<!--Diseño de Sistemas de Información -->

Chain of Responsibility {

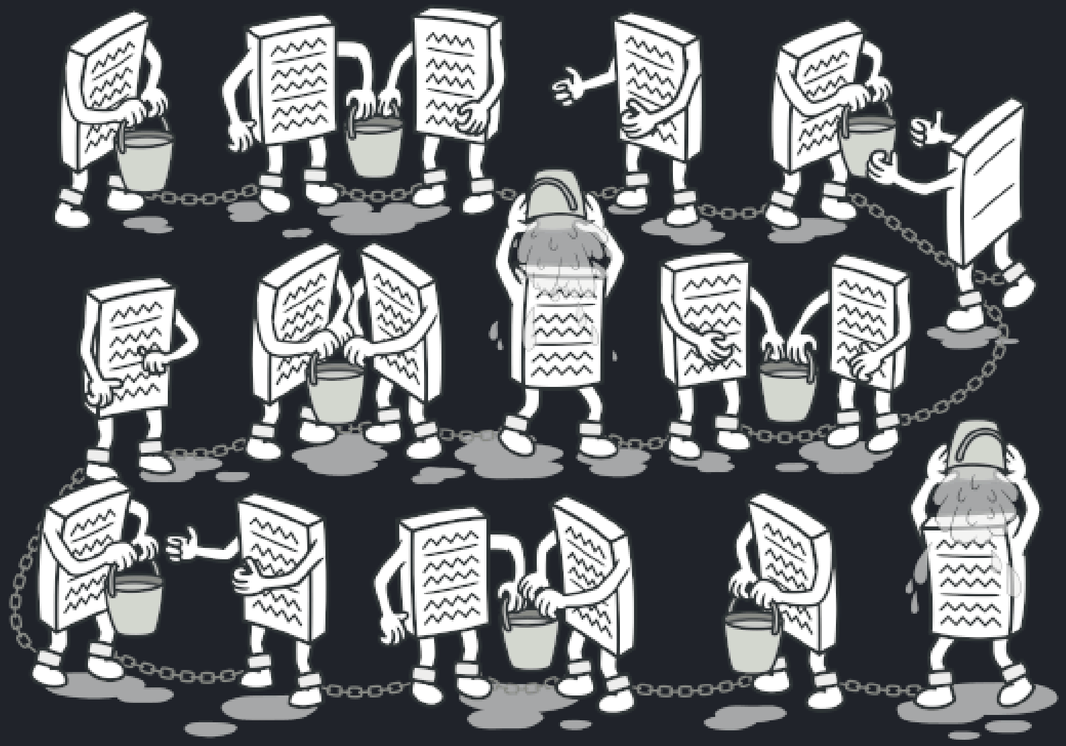
<Patrón de Diseño/>

}

Contenidos

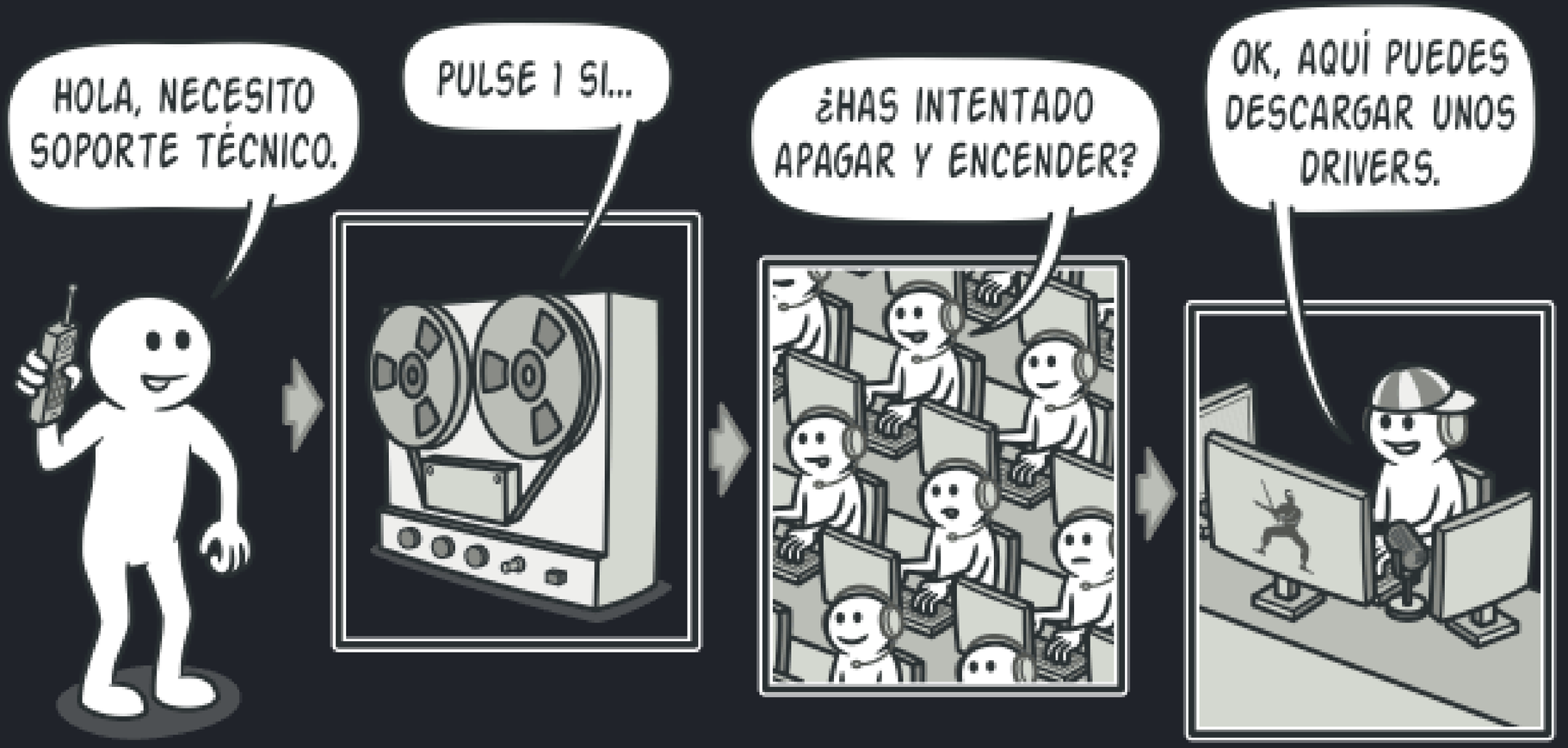
Objetivo {

Permitir que múltiples objetos manejen una solicitud sin que el cliente esté acoplado de manera fija a un receptor específico.



