Patrones de Diseño: Patrones de Comportamiento. Tema 5-3: Command

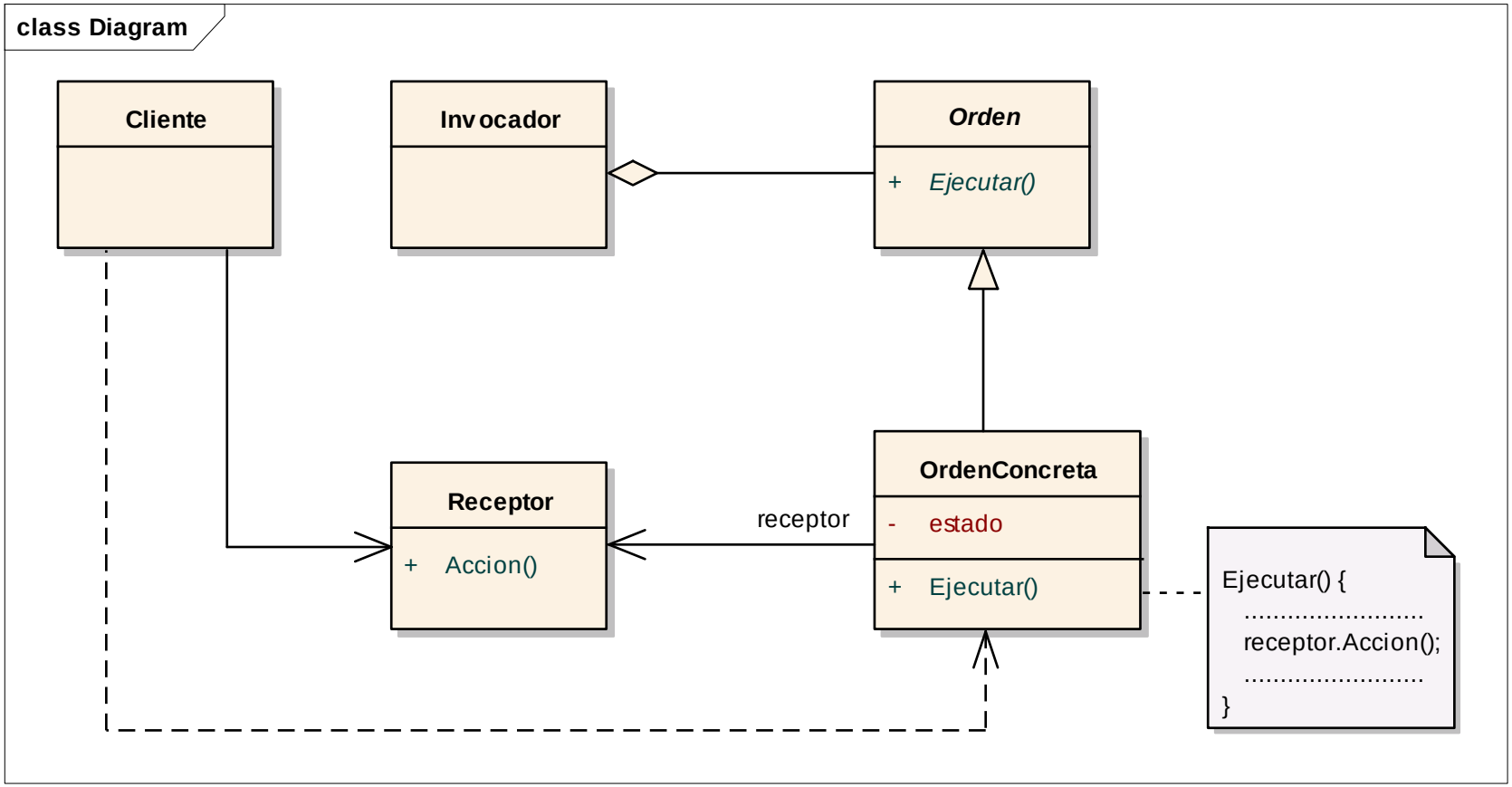
Descripción del patrón

- **Nombre:**
  - Comando, orden
  - También conocido como action, transaction
- **Propiedades:**
  - Tipo: comportamiento
  - Nivel: objeto, componente
- **Objetivo o Propósito:**
  - Encapsular un comando en un objeto. Este objeto contiene el comportamiento y los datos necesarios para una acción específica. Permite parametrizar a los clientes con diferentes peticiones, hacer cola o llevar un registro de las peticiones. Además permite deshacer operaciones.

