회고록

기본타입→객체타입(언박싱,박싱)

https://inpa.tistory.com/entry/JAVA-%E2%98%95-wrapper-class-Boxing-UnBoxing

이터레이터, 이뉴멀에이트

모든객체는 object객체를 상속받음

제네릭이란 이터러블 객체에 어떤 타입이 들어간다 라는 타입을 지정해주는 것

제네릭을 쓰지 않고 쓰면 object가 기본으로 들어가는데, 만약 object로 하면 모든 타입이 들어가서 사용자가 사용할 때 전부 object타입으로 되브로, 항상 형변형을 시켜줘야하는 불편함이 있어서 제네릭을 넣어주면 그 안에 뭐가 들어가는지 정의할 수 있어서 간단함

java의 map이 파이썬의 딕셔너리와 비슷한가?

Collection이란?

Map단어가 들어간 java 자료구조 뭐가있고 특징이 무엇이고, 어떤 특징 때문에 어떻게 사용하는가?

nullpointer exception

set,list,map 에 데이터넣고 조회하는거까지 할 줄알아야한다(학습목표)

1. 기본 타입 → 객체 타입 (박싱, 언박싱)

- **기본 타입 (Primitive)**: `int`, `double`, `char` ... (8개)
- **래퍼 클래스 (Wrapper Class)**: `Integer`, `Double`, `Character` ...
  → 기본형을 객체로 감싸는 클래스

개념

- **박싱(Boxing)**: 기본타입 → 객체타입 변환

```
int a = 10;
Integer boxed = Integer.valueOf(a); // 박싱
```

- **언박싱(Unboxing)**: 객체타입 → 기본타입 변환

```
Integer obj = 20;
int b = obj.intValue(); // 언박싱
```

- **오토 박싱/언박싱**: 자바 5부터 자동 변환 지원

```
Integer x = 100;  // 오토 박싱
int y = x;        // 오토 언박싱
```

언제 쓰나?

- 제네릭/컬렉션은 객체만 다루므로 `int` 같은 기본 타입은 바로 못 담음 → `Integer`로 변환 필요.

---

2. 이터레이터(Iterator), 이뉴머레이션(Enumeration)

- **Iterator**
  - `Collection`에서 요소를 순회할 때 사용
  - `hasNext()`, `next()`, `remove()` 메서드 제공

```
List<String> list = List.of("A", "B", "C");
Iterator<String> it = list.iterator();
while (it.hasNext()) {
    System.out.println(it.next());
}
```

- **Enumeration**

- 자바 초창기 컬렉션(Vector, Hashtable)에서 사용되던 인터페이스
- `hasMoreElements()`, `nextElement()` 제공
- 요즘은 거의 `Iterator`로 대체됨

---

3. 모든 객체는 Object를 상속받음

- 자바에서 **모든 클래스는 암묵적으로 `java.lang.Object`를 상속**합니다.
- Object가 제공하는 대표 메서드:
  - `toString()` : 객체 문자열 표현
  - `equals()` : 동등성 비교
  - `hashCode()` : 해시값 반환
  - `clone()`, `finalize()`, `wait()`, `notify()` 등

---

4. 제네릭(Generic)

개념

- **컬렉션/클래스/메서드에 들어가는 타입을 외부에서 지정**할 수 있도록 해줌.
- 타입 안전성 + 형변환 코드 제거.

예시

```
List<Object> list = new ArrayList<>(); // 제네릭 미사용: 모든 타입 가능, 꺼낼 때 캐스팅 필요
list.add("Hello");
String s = (String) list.get(0); // 캐스팅 필요

List<String> strList = new ArrayList<>(); // 제네릭 사용
strList.add("Hello");
String s2 = strList.get(0); // 캐스팅 불필요
```

핵심: 제네릭 없으면 내부적으로 전부 Object로 처리 → 항상 캐스팅해야 해서 불편/위험.

제네릭 쓰면 **타입 안정성** 확보.

---

5. Java의 Map = Python의 Dictionary?

맞습니다!

- **Map**: Key → Value 저장, Key 중복 X, Value 중복 O
- 파이썬의 `dict` 와 같은 역할.

예:

```
Map<String, Integer> map = new HashMap<>();
map.put("apple", 3);
map.put("banana", 5);

System.out.println(map.get("apple")); // 3
```

---

6. Collection이란?

- 자바에서 **데이터 그룹(집합)을 표현하는 인터페이스**
- `Collection` 인터페이스를 중심으로 **List, Set, Queue** 제공
- `Map`은 Collection을 직접 상속하진 않지만 컬렉션 프레임워크에 포함됨.

---

7. Map 관련 자료구조와 특징

- **HashMap**
  - Key/Value 저장, 순서 없음, 해시 기반 빠른 검색
- **LinkedHashMap**
  - 입력 순서를 유지
- **TreeMap**
  - Key를 자동 정렬 (Red-Black Tree 기반)
- **ConcurrentHashMap**
  - 멀티스레드 환경에서 안전

---

8. NullPointerException

- **null 참조에 대해 메서드/필드 접근 시 발생**

```
String s = null;
System.out.println(s.length()); // NullPointerException
```

예방 방법

- `null` 체크 (`if (s != null) ...`)
- Optional 사용
- @NonNull 어노테이션/도구 활용

---

9. Set, List, Map 사용 예시 (학습목표)

```
import java.util.*;

public class Demo {
    public static void main(String[] args) {
        // List: 순서 O, 중복 O
        List<String> list = new ArrayList<>();
        list.add("A");
        list.add("B");
        list.add("A");
        System.out.println("List: " + list); // [A, B, A]

        // Set: 순서 X, 중복 X
        Set<String> set = new HashSet<>();
        set.add("A");
        set.add("B");
        set.add("A"); // 중복 무시
        System.out.println("Set: " + set); // [A, B]

        // Map: Key-Value
        Map<String, Integer> map = new HashMap<>();
        map.put("apple", 3);
```

```
        map.put("banana", 5);
        map.put("apple", 10); // 같은 키면 덮어씀
        System.out.println("Map: " + map); // {banana=5, apple=10}

        // 조회
        System.out.println("List 첫번째: " + list.get(0));
        System.out.println("Set contains B: " + set.contains("B"));
        System.out.println("Map apple: " + map.get("apple"));
    }
}
```

✅ 정리

- 박싱/언박싱: 기본형 ↔ 객체형 변환
- Iterator/Enumeration: 컬렉션 순회 인터페이스(Iterator 권장)
- 모든 클래스는 Object 상속
- 제네릭: 타입 안전성 + 캐스팅 제거
- Map = 파이썬 dict 유사 (HashMap, TreeMap 등 존재)
- Collection = List, Set, Queue (Map도 프레임워크 일부)
- NPE: null 참조 시 발생
- Set/List/Map에 데이터 추가·조회 기본은 꼭 익혀야 함