어댑터 패턴 (Adapter Pattern)

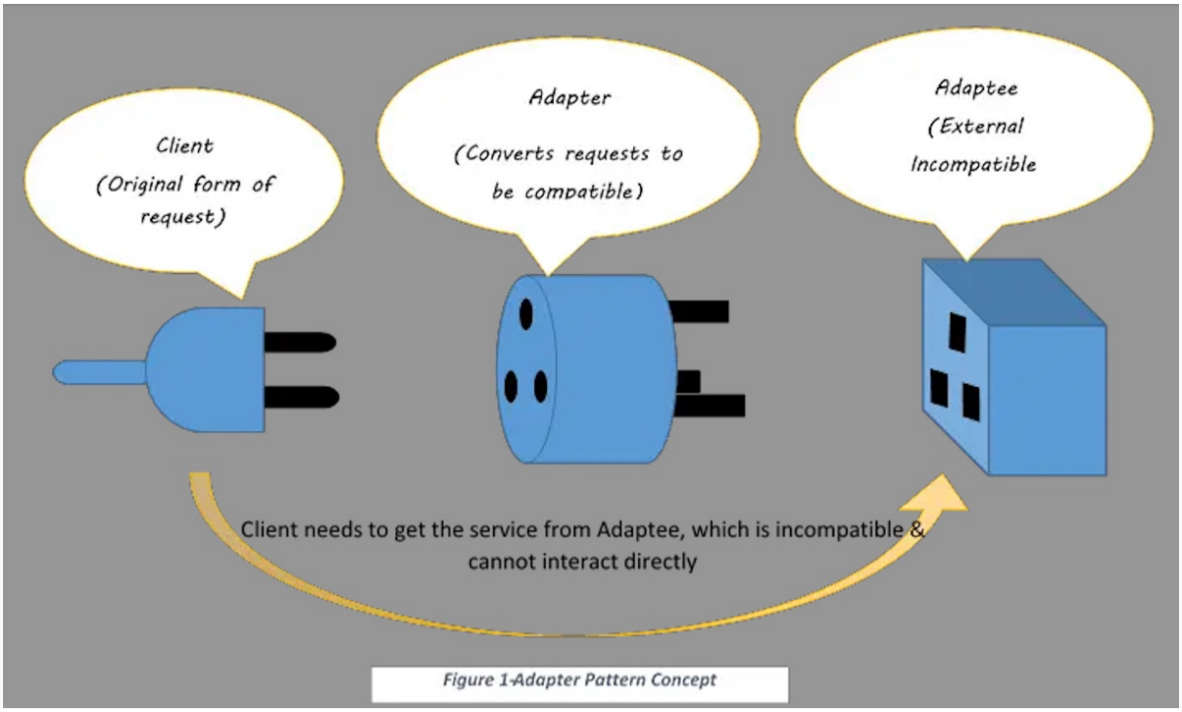

Figure 1-Adapter Pattern Concept

어댑터 패턴의 개념과 역할

어댑터 패턴(Adapter Pattern)은 서로 호환되지 않는 인터페이스를 갖는 두 객체를 연결하는 구조적인 디자인 패턴 중 하나입니다. 즉, 한 클래스의 인터페이스를 클라이언트에서 사용하고자 하는 다른 인터페이스로 변환하는 패턴입니다.

어댑터 패턴은 기존의 코드를 재사용하면서 새로운 인터페이스나 라이브러리를 적용해야 할 때 유용합니다. 기존의 코드가 변경되지 않도록 하면서도 새로운 코드와의 호환성을 보장할 수 있기 때문입니다. 따라서 어댑터 패턴은 소프트웨어 시스템의 유연성을 높일 수 있습니다.

어댑터 패턴의 구성 요소

Target 인터페이스(Target Interface)

어댑터 패턴의 클라이언트가 사용할 인터페이스입니다. 클라이언트는 Target 인터페이스를 통해 어댑터 객체를 사용합니다.

Adaptee

클라이언트가 사용할 수 없는 인터페이스를 가지고 있는 객체입니다. Adaptee 인터페이스는 Target 인터페이스와 다릅니다.

Adapter

Target 인터페이스를 구현하면서 Adaptee 인터페이스를 사용하는 객체입니다. Adapter 객체는 클라이언트로부터의 요청을 Adaptee 객체에게 전달하고, Adaptee 객체로부터의 결과를 다시 클라이언트에게 반환합니다. 즉, Adapter는 Target 인터페이스와 Adaptee 인터페이스 간의 브릿지 역할을 합니다.

