EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Oaxaca
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

CARRERA: INGENIERÍA EN SISTEMAS COMPUTACIONALES

Diseño e implementación de software con patrones

Patrón Command

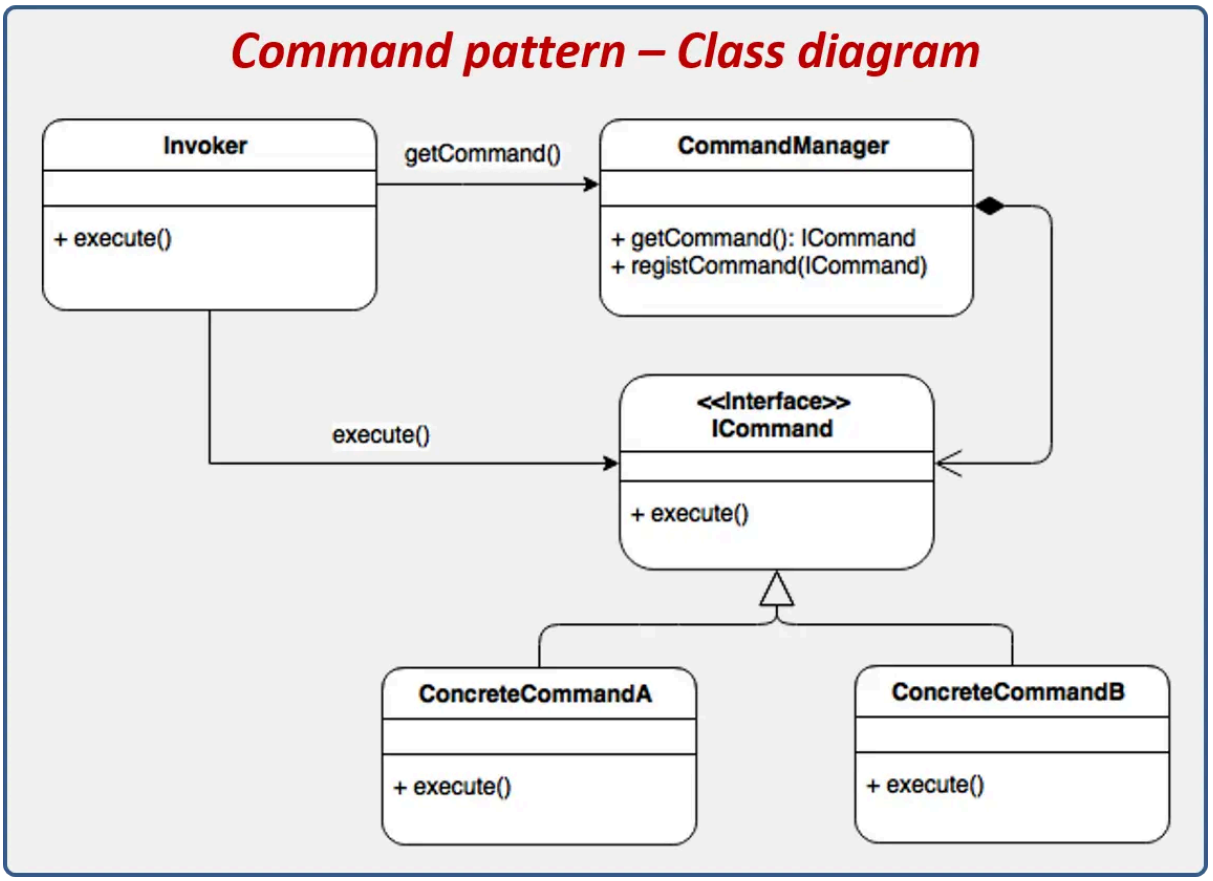
P R E S E N T A:

Giron Pacheco Fernando

Patrón Command

El patrón de diseño Command nos permite ejecutar operaciones sin conocer los detalles de la implementación de la misma. Las operaciones son conocidas como comandos y cada operación es implementada como una clase independiente que realiza una acción muy concreta, para lo cual, puede o no recibir parámetros para realizar su tarea. Una de las ventajas que ofrece este patrón es la de poder crear cuántos comandos requerimos y encapsularlos bajo una interface de ejecución.

Command es un patrón de diseño de comportamiento que convierte una solicitud en un objeto independiente que contiene toda la información sobre la solicitud. Esta transformación te permite parametrizar los métodos con diferentes solicitudes, retrasar o poner en cola la ejecución de una solicitud y soportar operaciones que no se pueden realizar.

