Universidad de Guadalajara
Centro Universitario de Los Valles

Materia: Patrones de Diseño y Frameworks

Maestra: Omar Hernández Calvario

ASIGNACIÓN 5

Tema: Patrón Adapter

Fecha: 6 de noviembre de 2024

Presenta: Irais Aldana Llanes

Tabla de contenido

Introducción

En el desarrollo de software, es común que un proyecto incluya componentes de diferentes fuentes o versiones, lo cual puede causar problemas de compatibilidad entre interfaces. El patrón de diseño Adapter surge como una solución a esta situación, actuando como un puente entre interfaces incompatibles y permitiendo que trabajen en conjunto sin necesidad de modificar su estructura original. En este trabajo, exploraremos el concepto y los usos del patrón Adapter, su importancia en la integración de sistemas y cómo se implementa en la programación orientada a objetos.

Desarrollo

Definición y Contexto del Patrón Adapter

El patrón Adapter pertenece a los patrones estructurales de diseño y permite que una interfaz existente sea utilizada como si fuera otra interfaz diferente. Funciona mediante la creación de una clase adaptadora que convierte la interfaz de una clase existente en otra que el cliente espera. Este patrón es especialmente útil en situaciones donde queremos reutilizar clases que, por diversas razones, no pueden ser modificadas directamente para cumplir con nuevas expectativas de funcionalidad.

Tipos de Adapter: Clase y Objeto

Existen dos formas de implementar el patrón Adapter: mediante herencia (Adapter de clase) o composición (Adapter de objeto):

- **Adapter de Clase**: Este enfoque se utiliza en lenguajes que soportan herencia múltiple, como C++. En este tipo de adapter, la clase adaptadora hereda tanto de la clase existente como de la interfaz objetivo, proporcionando la implementación necesaria para adaptar los métodos requeridos.
- **Adapter de Objeto**: Este es el tipo más común y se basa en la composición. La clase adaptadora tiene una referencia a una instancia de la clase existente y traduce las llamadas al formato esperado por el cliente. Este enfoque es flexible y permite adaptarse a cambios sin afectar la estructura interna de las clases existentes.

Implementación de Adapter en Programación Orientada a Objetos

Un ejemplo común del patrón Adapter es la integración de componentes antiguos en una nueva aplicación. Por ejemplo, imaginemos una aplicación de gráficos que necesita usar una clase de dibujo antigua (OldRectangle) en lugar de una clase moderna (NewRectangle) que utiliza coordenadas diferentes para dibujar formas.

```
// Interfaz de destino

interface Shape {

    void draw(int x, int y, int z, int j);

}

// Clase existente, incompatible con Shape

class OldRectangle {

    public void draw(int x, int y, int width, int height) {

        System.out.println("OldRectangle with coordinates: " + x + ", " + y + ", " + width + ", " + height);

    }

}

// Adapter que adapta OldRectangle a la interfaz Shape

class RectangleAdapter implements Shape {

    private OldRectangle adaptee;


    public RectangleAdapter(OldRectangle adaptee) {

        this.adaptee = adaptee;

    }

    public void draw(int x, int y, int z, int j) {

        adaptee.draw(x, y, z, j);

    }

}
```

En el ejemplo anterior la clase RectangleAdapter convierte la interfaz de OldRectangle en la interfaz Shape, lo cual permite a los nuevos componentes de la aplicación interactuar con OldRectangle sin problemas de compatibilidad.

Ventajas y Desventajas del Patrón Adapter

- **Ventajas**: Facilita la reutilización de código existente y la integración de sistemas sin modificar la estructura original de las clases.
- **Desventajas**: La creación de adaptadores puede aumentar la complejidad del sistema, y en algunos casos, puede ser difícil adaptarse a interfaces muy diferentes.

Conclusiones

El patrón de diseño Adapter es fundamental en el desarrollo de software orientado a objetos, permitiendo la integración y reutilización de componentes que, de otra manera, serían incompatibles. Su aplicación es esencial en sistemas que evolucionan constantemente, pues ofrece una solución elegante para extender la funcionalidad de clases sin cambiar su código original. Si bien el patrón Adapter conlleva cierta complejidad, las ventajas en términos de flexibilidad y modularidad superan ampliamente sus limitaciones, haciendo de este patrón una herramienta clave en el diseño de sistemas escalables y mantenibles.