2018/19

Patrón de diseño Command

Ayad Mercer Laaouissi Jones

Contenido

[Patrón de diseño Command 2](#_Toc4670331)

[Concepto 2](#_Toc4670332)

[Los propósitos son los siguientes: 2](#_Toc4670333)

[Ejemplo mundo real 2](#_Toc4670334)

[Ejemplo computador mundial 2](#_Toc4670335)

[Ilustración 2](#_Toc4670336)

[Ejemplo CommandPatternExample.java 3](#_Toc4670337)

[Ejemplo ModifiedCommandPatternExample.java 5](#_Toc4670338)

[Funcionamiento del patrón de diseño en los ejemplos 8](#_Toc4670339)

# Patrón de diseño Command

Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación. Se encapsula la petición como un objeto.

## Concepto

En general hay cuatro términos invocador, cliente, command y receptor. El objeto command puede invocar un método del receptor de una forma que ha sido especificado a esa clase receptor. El receptor empieza a procesar el trabajo. El objeto command es pasado por separado al objeto invocador para invocar el comando. El objeto cliente sostiene el objeto invocador y el objeto command. El cliente es solo quien realiza la decisión – que comando debe ejecutar – y entonces pasa el comando al objeto invocador (para esa ejecución).

## Los propósitos son los siguientes:

1. Encapsular un mensaje como un objeto, con lo que permite gestionar colas o registro de mensaje y deshacer operaciones.
2. Soportar restaurar el estado a partir de un momento dado.
3. Ofrecer una interfaz común que permita invocar las acciones de forma uniforme y extender el sistema con nuevas acciones de forma más sencilla.

Este patrón presenta una forma sencilla de implementar un sistema basado en comandos facilitando su uso.

## Ejemplo mundo real

Cuando dibujas algo con un lápiz, puedes necesitar deshacer (borrar y rehacer) algunas partes para hacerlo mejor.

## Ejemplo computador mundial

Un ejemplo en el mundo real en el escenario de pintura se aplica en Microsoft Paint. Puedes utilizar el menú o atajos de teclado para rehacer o deshacer en estos contextos. Si deseas realizar una aplicación que soporte deshacer, múltiples deshacer u operaciones similares, entonces este patrón puede ser tu salvador.

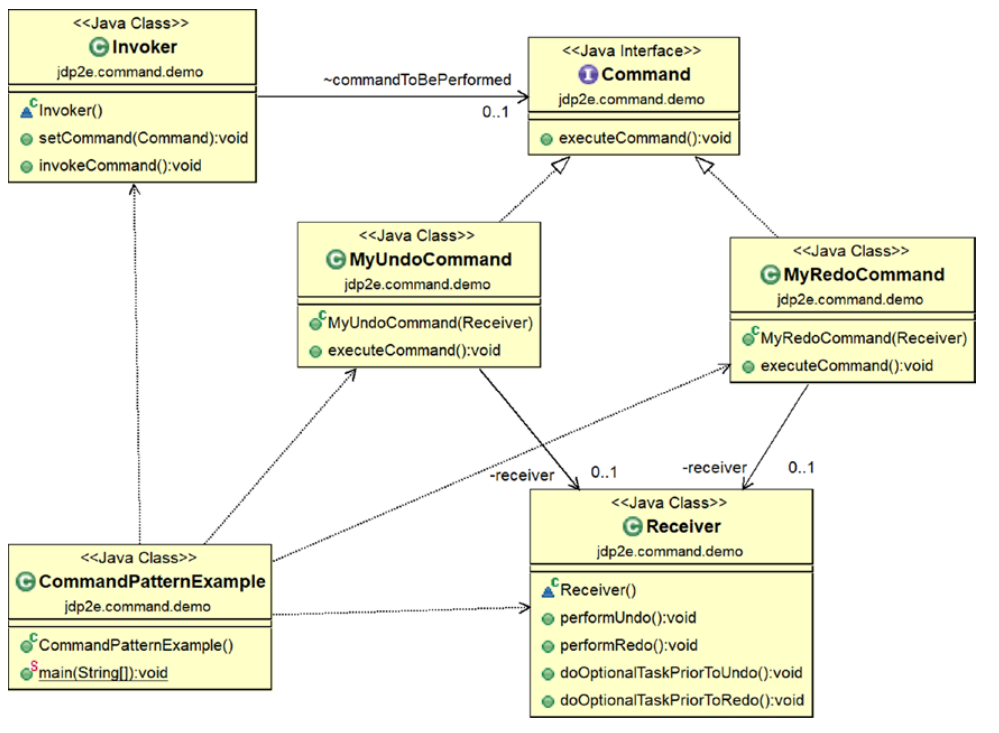
El patrón de diseño command es muy adecuado para manejar GUI interacciones. Funciona tan bien que Microsoft ha integrado en Windows Presentation Foundation (WPF) stack. La pieza más importante es la interfaz ICommand del sistema. Cualquier clase que implemente la interfaz ICommand puede ser usada para controlar eventos del teclado o ratón por los controles comunes WPF. Estos enlaces pueden hacerse mediante XAML o detrás en código.