}

Analogía en el mundo real {



}

Aplicabilidad {

01

Cuando debamos procesar distintos tipos de solicitudes, pero los tipos exactos de solicitudes y sus secuencias no se conozcan de antemano.

02

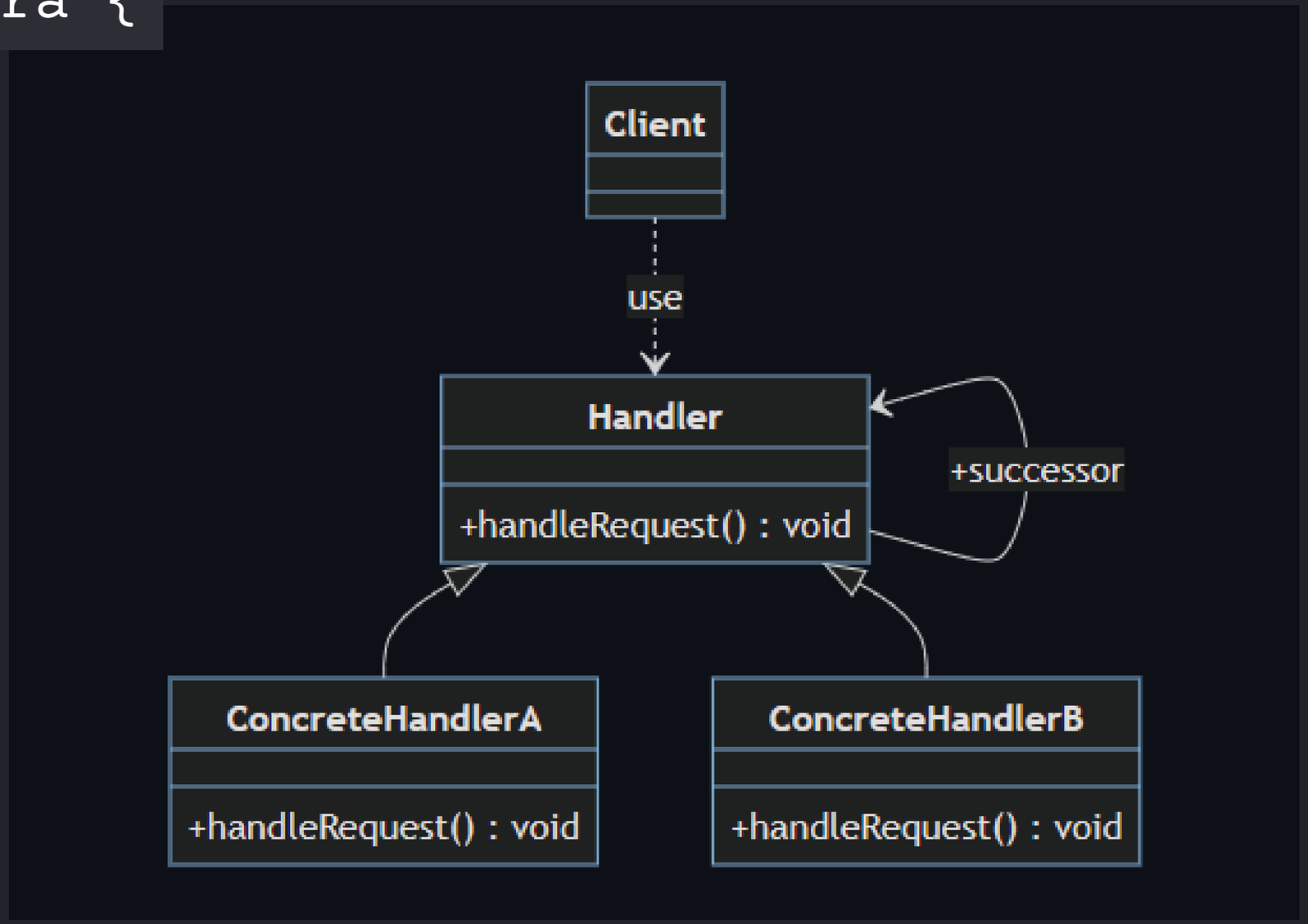
Cuando sea fundamental ejecutar varios manejadores en un orden específico.

