이터레이터 패턴

Tags 디자인 패턴

이터레이터 패턴 (iterator pattern)

이터레이터 패턴은 일련의 데이터 집합에 대하여 순차적인 접근(순회)을 지원하는 패턴이다.

데이터 집합이란 객체들을 그룹으로 묶어 자료의 구조를 취하는 컬렉션을 말한다. 대표적인 컬렉션으로 한번쯤은 들어본 리스트나 트리, 그래프, 테이블 등이 있다.

보통 배열이나 리스트 같은 경우 순서가 연속적인 데이터 집합이기 때문에 간단한 for문을 통해 순회할수 있다. 그러나 해시, 트리와 같은 컬렉션은 데이터 저장 순서가 정해지지 않고 적재되기 때문에, 각 요소들을 어떤 기준으로 접근해야 할지 애매해진다.

예를 들어 트리 구조에서 어떤 상황에선 깊이(세로)를 우선으로 순회 해야 할 수도 있고, 너비(가로)를 우선으로 순회할수도 있기 때문이다.

이처럼 복잡하게 얽혀있는 자료 컬렉션들을 순회하는 알고리즘 전략을 정의하는 것을 이터레이터 패턴이라고 한다.

컬렉션 객체 안에 들어있는 모든 원소들에 대한 접근 방식이 공통화 되어 있다면 어떤 종류의 컬렉션에서도 이터레이터만 뽑아내면 여러 전략으로 순회가 가능해 보다 다형(多形) 적인 코드를 설계할 수 있게 된다.

이밖에도 이터레이터 패턴은 별도의 이터레이터 객체를 반환 받아 이를 이용해 순회하기 때문에, 집합체의 내부 구조를 노출하지 않고 순회 할 수 있다는 장점도 있다.

> 💡 자바의 컬렉션 프레임워크(JCF)에서 각종 컬렉션을 무리없이 순회할수 있는 것도 내부에 미리 이터레이터 패턴이 적용되어 있기 때문이다.

사용 시기

- 컬렉션에 상관없이 객체 접근 순회 방식을 통일하고자 할 때
- 컬렉션을 순회하는 다양한 방법을 지원하고 싶을 때
- 컬렉션의 복잡한 내부 구조를 클라이언트로 부터 숨기고 싶은 경우

이터레이터 패턴의 장점

- 일관된 이터레이터 인터페이스를 사용해 여러 형태의 컬렉션에 대해 동일한 순회 방법을 제공한다.
- 컬렉션의 내부 구조 및 순회 방식을 알지 않아도 된다.
- 집합체의 구현과 접근하는 처리 부분을 반복자 객체로 분리해 결합도를 줄일 수 있다.
- 순회 알고리즘을 별도의 반복자 객체에 추출하여 각 클래스의 책임을 분리하여 단일 책임 원칙(SRP)을 준수한다.
- 데이터 저장 컬렉션 종류가 변경되어도 클라이언트 구현 코드는 손상되지 않아 수정에는 닫혀 있어 개방 폐쇄 원칙(OCP)을 준수한다.

이터레이터 패턴의 단점

- 클래스가 늘어나고 복잡도가 증가한다.
  - 만일 앱이 간단한 컬렉션에서만 작동하는 경우 패턴을 적용하는 것은 복잡도만 증가할 수 있다.
  - 이터레이터 객체를 만드는 것이 유용한 상황인지 판단할 필요가 있다.
- 구현 방법에 따라 캡슐화를 위배할 수 있다.

Reference

https://inpa.tistory.com/entry/GOF-💠-반복자Iterator-패턴-완벽-마스터하기