## Ilustración

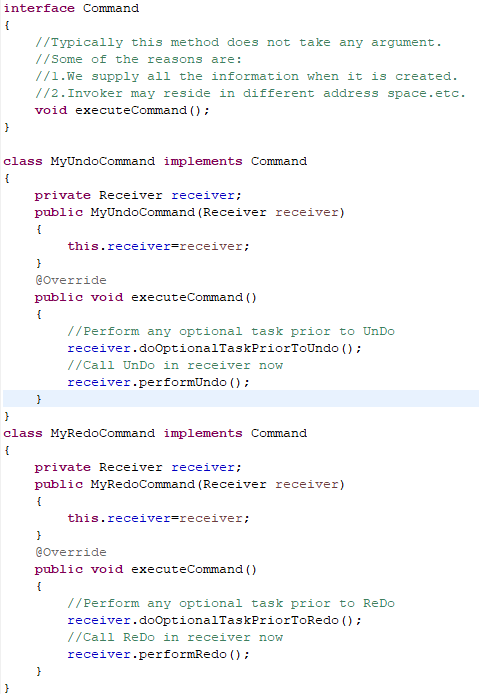
Consideramos el siguiente ejemplo. Indica los mismos nombres que hemos utilizado en nuestra explicación para facilitar el entendimiento a la hora de ver los ejemplos.

## Ejemplo CommandPatternExample.java

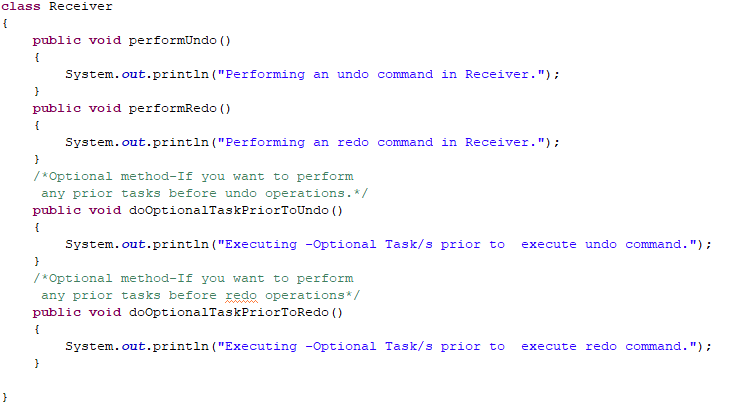
#### En la siguiente figura podemos observar el diagrama de clases.