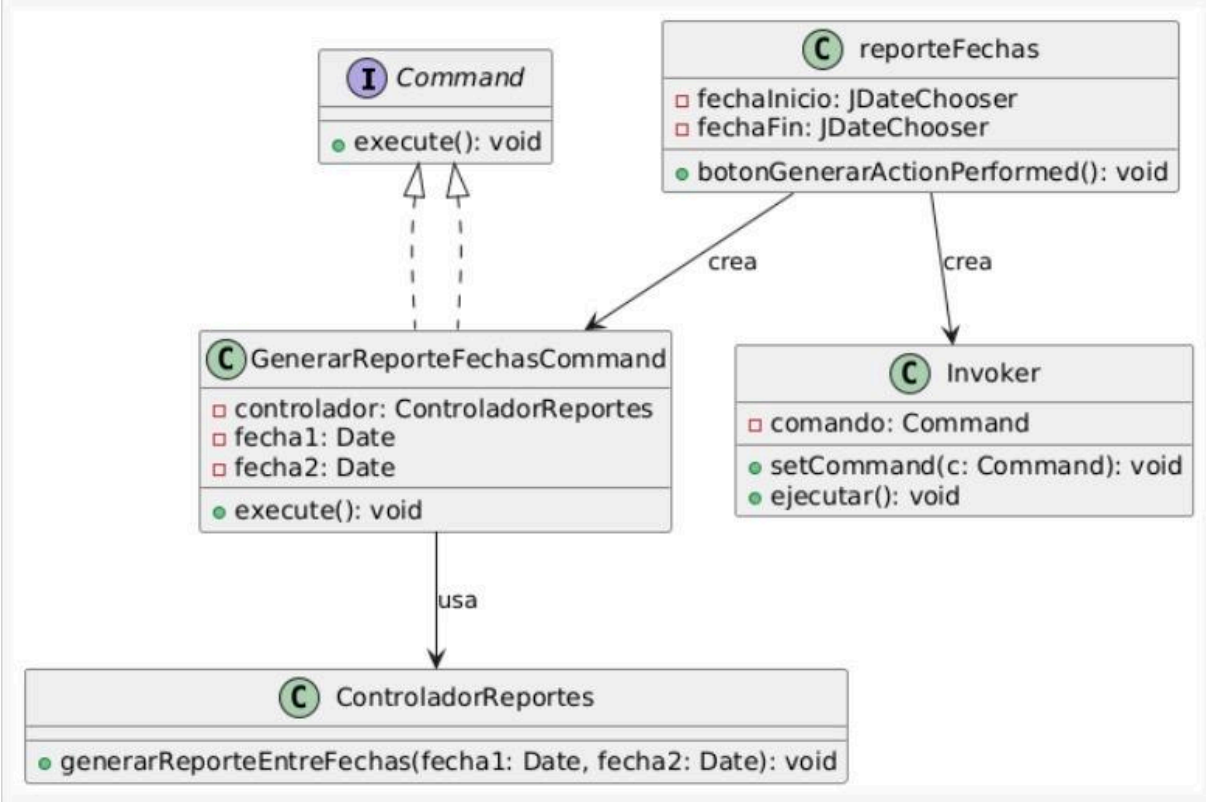


Los elementos del patrón Command se describen a continuación:

- ICommand: Interface que describe la estructura de los comandos, la cual define el método de ejecución genérico para todos los comandos.
- Concrete Command: Representan los comandos concretos, éstos deberán heredar de ICommand. Cada una de estas clases representa un comando que podrá ser ejecutado de forma independiente.

- Command Manager: Este componente nos servirá para administrar los comandos que tenemos disponibles en tiempo de ejecución, desde aquí podemos solicitar los comandos o registrar nuevos.
- Invoker: El invoker representa la acción que dispara alguno de los comandos.

Diagrama UML del patrón



Aplicación en reportes con fechas (JDateChooser)

Imagina que tienes un formulario reporteFechas.java con dos JDateChooser y un botón "Generar Reporte". Normalmente, ese botón llamaría directamente a:

```
controlador.generarReporteEntreFechas(fecha1, fecha2);
```

Paso 1: Crear comando concreto GenerarReporteFechasCommand
package command;

```
import java.util.Date;
import controlador.ControladorReportes;

public class GenerarReporteFechasCommand implements Command {
    private ControladorReportes controlador;
    private Date fechaInicio;
```

```
    private Date fechaFin;

    public GenerarReporteFechasCommand(ControladorReportes
controlador, Date fechaInicio, Date fechaFin) {
        this.controlador = controlador;
        this.fechaInicio = fechaInicio;
        this.fechaFin = fechaFin;
    }

    @Override
    public void execute() {
        controlador.generarReporteEntreFechas(fechaInicio,
fechaFin);
    }
}
```

Paso 2: Usar Invoker en el formulario reporteFechas.java

```
import command.GenerarReporteFechasCommand;
import command.Invoker;

Date fecha1 = fechaInicio.getDate();
Date fecha2 = fechaFin.getDate();

ControladorReportes controlador = new ControladorReportes();
Command comando = new GenerarReporteFechasCommand(controlador,
fecha1, fecha2);

Invoker invoker = new Invoker();
invoker.setCommand(comando);
invoker.ejecutar();
```