

JAVA Generic Method

프로그램은 사람이 이해하는 코드를 작성.
느려도 꾸준하면 경기에서 이긴다.

Content

6. Generic Method

1. Generic 이란
2. Generic Class
3. Generic Method

“ Data type을 특정한 type하나로 정하지 않고 사용할 때마다 바뀔 수 있게 범용적이고 포괄적으로 지정 ”

Generic

- 포괄적인, 총칭의, 회사 이름이 붙지 않은, 일반 명칭으로 판매되는 사전적 의미로, 딱 하나를 정하지 않고 범용적이고 포괄적이라는 의미
- Data type을 특정한 type하나로 정하지 않고 사용할 때마다 바뀔 수 있게 범용적이고 포괄적으로 지정한다 라는 의미
- Object Class의 한계를 극복
- 호출되는 시점에 실제 Generic 타입을 지정

01. 왜 Generic가 만들어 졌을까?

```
public class GenericMethodMain {  
    public static void main(String... args) {  
  
        List list = new ArrayList<>();  
        list.add("홍길동");  
        list.add("홍당무");  
        list.add("홍사과");  
  
        for (int i = 0; i < list.size(); i++) {  
            String str = list.get(i);  
            System.out.println(str);  
        }  
    }  
}
```

java: incompatible types: java.lang.Object
cannot be converted to java.lang.String

```
public class GenericMethodMain {  
    public static void main(String... args) {  
  
        List list = new ArrayList<>();  
        list.add("홍길동");  
        list.add("홍당무");  
        list.add("홍사과");  
  
        for (int i = 0; i < list.size(); i++) {  
            String str = (String) list.get(i);  
            System.out.println(str);  
        }  
    }  
}
```

Type Casting 이 빈번하게 일어나서 성능 저하 발생
- 수만개의 Data가 있다면

```
public class GenericMethodMain {  
    public static void main(String... args) {  
  
        List<String> list = new ArrayList<>();  
        list.add("홍길동");  
        list.add("홍당무");  
        list.add("홍사과");  
  
        for (int i = 0; i < list.size(); i++) {  
            String str = list.get(i);  
            System.out.println(str);  
        }  
    }  
}
```

Best Code

```
for (String str: list) {  
    System.out.println(str);  
}
```

Generic Class

- Class or Interface로 정의 할 수 있음
- Class 를 정의 하는 시점에 타입을 지정 하는 것이 아니라 객체 생성 시점에 실제 타입을 지정

➤ 문법 구조

선언 : 접근지정자 class 클래스명 <Generic Type [, Generic Type ..] >

class or Interface에서 사용 되는
Generic Type 정의

호출 : 제너릭클래스명 <실제 Type> 변수명 = new 제너릭클래스명<실제 Type>()

입력 매개변수의 실제 타입을 의미함

➤ 관례적 표기와 의미

Generic Type	의미
T	타입 (Type)
K	키 (Key)
V	값 (Value)
N	숫자 (Number)
E	원소 (Element)

1. Generic Class

6. Generic 6-2. Generic Class

Class 내부에 사용 하는 Generic Type

```
public class GenericClass<T, V> {  
    private T name;  
    private V price;  
  
    GenericClass(T name, V price) {  
        this.name = name;  
        this.price = price;  
    }  
  
    public T getName() {  
        return name;  
    }  
  
    public void setName(T name) {  
        this.name = name;  
    }  
  
    public V getPrice() {  
        return price;  
    }  
  
    public void setPrice(V price) {  
        this.price = price;  
    }  
}
```

```
public static void main(String... args) {  
  
    List<GenericClass<String, Integer>> cars = new ArrayList<>();  
    cars.add((new GenericClass<>("쏘나타", 10000)));  
    cars.add((new GenericClass<>("그랜저", 20000)));  
  
    for(GenericClass<String, Integer> car: cars) {  
        System.out.println(car.getName() + "\t가격 : " + String.format("%,d", car.getPrice() )  
            + " (부가세 : " + String.format("%,.1f", car.getPrice().floatValue() * 0.1) + ")");  
    }  
  
    List<GenericClass<String, Float>> fruits = new ArrayList<>();  
    fruits.add((new GenericClass<>("사과", 256.5f)));  
    fruits.add((new GenericClass<>("복숭아", 345.6f)));  
  
    for(GenericClass<String, Float> fruit: fruits) {  
        System.out.println(String.format("%s \t가격 : %,1f ( 부가세 : %,1f )" ,  
            fruit.getName()  
            , fruit.getPrice()  
            , fruit.getPrice() * 0.1 ));  
    }  
}
```