클래스 어댑터 패턴과 객체 어댑터 패턴의 차이점

클래스 어댑터 패턴과 객체 어댑터 패턴은 어댑터 패턴의 구현 방식에서 차이가 있습니다.

클래스 어댑터 패턴은 다중 상속을 이용하여 타겟 인터페이스를 구현하고, 어댑터 클래스에서 어댑티의 인터페이스를 상속받아 타겟 인터페이스를 구현합니다. 이 때, 어댑터 클래스에서 어댑티 객체를 생성하여 사용합니다.

반면에 객체 어댑터 패턴은 구성(composition)을 이용하여 타겟 인터페이스를 구현하고, 어댑터 클래스에서 어댑티 객체를 참조하여 타겟 인터페이스를 구현합니다. 이 때, 어댑터 클래스에서 어댑티 객체를 인자로 받아 사용합니다.

클래스 어댑터 패턴은 다중 상속을 지원하지 않는 언어에서는 구현이 어렵습니다. 또한, 어댑터 클래스에서 어댑티 객체를 생성하기 때문에 객체 생성 비용이 추가로 발생할 수 있습니다.

반면에 객체 어댑터 패턴은 구성을 이용하기 때문에 어댑터 클래스에서 어댑티 객체를 참조하기 때문에 객체 생성 비용이 적게 발생합니다. 또한, 어댑터 클래스와 어댑티 객체를 느슨하게 결합할 수 있기 때문에 유연한 구현이 가능합니다.

어댑터 패턴의 장단점

장점

- 호환성 문제를 해결할 수 있습니다. 기존에 존재하는 클래스나 객체를 수정하지 않고도 다른 인터페이스를 제공할 수 있기 때문입니다.
- 유연성과 확장성이 뛰어납니다. 새로운 인터페이스를 지원하도록 어댑터 클래스를 추가하거나, 새로운 어댑터를 만들어 추가할 수 있습니다.
- 코드의 재사용성이 높아집니다. 기존에 존재하는 클래스나 객체를 활용하므로, 코드를 재작성할 필요가 없습니다.

단점

- 어댑터 패턴은 구조를 복잡하게 만들 수 있습니다. 어댑터 클래스를 추가하거나 인터페이스를 맞춰주기 위해 많은 코드가 필요할 수 있습니다.
- 어댑터 클래스의 추가로 인해 성능 문제가 발생할 수 있습니다. 런타임 시 어댑터 객체를 생성하고 호출하는 과정에서 오버헤드가 발생할 수 있습니다.

어댑터 패턴의 구현 방법 및 주의사항

첫째는 클래스 어댑터 패턴이며, 이는 어댑터 클래스가 타겟 인터페이스를 직접 구현하고 어댑티 클래스를 상속받아서 어댑티의 기능을 타겟 인터페이스의 메서드로 호출하는 방식입니다.

둘째는 객체 어댑터 패턴이며, 이는 어댑터 클래스가 타겟 인터페이스를 구현하고 어댑티 클래스의 객체를 멤버 변수로 가지며, 어댑티 클래스의 기능을 타겟 인터페이스의 메서드로 호출하는 방식입니다.

어댑터 패턴을 구현할 때 주의해야 할 점은 다음과 같습니다.

1. 타겟 인터페이스와 어댑티 클래스의 기능이 완전히 일치하지 않는 경우, 즉 어댑티 클래스의 기능을 모두 구현할 수 없는 경우 어댑터 패턴을 사용할 수 없습니다.

2. 클래스 어댑터 패턴을 사용할 때는 어댑터 클래스가 어댑티 클래스의 하위 클래스가 되므로, 어댑티 클래스의 메서드 중에서는 오버라이드 할 수 없는 메서드가 있는 경우 클래스 어댑터 패턴을 사용할 수 없습니다.
3. 객체 어댑터 패턴을 사용할 때는 어댑터 클래스와 어댑티 클래스의 객체 간의 인터페이스가 일치해야 합니다. 이를 위해 어댑터 클래스는 어댑티 클래스의 객체를 참조하는 멤버 변수를 가지고 있어야 합니다.
4. 어댑터 클래스의 구현은 타겟 인터페이스를 구현하는 것이므로, 타겟 인터페이스의 변경이 어댑터 클래스에 영향을 주지 않도록 설계해야 합니다. 즉, 어댑터 클래스는 어댑티 클래스의 기능을 감싸는 역할만 하도록 만들어져야 합니다.