1. 제네릭

1. 제네릭이란

1. 자바 제네릭은 타입의 안정성과 재사용성을 높여주는 기능.
2. 제네릭을 사용하면 클래스나 메소드를 정의할 때 타입을 지정하지 않고 인스턴스 생성할 때나 메소드를 호출할 때 타입을 지정하여 타입만 변경하여 계속 사용 가능.
3. 제네릭을 사용하면 중복된 코드를 제거할 수 있고 유지보수성과 가독성이 좋아진다.
4. 의미상의 제네릭

- T : Type
- K : Key
- V : Value
- E : Element
- N : Number

2. 제네릭 클래스

1. 제네릭 클래스는 클래스명 다음에 (타입매개변수) 을 붙여서 정의하는 클래스.
2. 에는 어떤 클래스도 사용할 수 있다.
3. 정의

```
public class Clss<T> {
   List<T> listT;
   T t;
   Map<String, T> tMap;
}
```

4. 사용

```
Clss<타입으로 사용할 클래스> cl = new Clss<타입으로 사용할 클
```

3. 제네릭 메소드

1. 메소드의 반환타입 앞에 <T>를 붙여서 정의한다. T는 임의의 이름으
2. public <T> T test(T[] tArr or T t or List<T> tList .....)

4. 제한된 제네릭

1. <T>에 제한을 줄 수 있다. 어떤 클래스를 상속받은 클래스만 허용하
2. <T extends 부모클래스> : 부모클래스를 상속받은 클래스만 T로 지
3. <T super 자식클래스> : 자식클래스에 상속을 해준 부모클래스와 자

5. 와일드카드(?)

1. 와일드 카드 <?>는 메소드의 매개변수에 List, Map, Set...에 어떤 타입의 클래스든 다 지정할 수 있게 하고 싶을 때 사용한다.

```
public int add(List<Integer> intList) {

}
public int add(List<Long> longList) {

}
```

- 위 두 개의 메소드는 타입이 다르지만 매개변수의 형태가 리스트로 동일하다. 그래서 메소드 오버로딩이 성립되지 않는다. 위 문제를 방지하고자 할 때 와일드카드를 사용한다.

```
public int add(List<?> intList) {

}
```

2. 와일드카드도 extends와 super 키워드를 사용해서 제한을 걸 수 있다.