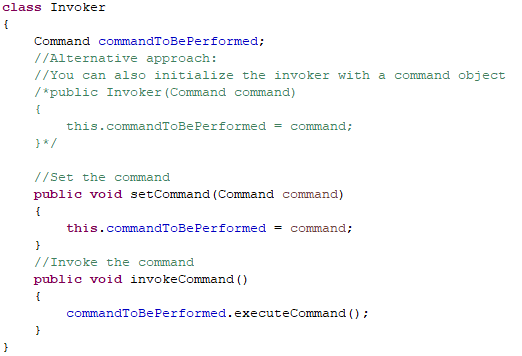
#### Command y sus tipos de comando



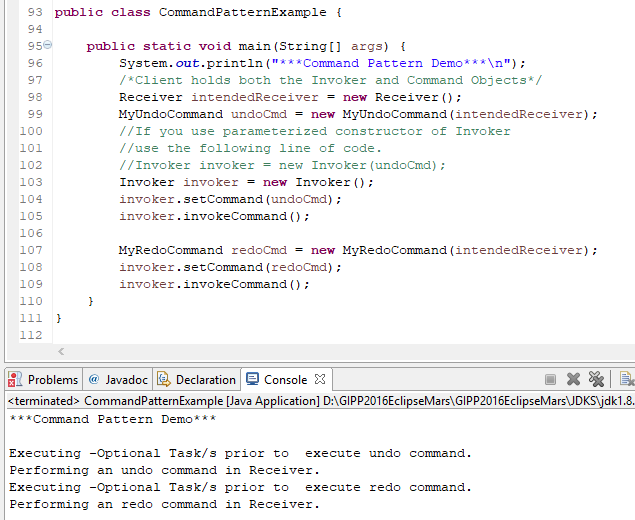
#### Receptor



#### Invocador



#### Salida por consola “Cliente”



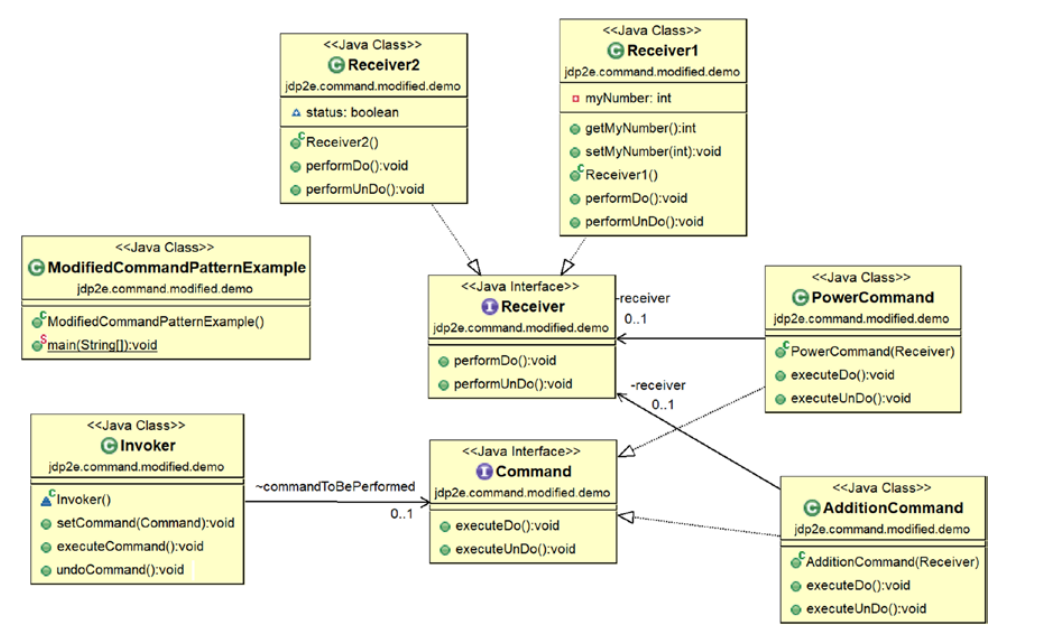
#### Funcionamiento

En este ejemplo utilizamos el patrón de diseño command para ejecutar el contenido de uno de los dos tipos de comandos que le indicamos al invocador. Nuestra interfaz Command ejecuta el contenido sin conocer realmente el contenido de esta operación.

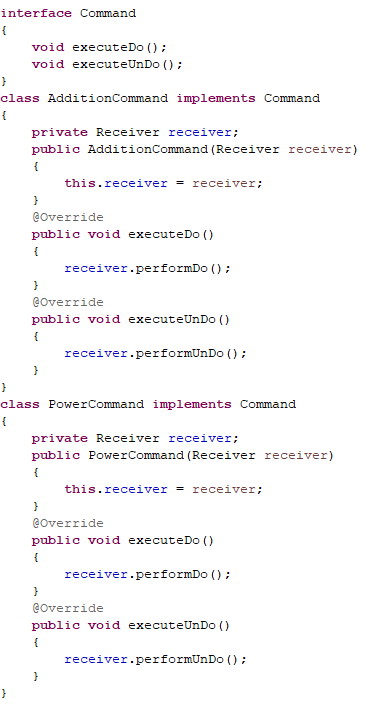
Lo primero que hacemos en este ejemplo es pasar el receptor a uno de los dos tipos de comando (MyUndoCommand o de tipo MyRedoCommand). Al invocador le pasamos este tipo de comando (MyUndoCommand o de tipo MyRedoCommand). El último paso es invocar el comando invokeCommand() del objeto invoker que realizara la acción dependiendo del tipo de comando que le hemos pasado al invocador.

## Ejemplo ModifiedCommandPatternExample.java

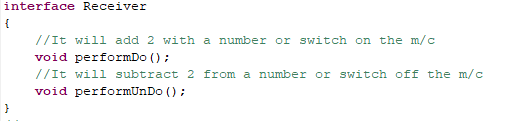
#### En la siguiente figura podemos observar el diagrama de clases.



