Patrones de Diseño: Patrones de Comportamiento. Tema 5-1: Introducción

Patrones de Comportamiento

- Los Patrones de Comportamiento están relacionados con el flujo de control de un sistema y la asignación de responsabilidades a objetos.
- Se centran en la interacción y cooperación entre clases. Los patrones de comportamiento hablan de como interaccionan o se comunican entre sí los objetos para conseguir ciertos resultados.
- Ciertas formas de organizar el control de un sistema pueden derivar en grandes beneficios para la eficiencia y el mantenimiento del sistema.

Clasificación Patrones de Comportamiento

- **Chain of Responsability:** Establece una cadena en un sistema, para que un mensaje pueda ser manejado en el nivel en el que se recibe en primer lugar, o ser redirigido a un objeto que pueda manejarlo.
- **Command:** Encapsula un comando en un objeto de tal forma que pueda ser almacenado, pasado a métodos y devuelto igual que cualquier otro objeto.
- **Interpreter:** Define un intérprete para un lenguaje.
- **Iterator:** Proporciona una forma coherente de acceder secuencialmente a los elementos de una colección, independientemente de su tipo.

Clasificación Patrones de Comportamiento

- **Mediator:** Simplifica la comunicación entre los objetos de un sistema introduciendo un único objeto que gestiona la distribución de mensajes entre otros objetos.
- **Memento:** Guarda una "imagen" del estado de un objeto, de forma que pueda ser devuelto a su estado original sin revelar su contenido al resto del mundo.
- **Observer:** Proporciona a los componentes una forma flexible de enviar mensajes de difusión a los receptores interesados.
- **State:** Permite modificar fácilmente el comportamiento de un objeto en tiempo de ejecución dependiendo de su estado.

Clasificación Patrones de Comportamiento

- **Strategy:** Define un grupo de clases que representa un conjunto de posibles comportamientos. Estos comportamientos pueden ser fácilmente intercambiados en una aplicación, modificando la funcionalidad en cualquier instante.
- **Visitor:** Proporciona una forma fácil y sostenible de ejecutar acciones en una familia de clases. Este patrón centraliza los comportamientos y permite que sean modificados o ampliados sin cambiar las clases sobre las que actúan.
- **Template Method:** Proporciona un método que permite que las subclases redefinan partes del método sin rescribirlo.