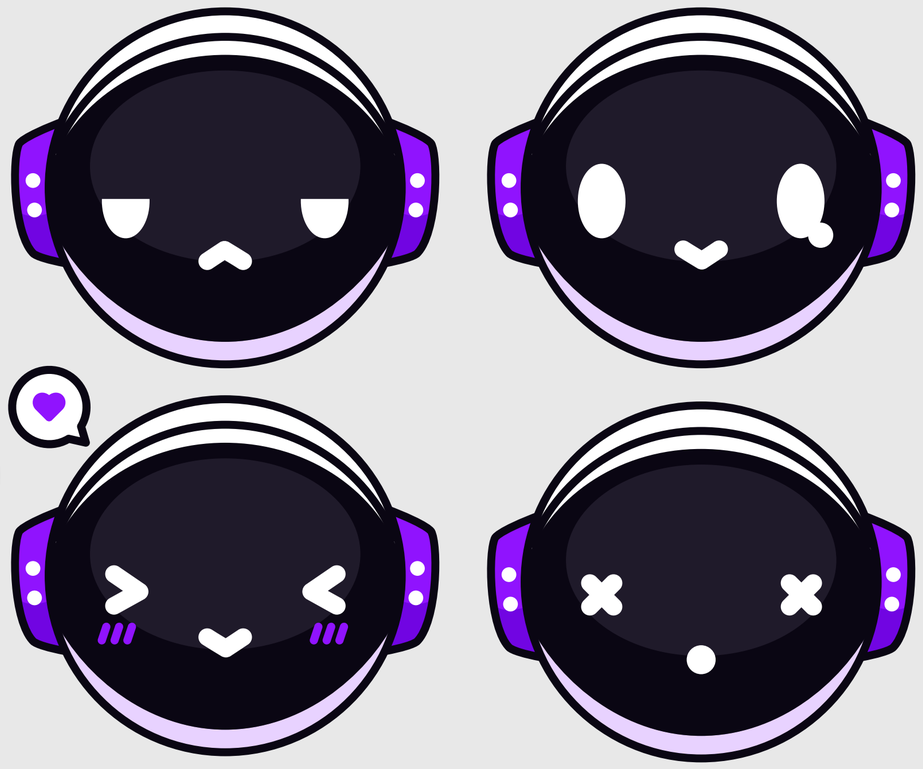
Patrones de diseño de comportamiento



Estos tipos de patrones están enfocados en cómo, por medio de algoritmos, los objetos interactúan y delegan responsabilidades, facilitando las diversas formas en que pueden interactuar los objetos en tareas complejas.

Algunos de los principales patrones de comportamiento son:

Chain of responsibility (Cadena de responsabilidad): por medio de una "cadena" de manejadores se pueden pasar solicitudes, de esta manera cada manejador decide procesar la solicitud o pasarlo al siguiente manejador en la cadena. Esto es especialmente útil para procesos que esperan diferentes tipos de solicitudes pero no se conoce su tipo exacto previamente.

Observer (Observador): permite definir una suscripción para informar a otros objetos sobre cualquier cambio (eventos) que ocurran en el objeto que observan. Facilita los cambios en otros objetos cuando otro (objeto observado) cambie su estado.

Strategy (Estrategia): este patrón permite definir una serie de algoritmos dentro de clases individuales y hacer los objetos intercambiables. Facilita la ejecución del código que tenga muchas clases similares pero difieren en algún comportamiento, ademas se puede cambiar un algoritmo por otro en tiempo de ejecución.

Memento (Recuerdo): este patrón permite guardar y restaurar un estado previo de un objeto sin revelar detalles de un implementación. Facilita la creación de un historial de estados por los que ha pasado un objeto.

Mediator (Mediador): este patrón restringe comunicaciones directas entre los objetos, lo cual los obliga a colaborar a través del objeto mediador, permitiendo reducir las dependencias entre objetos. Es especialmente útil cuando hay clases que están acopladas a otras clases.

Tambien existen otros como:
command, iterator, state, template method y visitor.

Links de referencia 1 Cursos relacionados Más recursos