타입 안전 이종 컨테이너를 고려하라

최 혁

Intro

Set<E>, Map<K,V> 같은 컬렉션이나 ThreadLocal<T>, AtomicReference<T> 등의 단일원소 컨테이너는 매개변수화할 수 있는 타입의 수가 제한된다. (보통 한 두개)

데이터베이스의 행과 열 중 모든 열을 타입 안전하게 이용하는 것처럼 좀 더 유연하게 타입 검증할 방법은 없을까?

이종 컨테이너(heterogeneous container)

이종 컨테이너: 각 요소의 타입이 다를 수 있는 컨테이너
(하나의 컨테이너에서 int, double, string 등 다양한 타입의 요소들을 함께 관리할 수 있는 컨테이너)

```
public static void main(String[] args) {

  Map<Class<?>,Object> favorites = new HashMap<>();

  favorites.put(String.class,"안녕");
  favorites.put(Integer.class,"숫자인 척 하는 문자");

  Integer o = (Integer) favorites.get(Integer.class); // ClassCastException
}
```

맵의 값 타입은 단순히 Object이기에 키와 값 사이의 타입 관계를 보증하지 않는다.

이종 컨테이너는 다양한 타입의 데이터를 한 곳에 모으고 관리하는 것이 편리하지만, 타입 안전성이 보장되지 않을 수 있어서 **타입 안전성이 보장된 이종 컨테이너를 사용하는 것이 좋다.**

```
//타입 안전 이종 컨테이너
public class Favorites {
    public <T> void putFavorite(Class<T> type, T instance);
    public <T> T getFavorite(Class<T> type);
}
```

```
class Favorites {
    private Map<Class<?>, Object> favorites = new HashMap<>();

    public <T> void put(Class<T> type, T instance) {
        favorites.put(Objects.requireNonNull(type), instance);
    }

    public <T> T get(Class<T> type) {
        return type.cast(favorites.get(type));
    }
}
```

- Favorites 인스턴스는 타입 안전하다.(String을 요청하면 Integer를 반환하지 않는다)
- 모든 키의 타입이 제각각이라, 일반적인 맵과 달리 여러 타입의 원소를 담을 수 있다.

favorite 클래스가 알아두어야 할 제약 1

악의적인 클라이언트가 Class 객체를 제네릭이 아닌 `로 타입`으로 넘기면 Favories 인스턴스의 타입 안전성이 깨진다.

```
favorites.put((Class) Integer.class, "Invalid Type");
```

Favorites가 타입 불변식을 어기는 일이 없도록 보장하기 위해서는 putFavorite() 메서드를 다음과 같이 수정해야 한다.

```
    public <T> void putFavorite(Class<T> type, T instance) {
        favorites.put(Objects.requireNonNull(type), type.cast(instance));
    }
```

Collections.checkedSet, .checkedList, .checkedMap같은 메서드가 이 방식을 적용한 컬렉션 래퍼들이다.

1. 이 정적 펙터리는 Class 객체를 받는다.
2. 제네릭으로 Class 객체와 컬렉션의 컴파일타임 타입이 같음을 보장한다.
3. 내부 컬렉션들을 실체화한다.

```
public static <E> List<E> checkedList(List<E> list, Class<E> type) {
    return (list instanceof RandomAccess ?
            new CheckedRandomAccessList<>(list, type) :
            new CheckedList<>(list, type));
}
```

위 컬렉션을 사용하면 로 타입으로부터 타입 안정성을 지킬 수 있다..

favorite 클래스가 알아두어야 할 제약 2

실체화 불가 타입에는 사용할 수 없다
(String이나 String[]는 저장할 수 있어도 List<String>은 저장할 수 없다)

List<String>용 Class 객체를 얻을 수 없기 때문이다.
(List<String>.class는 문법 오류)

만약 Favorites이 허용하는 타입을 제한하고 싶다면?

=> 한정적 타입 토큰을 활용하면 가능하다!
(한정적 타입 토큰은 단순히 한정적 타입 매개변수나 한정적 와일드카드를 사용해 표현 가능한 타입을 제한하는 타입 토큰이다.)

Annotation 은 한정적 타입 토큰으로 적극적으로 활용된다.

```
public <T extends Annotation> T getAnnotation(Class<T> annotationType);
```