ADT & DS

💡 ADT (Abstract Data Type)

- 추상자료형
- 개념적으로 어떤 동작이 있는지만 정의
- how에 대해서는 다루지 않음. (구현에 대해서는 다루지 않음)
- 자료 구조의 특징, 속성, Operations 정의
- ex) Stack, Queue, List, Tree ...
- 예를 들어 Stack ADT는 push, pop, isEmpty 등의 연산을 정의하지만, 이러한 연산이 내부적으로 어떻게 구현되는지는 정의하지 않음.

💡 DS (Data Structure)

- 자료구조
- ADT에서 정의된 동작을 실제로 구현한 것
- 구현체가 있어야 함.
- ex) Array, LinkedLIst, HashTable, Heap, Graph ...
- 데이터가 메모리에 어떻게 저장되고 관리될지를 구체적으로 정의함.

📌 ADT 와 DS의 차이점

- 추상화 수준
  - ADT는 데이터 구조의 논리적 특성을 정의하는데 초점을 맞추며, 구현 세부사항은 숨김.
  - DS는 데이터가 실제로 어떻게 저장되고 관리되는지에 대한 구체적인 세부 사항을 다룸.
- 구현과 인터페이스
  - ADT는 인터페이스에 중점을 두고 구현 세부사항은 고려하지 않음.
  - DS는 실제 메모리 내에서 데이터가 어떻게 구현되는지를 포함함.
- 사용예시
  - ADT는 설계 시 데이터의 타입와 연산에 대한 명세에 집중함.
  - DS는 프로그램의 성능을 최적화하기 위해 적절한 저장 및 관리 방식을 고려하는데 사용.

✔ 실무와 어떻게 연결될 수 있을까 ?

- 월드컵에서 골을 넣은 선수를 관리하는 자료구조를 만들어 보자.

- ADT

- Alphabetical Ordered (이름순) 라는 이름의 자료 구조를 정의.
- Operations
  - `add(personName)`
  - `getAllPlayers()` : 알파벳 순서대로 반환
  - `remove(personName)`

- DS

- 위 ADT를 도출 후, 실제로 자료구조를 만들어야 하는 사람이 구체적으로 위의 기능을 수행할 DS를 만듦.
- ADT를 실제로 구현하기 위해서 어떤 방법을 사용하는지.
- ex) 알파벳순으로 정렬하기 위해서 `sort()` 메서드를 사용한다. / 중복 제거를 위해서 `set()` 을 사용한다...
- 실무의 업무 분담 및 모듈화 상향

✔ feat. Java

💡 Interface 와 Class를 ADT와 DS와 완전 동일하다고 볼 수 있는가 ?

> *class = DS (🔺)*
> *Interface = ADT (⭕)*

- Interface → ADT와 유사
  - 인터페이스는 Java 에서 ADT와 유사한 개념을 제공함.
  - ADT의 정의
    - ADT는 데이터 타입의 추상적인 정의와 그 데이터 타입에 대해 수행할 수 있는 연산들의 집합을 제공. (단, 구현 세부사항은 제공하지 않음)
  - 자바 인터페이스
    - 인터페이스는 메서드의 시그니처(이름, 매개변수, 반환 타입)를 정의하지만, 이 메서드들이 실제로 어떻게 구현될지에 대해서는 정의하지 않음.
    - 클래스가 이 인터페이스를 구현하도록 하여 실제 메서드의 구체적인 동작을 정의함.
- Class → DS와 유사하지만 완전히 동일하지는 않음
  - 클래스는 Java에서 DS와 비슷한 역할을 함.
  - DS의 정의
    - 데이터 구조는 데이터의 실제 저장 방식과 데이터를 조작하는 연산의 구현을 포함.
  - 자바 클래스

- 클래스는 데이터 구조의 구현을 위한 수단을 제공하며, 데이터(필드)와 데이터를 조작하는 연산(메서드)을 묶어 표현함.
- 클래스는 데이터 구조의 구체적인 구현체가 될 수 있지만, 모든 클래스가 데이터 구조를 나타내는 것은 아님.

💡 결론

- ADT와 DS는 밀접하게 관련이 있으며, 종종 함께 사용된다.
- ADT는 어떤 데이터 타입이 수행해야 할 연산을 정의하는 반면, DS는 이러한 ADT가 실제로 메모리 상에서 어떻게 구현될지를 결정한다.
- 위 개념을 잘 알면 설계에 도움이 된다.