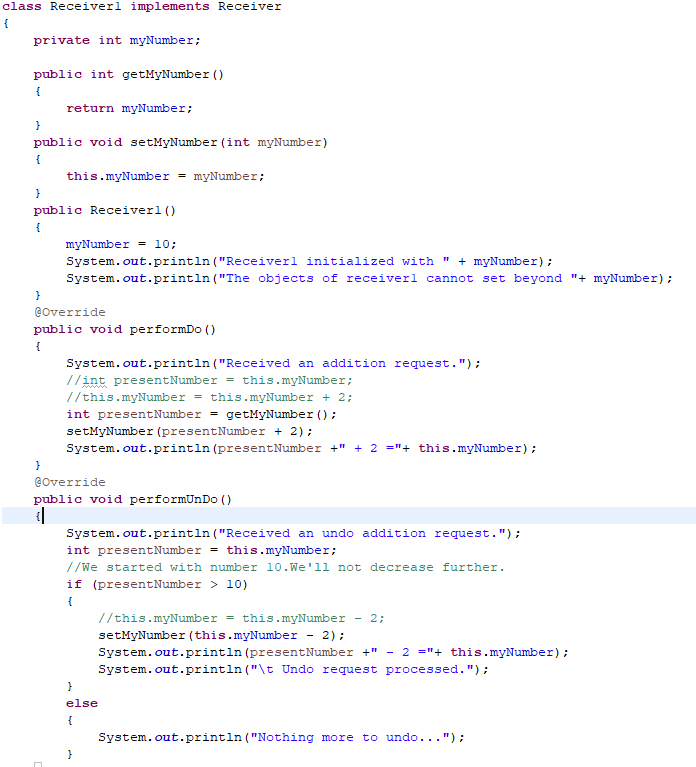
#### Command y sus tipos de comando



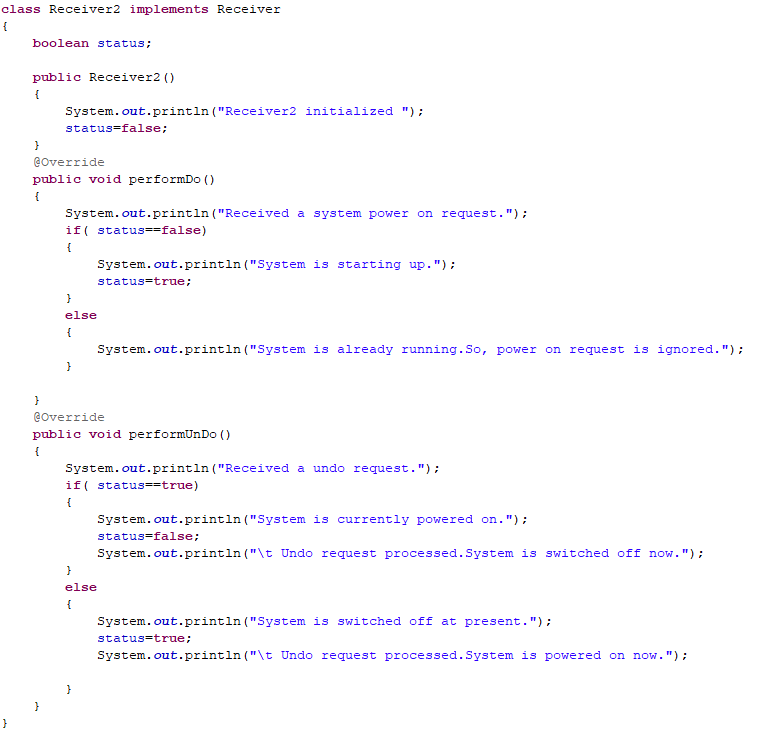
#### Receptor “Interfaz”



