[읽을거리] 3. 맵

시간복잡도

Map의 종류별 시간복잡도는 다음과 같습니다.

	get()	containsKey()	next()
HashMap	O(1)	O(1)	O(h/n)
LinkedHashMap	O(1)	O(1)	O(1)
IdentityHashMap	O(1)	O(1)	O(h/n)
WeakHashMap	O(1)	O(1)	O(h/n)
EnumMap	O(1)	O(1)	O(1)
TreeMap	O(log n)	O(log n)	O(log n)
ConcurrentHashMap	O(1)	O(1)	O(h/n)
ConcurrentSkipListMap	O(log n)	O(log n)	O(1)

일반적으로 Map은 O(1) 로 알려져 있지만, 극단적으로 Map에 저장하는 Key가 모두 충돌(collision)하는 경우 LinkedList를 사용하는 Map일 때 최대 O(n)의 시간복잡도를 갖게 될 것입니다. 이를 해결하기 위해 충돌되는 값들을 TreeNode 에 저장할 경우 O(log n) 시간 내에 처리가 가능합니다.

생성과 함께 초기값 지정하기

잘 알려지지 않은 java 문법 중 double brace initialize 문법이 있습니다.

보통 맵을 만들고 값을 추가할 때 이렇게 사용합니다.

```
Map<String, Integer> map = new HashMap<>();
map.put("A", 1);
map.put("B", 2);
map.put("C", 3);
```

double brace initialize 문법을 사용하면 다음과 같이 생성과 함께 초기값을 지정할 수 있습니다.

```
Map<String, Integer> map = new HashMap<>() {{
    put("A", 1);
    put("B", 2);
    put("C", 3);
}};
```

double brace initialize 문법을 사용하면 다음의 장점이 있습니다.

- 기존 방식에 비해 코드가 줄어듭니다.
- 코드가 더 읽기 쉽습니다
- 동일한 식(expression)에서 생성과 함께 초기화가 수행됩니다.

하지만 다음과 같은 단점도 있다는 것을 유의하여 사용해야 합니다.

- 모호하고 널리 알려지지 않은 문법
- 사용할 때마다 추가 클래스가 생깁니다.
- 상속된 클래스가 final로 선언된 경우 동작하지 못합니다.
- 숨겨진 참조(reference)를 보유하고 있어 메모리 누수를 유발할 가능성이 있습니다.

이러한 단점들 때문에 안티패턴으로 간주되기도 합니다.

맵의 모든 값 순회하기

entrySet 사용

map의 entrySet 을 사용하면 key, value 쌍의 목록을 얻을 수 있습니다.

```
Map<String, Integer> map = new HashMap<>() {{
    put("A", 1);
    put("B", 2);
    put("C", 3);
}};
for (Map.Entry<String, Integer> entry : map.entrySet()) {
    System.out.println(entry.getKey() + ":" + entry.getValue());
}
```

keySet 사용

keySet을 사용해서 key의 목록을 얻고, 이를 통해 value를 얻어 사용합니다.

```
Map<String, Integer> map = new HashMap<>() {{
    put("A", 1);
```

```
    put("B", 2);
    put("C", 3);
}};
for (String key : map.keySet()) {
    System.out.println(key + ":" + map.get(key));
}
```

데이터 양이 많을 경우 key를 통해 value를 찾는 과정이 추가되므로, entrySet을 사용하는 것이 효율적일 수 있습니다.

Iterator 사용

entrySet 이나 keySet 모두 Set 으로 제공되고 Collection에서 제공되는 Iterator를 사용할 수 있습니다.

```
Map<String, Integer> map = new HashMap<>() {{
    put("A", 1);
    put("B", 2);
    put("C", 3);
}};
Iterator<Map.Entry<String, Integer>> iter = map.entrySet().iterator();
while (iter.hasNext()) {
    var entry = iter.next();  // 타입추론
    System.out.println(entry.getKey() + ":" + entry.getValue());
}
```

java 10 이상을 사용한다면 `Iterator<Map.Entry<String, Integer>>` 같은 복잡한 타입을 사용하지 않아도 `var` 를 사용해 타입추론된 변수를 사용할 수 있습니다.

```
var iter = map.entrySet().iterator();  // 타입추론
```