Generics <> Ⅲ

제네릭을 구성하는 다양한 문법적 요소 Ⅱ

와일드카드 <?>

- **FruitBox<?> box**
  - **어떤 자료형을 기반 한 제네릭 클래스의 인스턴스이건, 다 참조하겠다.**
  - **제네릭에 올 수 있는 데이터 타입은 모든 클래스가 올 수 있다.**
  - FruitBox<? extends Object> box와 동일

Apple extends Fruit의 경우

- **FruitBox<? extends Fruit> box1 = new FruitBox<Fruit>();**
- **FruitBox<? extends Fruit> box2 = new FruitBox<Apple>();**
  - **어떤 자료형을 기반으로 한 제네릭 클래스의 인스턴스이건**
    다 참조하는데, 제네릭의 자료형은 Fruit 자료형을 상속받은 자료형으로 제한한다.

상위, 하위 클래스를 제한하는 용도의 와일드카드 <?>

- **하위 클래스 : FruitBox<? extends Apple> box1**
  - **~를 상속하는 클래스면 무엇이든지**
  - **Apple을 상속하는 클래스의 인스턴스라면 무엇이던지 참조가 가능한 참조변수**

- **상위 클래스 : FruitBox<? super Apple> box2**
  - **~이 상속하는 클래스면 무엇이든지**
  - **Apple이 상속하는 클래스의 인스턴스라면 무엇이던지 참조가 가능한 참조변수**
  - **직, 간접 상위 클래스 모두**

제네릭 클래스의 상속

i. **상위 클래스의 T를 결정하지 않고 상속하는 경우**

```
class AAA<T> {
     T itemAAA;
}

class BBB<T> extends AAA<T> {
     T itemBBB;
}
```

  - 하위 클래스도 제네릭 클래스가 된다!
  - BBB<Integer> bbb = new BBB<Integer>();

ii. **상위 클래스의 T를 결정헤서 상속하는 경우**

```
class AAA<T> {
     T itemAAA;
}

class BBB extends AAA<String> {
     T itemBBB;
```

```
}
```

- 하위 클래스는 제네릭 클래스가 아니다! (제네릭이 결정 되었으므로)
- `BBB bbb = new BBB();`

제네릭 인터페이스의 구현

```
interface MyInterface<T> {
    public T myfunc(T item);
}
```

i. 상위 인터페이스의 T를 결정하지 않고 구현하는 경우

```
class MyClass<T> implements MyInterface<T> {
    @Override
    public T myFunc(T item){
        return item;
    }
}
```

ii. 상위 인터페이스의 T를 결정헤서 구현하는 경우

```
class Myclass implements MyInterface<String> {
    @Override
    public String myFunc(String item){
        return item;
    }
}
```

기본자료형의 이름은 제네릭에 사용 불가!

- **`FruitBox<int> box1 = new FruitBox<int>(); X`**
- **`FruitBox<double> box1 = new FruitBox<double>(); X`**
- **제네릭을 클래스와 인스턴스로 제한했다.**

- **기본자료형을 기반으로 제네릭을 사용할 때에는 Wrapper 클래스를 사용한다!**
- **`FruitBox<Integer> box1 = new FruitBox<Integer>();`**
- **`FruitBox<Double> box1 = new FruitBox<Double>();`**
- **Wraper Class를 사용하더라도, Auto Boxing, Auto UnBoxing에 의해 기본자료형 사용에 구애를 받지 않는다.**
