurar una clase con un comportamiento de entre muchos posibles, también lo usamos cuando se necesitan distintas variantes de un algoritmo o cuando un algoritmo usa datos que los clientes no deberían conocer, el patrón evitar exponer estructuras complejas y dependientes del algoritmo.

Para dichos patrones de diseño, ¿Cuáles son las desventajas de usarlos?

Observer .-

- No se especifica el receptor de una actualización. Se envía a todos los objetos interesados.
- Actualizaciones inesperadas. Se podrían producir actualizaciones en cascada muy ineficientes.

Strategy .-

- Factoriza aspectos comunes de una familia de algoritmos y utilizarlos en las clases base de la jerarquía.
- Aumenta cohesión del cliente.
- Sistematiza el uso de implementaciones alternativas
- El cliente es el responsable de crear estrategias, por tanto debe comprender las posibilidades que ofrecen, esto es, debe ser relevante para el contexto del cliente.
- Menor eficiencia. Aumenta el número de objetos creados.

NOTA: La pratica01 se corre usando el comando “ant” desde la terminal.