Seu objetivo é encapsular uma solicitação como um objeto, o que lhe permite parametrizar outros objetos com diferentes solicitações, enfileirar ou registrar solicitações e implementar recursos de cancelamento de operações. Isso inclui informações como o nome do método, o objeto que o método pertence e os valores dos parâmetros do método.

Command, é uma interface para execução de operações. Na sua forma mais simples, esta interface inclui uma operação, Execute.

As subclasses concretas de Command especificam a recepção através da implementação de Execute para invocar a solicitação.

O receptor tem o conhecimento necessário para poder executar a solicitação, porém externamente os objetos não sabem quais ações estão sendo executadas no receptor, eles apenas visualizam o método execute() que irá executar as suas solicitações. Desta forma, todos os clientes de objetos command tratam cada objeto como uma "caixa preta", simplesmente invocando o método execute() sempre que o cliente exige "serviço" do objeto.

Usar objetos de Command faz com que seja mais fácil construir componentes gerais que precisam delegar, sequenciar ou executar chamadas de métodos em um momento de sua escolha, sem a necessidade de conhecer a classe do método ou os parâmetros do método. Usar um objeto invoker permite contabilizar sobre as execuções de comando a serem realizadas convenientemente, bem como a implementação de diferentes modos para comando, que são geridos pelo objeto invoker, sem a necessidade do cliente estar ciente da existência da contabilidade ou modos.