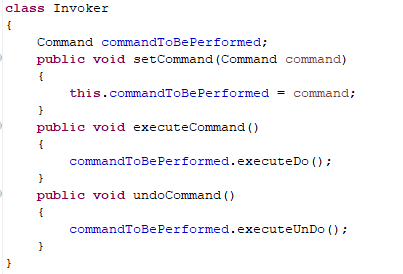
#### Receptor 1



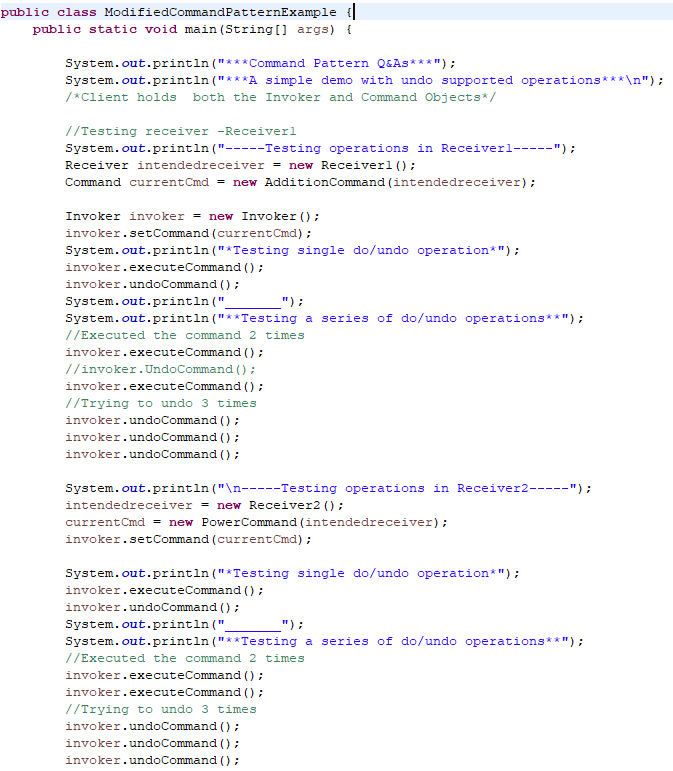
#### Receptor 2



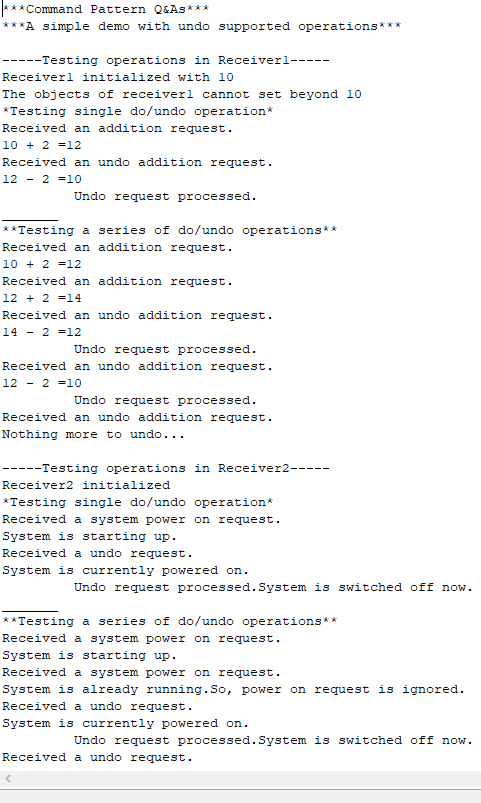
#### Invocador



#### Cliente



#### Salida por consola



#### Funcionamiento

En este ejemplo tenemos dos tipos de receptor que implementan la interfaz receptor. El método executCommand() podra añadir 2 al valor o encender una máquina y el undoCommand() podrá quitar 2 cuando el valor no es menor a 10 o apagar la máquina.

Tiene la misma estructura que en el ejercicio anterior la única diferencia es que en esta tenemos distintos tipos de receptores que al ejecutar el comando va a conseguir hacer una cosa u otra, y al rehacer conseguirá rehacer a su estado anterior. El mismo funcionamiento cuando utilizamos en nuestro ordenador los comandos rehacer a un valor anterior o deshacer.

## Funcionamiento del patrón de diseño en los ejemplos

Los ejemplos anteriores lo podemos conseguir gracias a que indicamos que Command es una interfaz y las clases que implementen esta interfaz deberán escribir sus métodos. De esta manera podemos pasar al invocador cualquier objeto que implementa esta interfaz, nuestro tipo de comando deberá implementa esta interfaz. Al pasar al invocador cualquier objeto que implementa esta interfaz podemos hacer uso de los métodos que tenemos en la interfaz y de este modo se consigue que ejecute contenido Command sin conocer realmente el contenido de esta operación, debido a que el contenido estará en el tipo de comando que hemos pasado al invocador.

La utilización de la interfaz es para poder conseguir multiples comandos o receptores (como en el caso ModifiedCommandPatternExample.java).