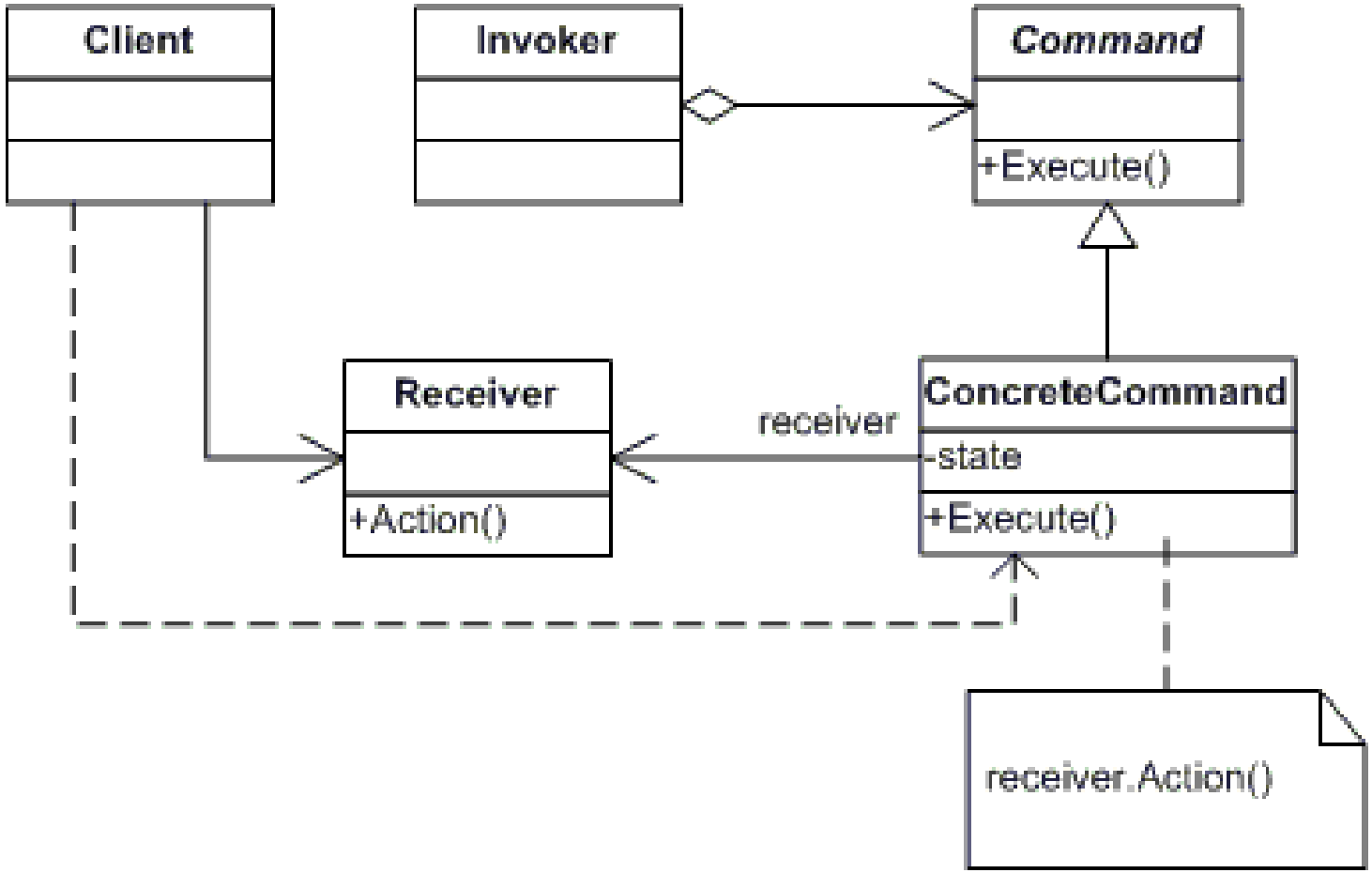
Aplicabilidad

- Use el patrón Command cuando:
  - Parametrizar objetos con una acción a realizar.
  - Se quiera desacoplar la fuente de una petición del objeto que la cumple.
  - Haya que implementar un mecanismo de rehacer/deshacer acciones.
  - Haya que poner en cola y ejecutar comandos en momentos distintos.
  - Permitir registrar los cambios de manera que se puedan volver a aplicar en caso de una caída del sistema.
  - Estructurar un sistema mediante operaciones de alto nivel basadas en operaciones más sencillas (primitivas). Sistemas transaccionales.

Estructura


class Diagram
Cliente
Invocador
Orden
+ Ejecutar()
Receptor
+ Accion()
receptor
OrdenConcreta
- estado
+ Ejecutar()
Ejecutar() {
.....
receptor.Accion();
.....
}

Estructura


Client
Invoker
Command
+Execute()
Receiver
+Action()
receiver
ConcreteCommand
-state
+Execute()
receiver.Action()

Estructura. Participantes

- **Orden (Comando):** Interfaz en la que se definen los métodos que se usarán por el invocador. Además estos métodos serán implementados por cada uno de los comandos concretos. En el diagrama solo se muestra el método ejecutar, pero se pueden añadir más métodos como deshacer y rehacer.
- **OrdenConcreta (Comando concreto):** Implementación de la interfaz Comando. Mantiene una referencia con el receptor. Implementa el método ejecutar para que realice la acción propia de cada comando concreto.
