Collections with generics

2018년 6월 5일 화요일 오후 3:12

• ArrayList
- add(), remove() ···

• Sorting
- Collections.sort()
- sort() method declaration :

• Generics
- 클래스 내부에서 사용할 data type을 외부에서 지정하는 기법. (컴파일 안에 타입쓰는 거) 타입 체크를 엄격하게 해줌.
- ArrayList ← Object는 모두 가능
- ArrayList<Animal> ← Animal object만 가능 (Animal이 generic type)
- public <T> void do(ArrayList<T> list) { ··· }
  만약 <String>이라면 body에서 <T>가 String으로 처리하게 됨. (치환됨)

• Comparable
- 비교 가능한 object.
- 실제 비교 CompareTo() (equals()는 그 자체가 같은지 비교.)
- Partial order 개념과 비슷함. : 숫자나 숫자가 아닌 것을 어떻게 정렬/비교할 것인가?
- CompareTo()를 어떻게 정의하느냐에 따라 비교의 기준이 달라짐.

• Comparator
- 좀더 OOP스러운 방법.
- Comparable 과의 차이?

• Duplicate
- LIST
- SET: 중복되는 elements를 제거하고 unique한 element만 담는 것.
  └ HashSet
```
HashSet<Song> songSet = new HashSet<Song>();
public boolean equals(Object o) {
    Song s = (Song) o;
    return this.getTitle().equals(o.getTitle());
}
```
  └ TreeSet
- Map : key-value pair로 데이터를 담는 것.
  └ HashMap
```
HashMap<String, Integer> score = new HashMap<String, Integer>();
score.put("a", 0);
score.put("b", 50);
```

• 대체 Generics
```
public void do(ArrayList<Animal> anim) { ... }
do(new ArrayList<Animal>());
do(new ArrayList<Dog>()); ← 완전!
```
그동 array라면 가능.
```
public void do(Animal[] anim) { ··· }
do(Dog dogs[]);
```

• WildCard
```
public void do(ArrayList<? extends Animal> anim) { ··· }
```
이렇게하면 Animal의 subclass를 모두 받을 수 있음.
그런데 anim의 type이 지정되지 않았으므로 anim에 object를 추가할 수 없음. (read만 가능)