03

Cuando el grupo de manejadores y su orden deban cambiar durante el tiempo de ejecución.

}

Estructura {


Client
use
Handler
+handleRequest() : void
+successor
ConcreteHandlerA
+handleRequest() : void
ConcreteHandlerB
+handleRequest() : void

}

Consecuencias {

Ventajas.

- **Se puede controlar el orden de control de solicitudes.**
- **Principio de responsabilidad única.**
- **Principio de abierto/cerrado.**

Desventajas.

- **Algunas solicitudes pueden no procesarse.**

}

El Código {

```
using System;

// Clase para representar una solicitud de vacaciones
7 references
class SolicitudVacaciones
{
    4 references
    public int Dias { get; set; }
    1 reference
    public SolicitudVacaciones(int dias)
    {
        Dias = dias;
    }
}

// Clase abstracta que define el Handler (manejador)
5 references
abstract class Supervisor
{
    5 references
    protected Supervisor? SupervisorSiguiente;

    2 references
    public void EstablecerSiguienteSupervisor(Supervisor supervisor)
    {
        SupervisorSiguiente = supervisor;
    }

    6 references
    public abstract void ProcesarSolicitud(SolicitudVacaciones solicitud);
}
```

```
// Implementación concreta de Supervisor
2 references
class SupervisorJunior : Supervisor
{
    4 references
    public override void ProcesarSolicitud(SolicitudVacaciones solicitud)
    {
        if (solicitud.Dias <= 5)
        {
            Console.WriteLine("Solicitud aprobada por el Supervisor Junior.");
        }
        else if (SupervisorSiguiente != null)
        {
            SupervisorSiguiente.ProcesarSolicitud(solicitud);
        }

    }
}

// Implementación concreta de Supervisor
2 references
class SupervisorSenior : Supervisor
{
    3 references
    public override void ProcesarSolicitud(SolicitudVacaciones solicitud)
    {
        if (solicitud.Dias <= 10)
        {
            Console.WriteLine("Solicitud aprobada por el Supervisor Senior.");
        }
        else if (SupervisorSiguiente != null)
        {
            SupervisorSiguiente.ProcesarSolicitud(solicitud);
        }

    }
}
```

}

El Código {

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        // Configuración de la cadena de responsabilidad
        SupervisorJunior supervisorJunior = new SupervisorJunior();
        SupervisorSenior supervisorSenior = new SupervisorSenior();
        Gerente gerente = new Gerente();

        supervisorJunior.EstablecerSiguienteSupervisor(supervisorSenior);
        supervisorSenior.EstablecerSiguienteSupervisor(gerente);

        // Procesamiento de solicitudes
        SolicitudVacaciones solicitud1 = new SolicitudVacaciones(15);

        supervisorJunior.ProcesarSolicitud(solicitud1);
    }
}
```

}

<!--Diseño de Sistemas de Información -->

Gracias {

<Por="Franco Aguilar"/>

}