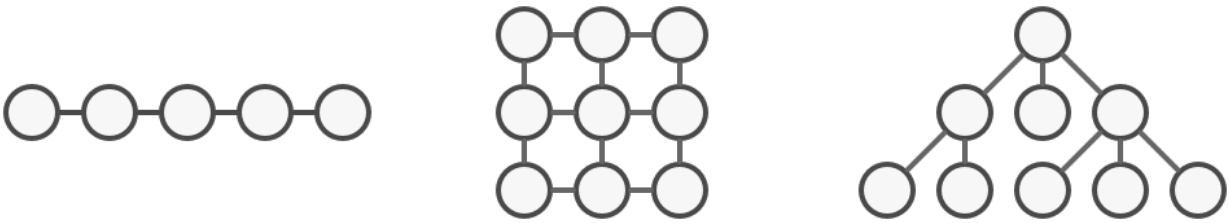
참고 자료

- https://refactoring.guru/design-patterns/iterator

정의

- behavioral design pattern
  - object의 behavior을 캡슐화하고 요청을 object에 위임하는 패턴
- collection의 자료구조를 모른 채 element들을 순회할 수 있는 디자인 패턴

상황 - 다양한 타입의 collection 순회

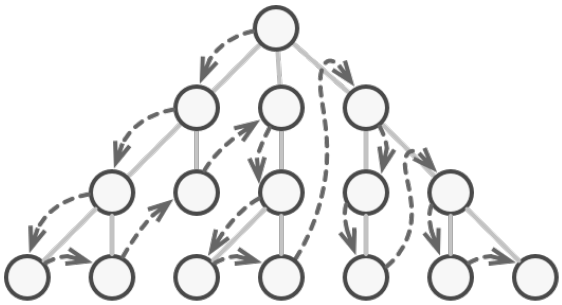


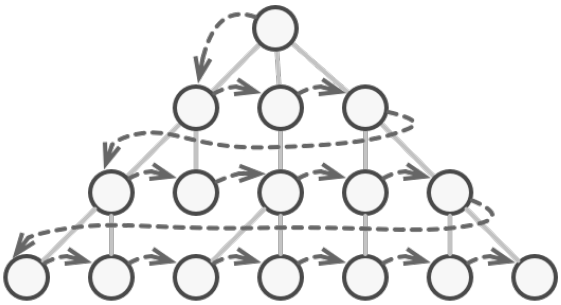
- 다양한 타입의 collection을 순회해야 합니다.

안좋은 접근방법 - collection에 순회알고리즘 구현

문제점

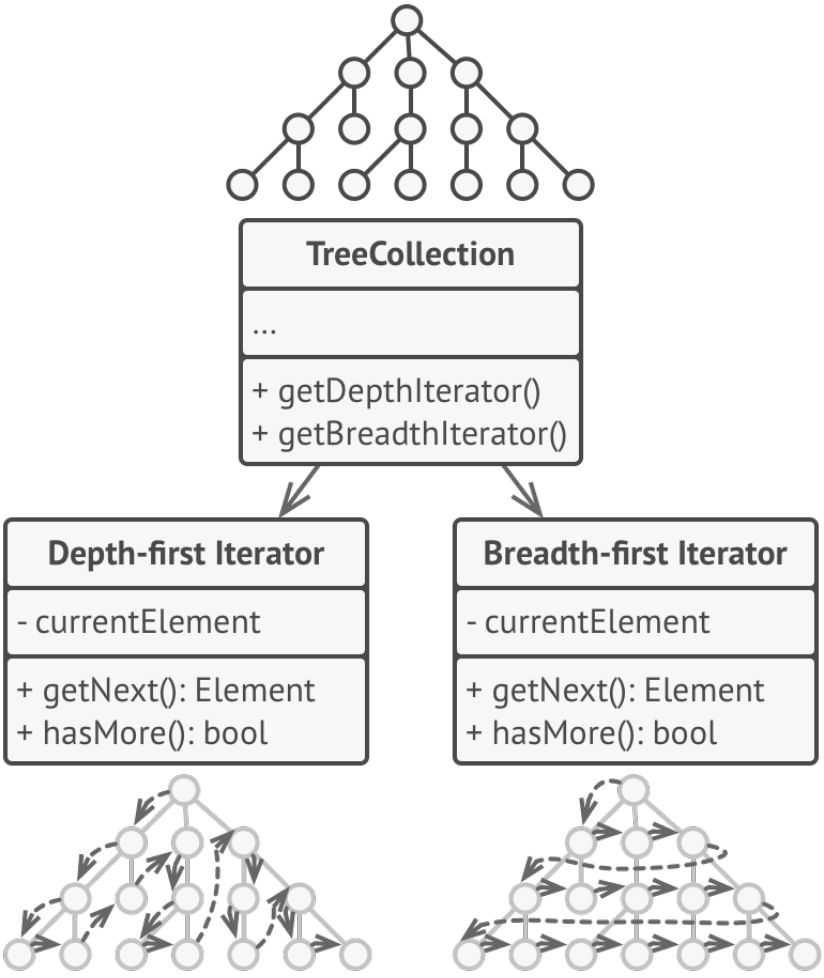
1. 복잡한 자료구조일 경우 순회가 어렵고, 여러 종류가 있을 수 있습니다.
   - 트리의 경우를 보면 DFS 또는 BFS로도 또는 랜덤하게 순회할 수 있는데 이 모든 순회 알고리즘을 collection에서 구현한다면 지저분(blur)해질 수 있습니다. 효율적인 저장한다는 collection의 주요기능이 있는데 말이죠.
2. 클라이언트가 코드가 특정한 collection에 의존하게 됩니다.
   - 클라이언트는 어떻게 저장되는지 관심없음에도 불구하고 각 collection을 통해서 자신의 element에 접근하는 다른 방법을 제공할것이므로 collection에 의존하게 됩니다.





