[읽을거리] 1. 자료구조와 알고리즘

컴퓨터 프로그래밍

우리는 프로그램을 통해서 컴퓨터에게 시키고 싶은 일들을 동작 시킬 수 있습니다. 이 프로그램을 만드는 과정을 프로그래밍 혹은 코딩 이라고 합니다. 컴퓨터가 이해하고 동작할 수 있도록 하기 위해서 우리는 컴퓨터가 동작할 수 있는 방식으로 프로그램밍을 해야 합니다.

우리는 자바언어를 통해서 프로그래밍을 할 예정입니다. 언어만 알면 프로그래밍을 할 수 있을까요? 물론 할 수 있습니다. 하지만 우리가 컴퓨터를 통해서 하고 싶은 일들을 더 효율적이고 잘 수행되게 하려면 컴퓨터가 일을 처리하는데 유리한 방식으로 프로그래밍 하는 것이 더 좋을 것입니다.

그래서 컴퓨터가 처리할 데이터를 어떻게 구성하는 지와 어떻게 처리하는 지를 이해하는 것이 필요합니다. 여기서 데이터를 어떻게 구성하는가에 대한 것이 자료구조라고 하는 것이구요. 어떻게 처리하는지에 대한 것이 알고리즘이라고 이해하시면 되겠습니다.

자료구조

우리가 컴퓨터를 활용해서 일을 처리하는 이유는 복잡한 계산이나 반복되는 일을 빠르게 처리하기 위해서 입니다. 그래서 자료구조를 얘기할 때는 기본적으로 많은 양의 데이터를 취급하는 것을 전제로 합니다. 그래서 많은 데이터를 어떻게 구성할 것인가 하는 것이 자료구조의 기본입니다.

많은 양의 데이터를 구성하는 방식은 크게 두가지로 나뉩니다. 순서대로 한 줄로 나열할 것이냐, 아니면 연관성 있는 것끼리 묶어서 관리할 것인가 하는 것입니다. 자료구조에서는 이것을 선형 자료구조, 비선형 자료구조 라고 부릅니다.

선형 혹은 비선형으로 데이터가 구성되어 있을 때 우리는 이 데이터에 4가지 일을 수행할 수 있습니다.

1. 새로운 데이터 추가하기 (Create)
2. 저장된 데이터 읽어오기 (Read)
3. 새로운 데이터로 변경하기 (Update)
4. 불필요한 데이터 삭제하기 (Delete)

이 각각의 일을들 Create, Read, Update, Delete 작업이라고 하고, 줄여서 CRUD 라고 쓰고 *크루드*라고 부릅니다.

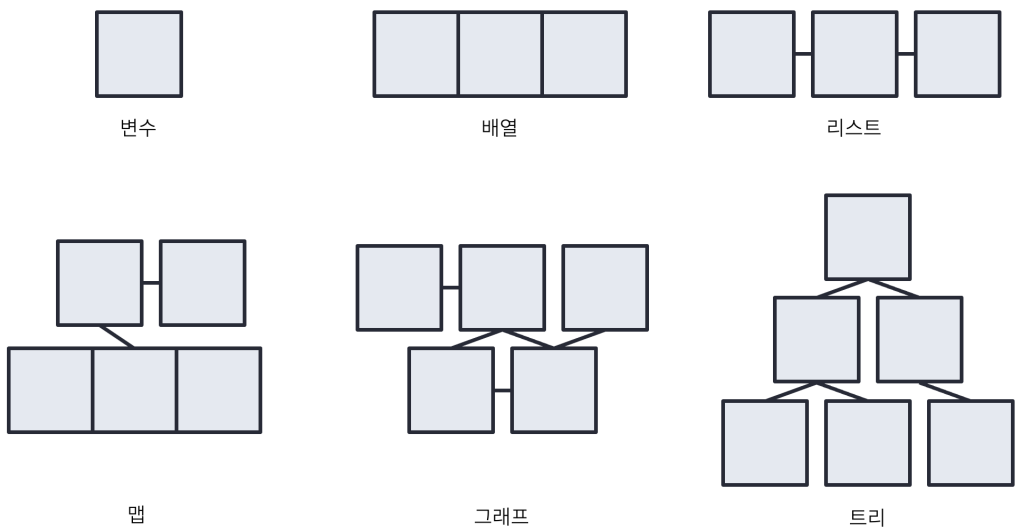
알고리즘

자료구조는 데이터가 구성되는 형태에 대한 얘기 입니다. 이 데이터에 CRUD작업을 수행하기 위해서 처리하는 다양한 방법들이 있습니다. 이러한 처리 방법들을 알고리즘 이라고 합니다.

