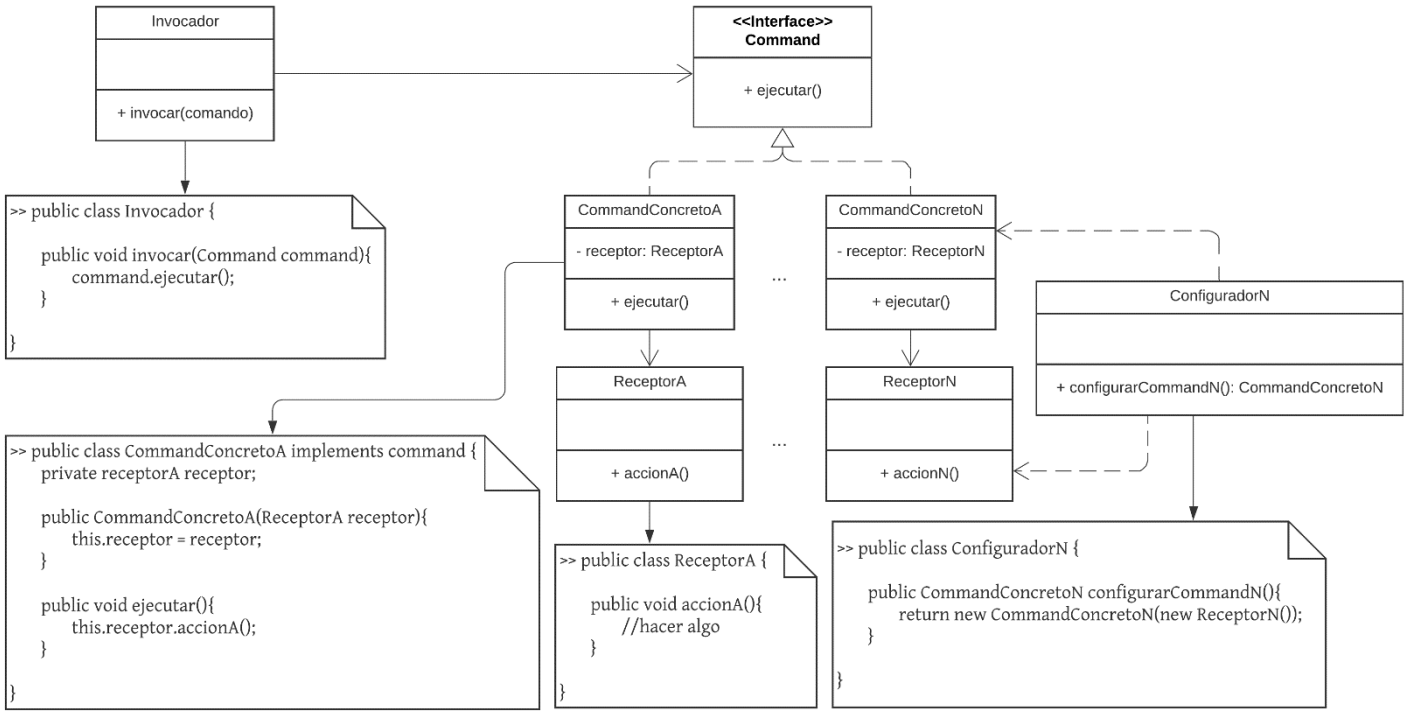
# Patrón Command

- **Clasificación**: Patrón de comportamiento.
- **¿Qué hace?**
  - ✓ Genera una abstracción por cada acción que se quiere realizar, encapsulando cada comportamiento en una clase.
  - ✓ Permite enviar órdenes para que alguien las realice en ese momento o en un momento posterior, sin interesar quién es el receptor de éstas.
- **Se sugiere su utilización cuando**:
  - ✓ Se requiere configurar en momento de ejecución una serie de acciones que debe realizar un objeto.
  - ✓ Se requiere configurar en momento de ejecución una serie de acciones a realizar en un momento posterior al de configuración, formando una cola.
  - ✓ Se requiere seguir adelante con el diseño e implementación sabiendo qué se quiere hacer, pero sin saber exactamente cómo hacerlo o quién lo hará.
- **Estructura genérica**:



*Consideraciones*:

- ✓ El invocador podría tener una colección de acciones (comandos) a ejecutar.
- ✓ El método *ejecutar* del command podría recibir algún parámetro en caso de ser necesario; pero siempre intentando mantener desacopladas al máximo posible las clases Invocadoras de los Comandos.
- ✓ El método *ejecutar* del command podría hacer más de una llamada a la clase receptora, no necesariamente una sola.
- ✓ El Configurador podría no existir si no se lo requiere (si el proceso de instanciación de un comando fuera "sencillo"); o podría existir y tener una lógica más compleja para el armado de un comando específico.
- ✓ No es estricta la relación uno a uno entre comando y receptor, sino que pueden haber varios comandos apuntando al mismo receptor.

- ¿Qué proporciona su uso?
  - ✓ Mayor mantenibilidad debido a que el comportamiento de las acciones es facilmente localizable.
  - ✓ Alta cohesión en los comandos concretos.
  - ✓ Extensibilidad y facilidad para agregar nuevos comandos/acciones.
- Code smells que soluciona/evita de forma directa:
  - ✓ Código duplicado
  - ✓ Clase dios