Iterator Pattern

결론 및 정리

정의

Iterator Pattern은 Behavioral Pattern 중 하나로, **컬렉션의 요소에 접근하고 순회하는 방법을 제공하면서, 그 컬렉션의 기본 구조(Tree, List ... 등) 를 노출하지 않는 패턴**.

- **Iterator**: 요소에 순차적으로 접근하는 인터페이스를 정의. 주요 메서드로는 `next()`, `hasNext()` 가 존재. 해당 메서드는 순회의 방법을 제공 / 제시.
- **ConcreteIterator**: Iterator 인터페이스를 구현하며, 순회 로직을 실제로 구현.

구현

1. 컬렉션에 대한 Iterator 인터페이스를 정의한다.
2. 해당 컬렉션의 ConcreteIterator를 구현한다.
3. Iterator Interface 타입의 변수를 선언 후, Concrete Iterator 를 안에 집어 넣는다. 만약 Iterator 변수를 생성하는 Class 를 따로 만들어 주고자 하는 경우, 해당 Class 를 Aggregate Class 라고 한다.

Context & Problem & Merit

- 다양한 컬렉션 구조에 대해 일관된 방법으로 요소를 순회하고자 할 때 Iterator Pattern을 사용.
- 컬렉션의 내부 구조를 노출하지 않고 요소에 접근하고 싶을 때 유용.

1. **추상화**: 컬렉션의 내부 구조와 독립적으로 요소를 순회할 수 있음.
2. **단일 책임 원칙**: 순회 로직이 컬렉션에서 분리되므로, 각 클래스는 자신의 주요 책임에만 집중.
3. **확장성**: 새로운 컬렉션 타입이 추가되더라도 기존 코드에 영향을 주지 않고, 해당 컬렉션에 대한 새로운 Iterator만 구현하면 됨.

강의 들으며

강의 노트 보며 :
Data 를 저장하는 다른 Collection 구조가 굉장히 많음. 이 각각의 구조에 대한 순회를 돌 때, 각각에 대해 다른 순회 코드가 존재함.
이에 따라 순회 (Iteration) 을 더 간단히 돌기 위해 만들어진 순회자 (Iterator) 가 바로 이것.

모든 순회를 포괄 할 수 있는 Iterator Interface 를 만들어 둠. 이 Interface 는 hasNext() / next() method 를 선언하여 둠으로써 다른 class 들이 사용할 수 있도록 함.

해당 interface 를 구현하는 다른 Concrete Iterator 는 변수들의 묶음을 자신의 뜻대로 가지고 있을 수 있는 Collection 변수 하나와, 종류에 따라 해당 Collection 을 순회하는 Iterator 의 method 를 정의함.