- **Invocador:** El que llama al método ejecutar del objeto Comando.
- **Receptor:** Para quien Comando cumple la petición solicitada.
- **Cliente:** Crea un objeto OrdenConcreta y establece su receptor.

Estructura. Variaciones

- **Variaciones del patrón:**
- **Deshacer.** El patrón se puede extender para proporcionar la capacidad de deshacer los comandos. Simplemente se amplía la interfaz Comando (Orden) y se añade el método deshacer, invirtiendo el último comando.

  Si se desea deshacer un conjunto de comandos y no solo el último debemos guardar un histórico de comandos.
- **MacroCommand.** Es una colección de comandos. Se suele componer mediante el patón Composite. Un objeto MacroCommand contiene una lista de subcomandos. Cuando se llama al método ejecutar se redirige a los subcomandos.

Consecuencias

- Se desacopla la parte de la aplicación que invoca los comandos de la implementación de los mismos.
- Al tratarse los comandos como objetos, se puede realizar herencia de los mismos, composiciones de comandos (mediante el patrón Composite).
- Permite reemplazar objetos comando en tiempo de ejecución y hace que los comandos sean objetos normales, teniendo todas las propiedades usuales de los objetos.
- Facilita la introducción de nuevos comandos; simplemente escribiendo otra implementación de la interfaz e introduciéndola en la aplicación.