기초적인 알고리즘에는 데이터의 순서를 바꾸는 방법, 원하는 데이터를 찾아내는 방법, 추가하거나 읽을 때 순서를 정하는 방법등 다양한 알고리즘들이 존재 합니다. 예를들어 데이터의 순서를 바꾸는 것을 Sort 한다라고 하는데 이 Sort 알고리즘에도 Bubble-Sort, Insert-Sort, Merge-Sort, Quick-Sort 등등 여러 다양한 방법들이 존재 합니다. 그리고 지금도 컴퓨터과학을 연구하시는 분들이 계속해서 새로운 알고리즘들을 만들어내고 있습니다.

즉 알고리즘이라고 하는 것은 데이터를 다루는 방법이고, 그 방법은 한 가지만 있는 것이 아니라 아주 다양하고 많은 방법들이 있습니다.

자료구조 오버뷰



Java Data Type

컴퓨터는 CPU의 종류에 따라서 데이터를 표현하는 크기가 다릅니다. 예를들어 32bit 컴퓨터에서 int 의 크기는 4bytes 이고, 64bit 컴퓨터에서 int의 크기는 8bytes 입니다.

JAVA는 JVM(Java Virtual Machine) 에서 수행되면서 CPU에 관계없이 JVM이 정해놓은 크기로 데이터를 사용합니다. 그래서 어떤 컴퓨터에서든지 Java 로 작성된 프로그램은 일관성있게 실행될 수 있습니다.

Mutable / Immutable

변경할 수 있는 값을 mutable, 변경할 수 없는 값을 immutable 이라고 합니다.

final이 붙은 primitive 값은 변경될 수 없으니 immutable 값이 됩니다. 하지만 final이 붙은 참조(reference) 변수의 경우 변경되지 못하기는 하지만 변경되지 못하는 값은 변수가 담고 있는 참조 주소값이 됩니다. 즉, 참조 변수가 가리키는 인스턴스를 변경할 수는 없다는 뜻입니다. 그렇다고 그것이 변수가 가리키는 인스턴스의 내용을 변경할 수 없다는 것을 의미하지는 않습니다.

Immutable Reference

string literal로 생성되는 String 객체의 경우 immutable 값입니다. 따라서 String의 내용이 변경되는 경우 기존의 값이 변경되는 것이 아니라 새로운 인스턴스가 생성되게 됩니다. 그리고 같은 내용의 문자열은 기존의 생성되었던 인스턴스를 재사용하게 되므로 같은 인스턴스를 가리키게 됩니다.

```
String str1 = "Hello";
String str2 = str1 + " World";  // 새로운 인스턴스가 만들어지고 str2가 참조하게 됨
System.out.println(str1 == str2);  // false

String str3 = "Hello";
System.out.println(str1 == str3);  // true
```

일부 Wrapper 클래스는 마치 Immutable 객체처럼 동작합니다.

```
Integer i1 = 42;
Integer i2 = 42;
System.out.println(i1 == i2);  // true
```

하지만, 일정 범위의 값에 대한 인스턴스를 미리 만들어두고 재사용하는 것으로, 그 범위를 벗어나면 객체 재사용을 하지 못하게 됩니다.

```
Integer i3 = 100000;
Integer i4 = 100000;
System.out.println(i3 == i4);  // false
```

Reference Address

참조객체의 참조 주소를 출력하고 싶을 때는 `System.identityHashCode` 메소드를 사용합니다.

```
Integer i = 42;
System.out.println("Integer@" + System.identityHashCode(i));  // Integer@366712642
i = 43;
System.out.println("Integer@" + System.identityHashCode(i));  // Integer@1829164700
```

시간복잡도

알고리즘의 성능을 측정하는 5가지 기준

1. 정확성 : 동일한 입력에 대해 항상 동일한 결과를 반환하는가
2. 작업량 : 얼마나 적은 연산으로 결과를 만들어 내는가 (`시간복잡도`)
3. 메모리 사용량 : 얼마나 적은 메모리를 사용하여 결과를 만들어 내는가 (`공간복잡도`)
4. 단순성 : 얼마나 단순한 구현으로 결과를 만들어 내는가
5. 최적성 : (컴퓨터의 동작 기준으로) 더 이상 개선할 여지가 없을 만큼 최적화 되어 있는가

시간복잡도의 표현 및 순서

- Big O 표현식 ex) $O(n)$, $O(n^2)$

복잡도 함수	분류 이름	대분류
1	상수 시간	다항 시간
$\lg n$	로그 시간	
n	선형 시간	
$n \lg n$	선형-로그 시간	
n^2	2차 시간	
n^3	3차 시간	
2^n	지수 시간	지수 시간
$n!$	팩토리얼 시간