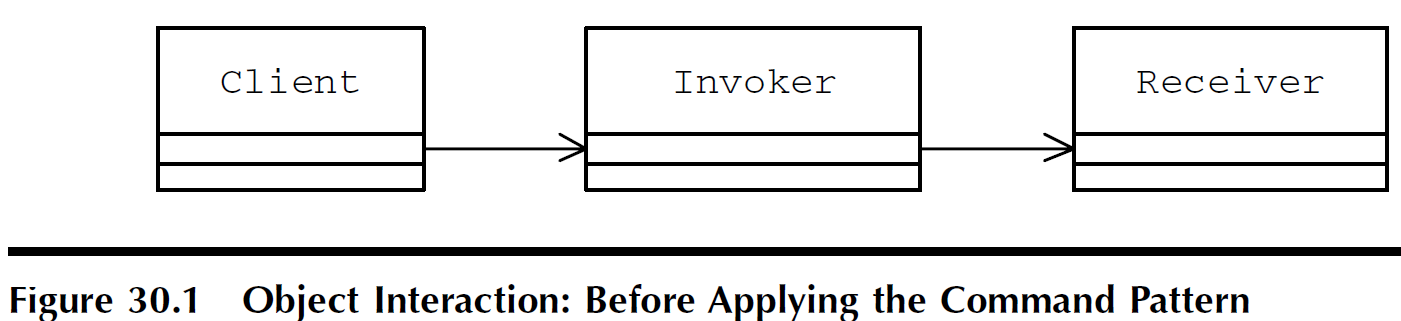
# Patrón Comando

En general, una aplicación orientada a objetos consta de un conjunto de objetos interactivos, cada uno de los cuales ofrece una funcionalidad limitada y enfocada. En respuesta a la interacción del usuario, la aplicación realiza algún tipo de procesamiento. Para este propósito, la aplicación hace uso de los servicios de diferentes objetos para el requisito de procesamiento. En términos de implementación, la aplicación puede depender de un objeto designado que invoca métodos en estos objetos al pasar los datos requeridos como argumentos. Este objeto designado puede denominarse invocador, ya que invoca operaciones en diferentes objetos. El invocador puede ser tratado como parte de la aplicación del cliente. El conjunto de objetos que realmente contienen la implementación para ofrecer los servicios requeridos para el procesamiento de la solicitud se puede denominar objetos de receptor.



En este diseño, la aplicación que reenvía la solicitud y el conjunto de objetos Receiver que ofrecen los servicios necesarios para procesar la solicitud están estrechamente vinculados entre sí, ya que interactúan entre sí directamente. Esto podría resultar en un conjunto de sentencias condicionales si en la implementación del invocador.

Cuando se agrega un nuevo tipo de característica a la aplicación, el código existente necesita ser modificado y viola el principio básico abierto-cerrado. El principio abierto-cerrado establece que un módulo de software debe ser:

* Abierto para extensión: debería ser posible alterar el comportamiento de un módulo o agregar nuevas características para la funcionalidad del módulo.
* Cerrado por modificación: dicho módulo no debe permitir que se modifique su código.

En pocas palabras, el principio abierto-cerrado ayuda a diseñar módulos de software cuya funcionalidad se puede ampliar sin tener que modificar el código existente.

## Objetivo del Patrón Comando

Usando el patrón Command, el invocador que emite una solicitud en nombre del cliente y el conjunto de objetos Receiver que prestan servicios se pueden desacoplar. **El patrón de Comando sugiere crear una abstracción para el procesamiento que se llevará a cabo o la acción que se tomará en respuesta a las solicitudes del cliente.** Esta abstracción puede diseñarse para declarar una interfaz común para ser implementada por diferentes implementadores concretos a los que se hace referencia como objetos Command. Cada objeto Command representa un tipo diferente de solicitud del cliente y el procesamiento correspondiente. En la Figura 30.2, la interfaz de Comando representa la abstracción. Declara un método de ejecución, que es implementado por dos de sus clases de implementador (comando): ConcreteCommand\_1 y ConcreteCommand\_2. Un objeto de comando determinado es responsable de ofrecer la funcionalidad requerida para procesar la solicitud que representa, pero no contiene la implementación real de la funcionalidad. Los objetos de comando hacen uso de los objetos de Receiver para ofrecer esta funcionalidad (Figura 30.3).