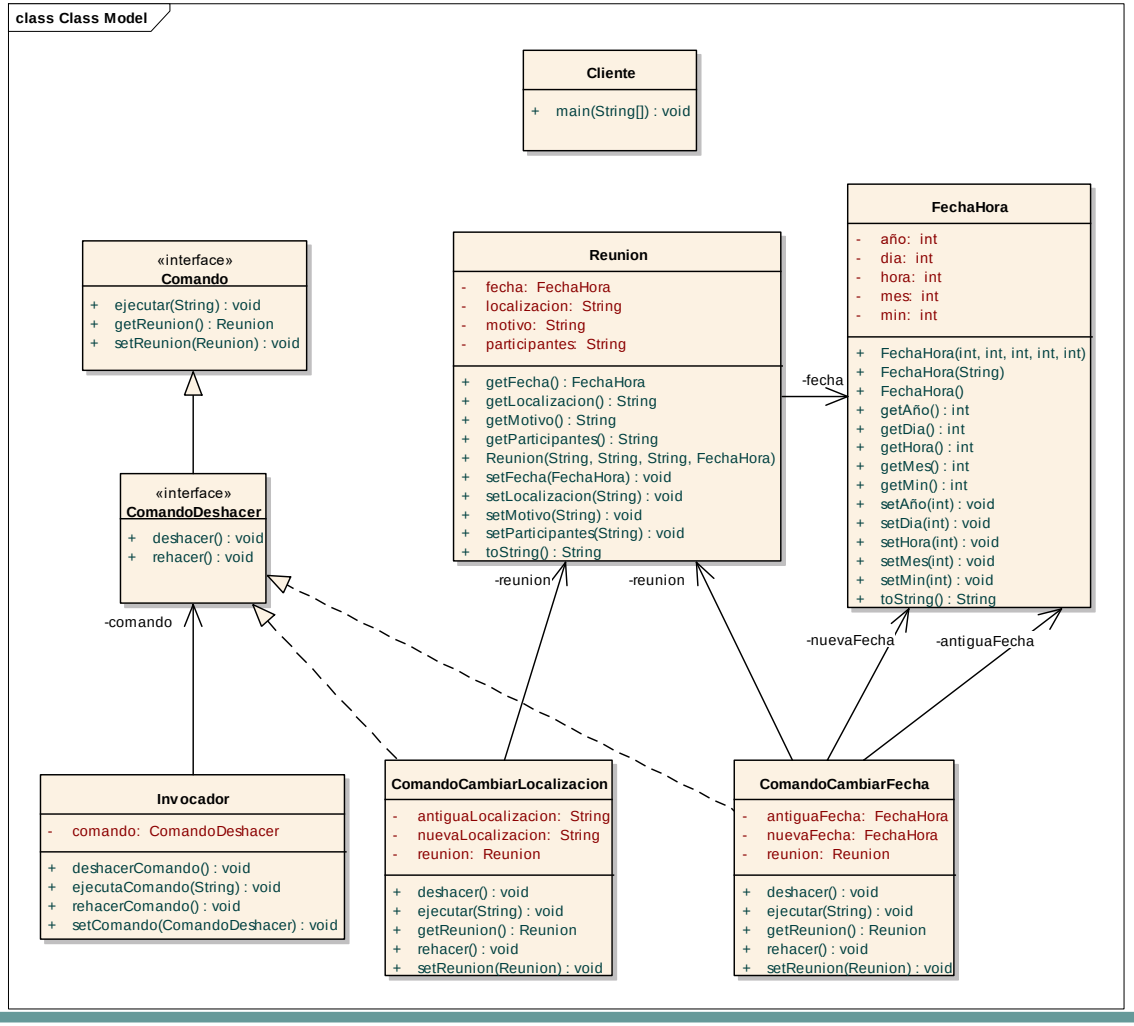
Patrones relacionados

- **Memento:** Puede servir para implementar la función de Deshacer ya que guarda el estado del receptor del comando.
- **Composite:** Permite realizar agrupaciones de comandos de forma similar a una macro.
- **Prototype:** Se puede utilizar para implementar la copia del comando al histórico de comandos.
- **Intérprete:** Podemos implementar un pequeño Intérprete mediante clases *Command*.

Código de ejemplo

Reuniones

Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
  - Orden: ComandoDeshacer.
  - OrdenConcreta: ComandoCambiarLocalizacion, ComandoCambiarFecha.
  - Invocador: Invocador.
  - Receptor: Reunión.
  - Cliente: Cliente.