해결방법 - 순회 알고리즘을 분리하라!

- iterator pattern의 핵심은 collection에서 순회 알고리즘을 iterator라는 object로 분리하는 것에 있습니다.



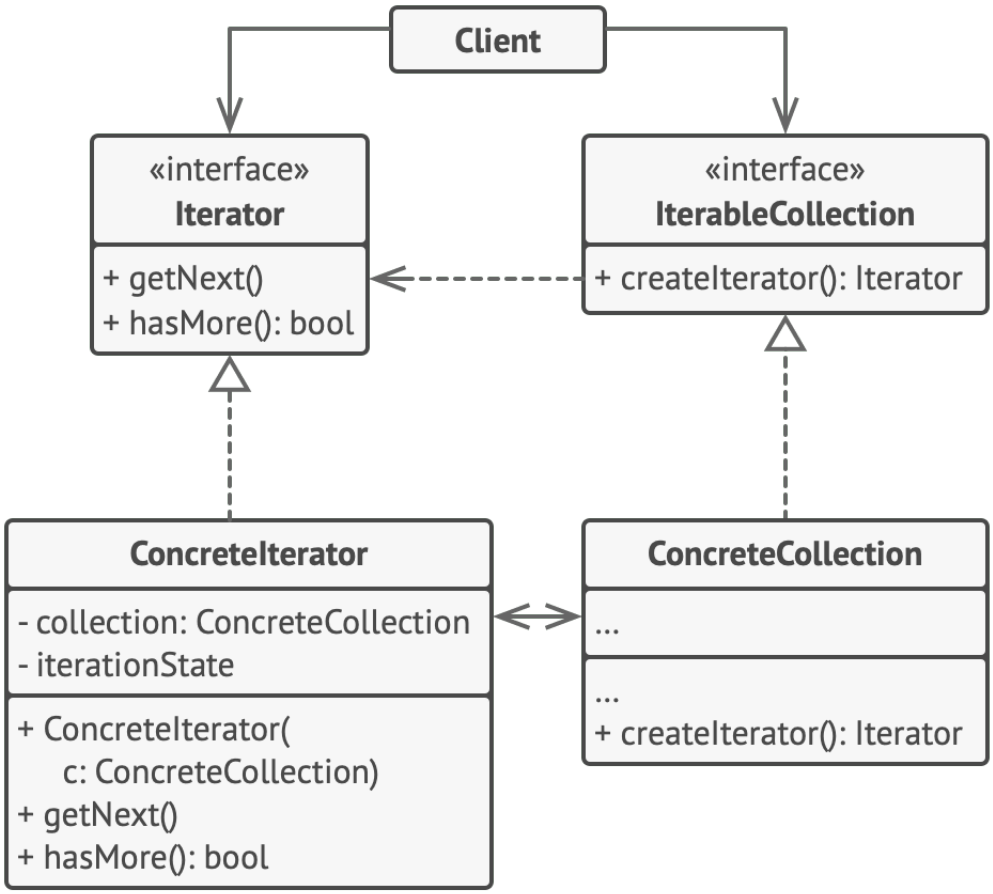
구현 방법 및 이점

1. 순회하는 부분을 iterator라는 class로 분리합니다.
    - traversal에 대한 세부정보를 숨길(capsulate) 수 있습니다.
        - 예를 들면 다음과 같습니다. 현재 position, element가 얼마나 남았는지를 숨길 수 있죠.
    - 순회하는 책임을 iterator에게 둘 수 있습니다.

2. element를 받을(fetch) 수 있는 주요 method를 구현합니다.
    - 이것을 통해 모든 element를 순회할 수 있습니다.

3. 모든 iterator가 동일한 interface를 구현하도록 합니다.
    - collection과 순회 알고리즘을 구분하지 않고 client코드가 사용할 수 있게 됩니다.

구조



1. Iterator

- collection을 순회할 때 필요한 알고리즘을 구현합니다.
    1. 다음 element를 fetching하기
    2. 현재 위치를 알아내기
    3. iteration을 재시작하기 등등

2. Concrete Iterators

- 순회하기 위한 특정 알고리즘을 구현합니다.
- iterator가 자기 자신의 진행을 추적하게 합니다.

  이렇게 함으로써 각 iterator가 동일한 collection에 대해 서로 독립적으로 순회할 수 있게 됩니다.

3. Collection

- iterator를 얻기 위한 한 개 또는 여러 개의 method를 선언합니다.

4. ConcreteCollection

- client가 요청할 때마다 새로운 iterator instance를 반환하도록 합니다.

5. Client

- client는 collection과 iterator를 interface를 통해 사용(works)합니다.
- client가 iterator를 생성하지 않고 collection이 생성하도록 합니다.
  - 하지만 특별한 경우엔 client가 생성할 수 있습니다. 예를 들어 클라이언트가 자신만의 특별한 Iterator를 생성하고 싶을 때 말이죠.

의문

collection Interface를 통해 타입이든지 상관 않는다고 하지만... collection이 tree, list 경우처럼 순회에 차이가 있을 경우 LSP위반하게 됩니다.

tree는 dfs, bfs와 같은 이름으로 iterator를 반환하지만 list에 이와 같은 iterator는 없습니다. 즉, LSP위반이라 볼 수 있죠.

따라서... 모두 동일한 collection을 두기보다는 tree, list...등등 lsp가 적용되는 collection끼리 묶어서 interface를 생성하는 것이 더 맞지 않나하네요