Generic Method

- Method 만들 때 파라미터와 반환 값의 자료형을 Generic로 선언 한 Method
- 입력 매개 변수 값으로 Generic Type을 유추 할 수 있을 경우는 생략 가능
- Generic Method 내부에서는 매개변수로 한 참조 변수의 메서드로 Object Class의 Method만 가능

➤ 관련적 표기와 의미

Generic Type	의미
T	타입 (Type)
K	키 (Key)
V	값 (Value)
N	숫자 (Number)
E	원소 (Element)

```
GenericMethod genericMethod = new GenericMethod();
int num01 = genericMethod.<Integer> method(10);
System.out.println(num01);

// 입력 매개 변수가 Genetic Type가 유추 할 수 있는 경우 생략 가능
int num02 = genericMethod.method(10);
System.out.println(num02);

String str02 = genericMethod.methodRnString("안녕" , 10);
System.out.println(str02);

genericMethod.methodPrint("name" , "홍길동");
```

```
10
10
안녕10
name : 홍길동
```

➤ 문법 구조

선언 : 접근지정자 < T [, ..] > T 메서드명(T t [, V v, ...]);

호출 : 참조객체.<실제 제네릭 타입>메서드명(입력매개변수)

입력 매개변수의 타입의 개수 만큼

입력 매개변수의 실제 타입을 의미함

```
<T> T method(T t) {
    return t;
}

// Generic Method 내부에서 매개변수의 산술 연산 되지 않음 -> Object
<T> String methodRnString(T t1, T t2){
    // t1 + t2 : Operator '+' cannot be applied to 'T', 'T'
    return t1 + "" + t2;
}

<K,V> void methodPrint(K t1, V v1){
    System.out.println(t1 + " : " + v1 );
}
```

Generic Method -

- extends : 매개 변수에 특정 타입만 받게 제한 할 때 사용
- 여러 개 사용시 임의로 타입 설정

```
String str03 = genericMethod.methodPrintRnString("name" , "홍길동");  
System.out.println(str03);  
  
System.out.println(genericMethod.checkEquals(10 , 2.1));  
System.out.println(genericMethod.checkEquals("사과" , "배"));  
System.out.println(genericMethod.checkEquals("사과" , "사과"));  
  
System.out.println(genericMethod.<String, Integer, Object>extednsMethod("사과" , 0, "첫문자 : "));  
System.out.println(genericMethod.extednsMethod("사과" , 0, "첫문자 : "));
```

첫번째 파라미터 : 문자
두번째 매개변수 : 숫자
세번째 파라미터 : Object

```
<K,V> void methodPrint(K t1, V v1){  
    System.out.println(t1 + " : " + v1 );  
}  
  
<K,V> V methodPrintRnString(K t1, V v1){  
    // t1 + t2 : Operator '+' cannot be applied to 'T', 'T'  
    System.out.println(t1 + " : " + v1 );  
    return (V) (t1 + " : " + v1);  
}  
  
<T, V> boolean checkEquals(V v1, V v2){  
    return v1.equals(v2);  
}  
  
// Generic Method 내부에서는 매개변수로 한 참조 변수의 메서드로  
// Object Class의 Method만 가능  
<T extends String, I extends Integer, V> String extednsMethod(T t, I i, V v){  
    char ch = t.charAt(i);  
    return v.toString() + ch ;  
}
```