제너릭 메소드

제너릭 메소드

❖ 제네릭 메소드

- 매개변수 타입과 리턴 타입으로 타입 파라미터를 갖는 메소드
- 제네릭 메소드 선언 방법
  - 리턴 타입 앞에 "<>" 기호를 추가하고 타입 파라미터 기술
  - 타입 파라미터를 리턴 타입과 매개변수에 사용

```
public <타입파라미터,...> 리턴타입 메소드명(매개변수,...) { ... }
```

```
public <T> Box<T> boxing(T t) { ... }
```

❖ 제네릭 메소드

- 제네릭 메소드 호출하는 두 가지 방법

```
리턴타입 변수 = <구체적타입> 메소드명(매개값);      //명시적으로 구체적 타입 지정
리턴타입 변수 = 메소드명(매개값);                  //매개값을 보고 구체적 타입을 추정
```

```
Box<Integer> box = <Integer>boxing(100);      //타입 파라미터를 명시적으로 Integer 로 지정
Box<Integer> box = boxing(100);               //타입 파라미터를 Integer 으로 추정
```

제너릭 메소드

❖ Box.java

```
public class Box<T> {
    private T t;

    public T get() {
        return t;
    }

    public void set(T t) {
        this.t = t;
    }
}
```

❖ Util.java

```
public class Util {
    public static <T> Box<T> boxing(T t) {
        Box<T> box = new Box<T>();
        box.set(t);
        return box;
    }
}
```

제너릭 메소드

❖ BoxingMethodExample.java

```
public class BoxingMethodExample {
    public static void main(String[] args) {
        Box<Integer> box1 = Util.<Integer> boxing(100);
        int intValue = box1.get();

        Box<String> box2 = Util.boxing("홍길동");
        String strValue = box2.get();
    }
}
```

제너릭 메소드

❖ Util.java

```
public class Util {
    public static <K, V> boolean compare(Pair<K, V> p1, Pair<K, V> p2) {

        boolean keyCompare = p1.getKey().equals(p2.getKey());
        boolean valueCompare = p1.getValue().equals(p2.getValue());

        return keyCompare && valueCompare;
    }
}
```

제너릭 메소드

❖ Pair.java

```
public class Pair<K, V> {
    private K key;
    private V value;

    public Pair(K key, V value) {
        this.key = key;
        this.value = value;
    }

    public void setKey(K key) {
        this.key = key;
    }

    public void setValue(V value) {
        this.value = value;
    }

    public K getKey() {
        return key;
    }

    public V getValue() {
        return value;
    }
}
```

제너릭 메소드

❖ CompareMethodExample.java

```
public class CompareMethodExample {
    public static void main(String[] args) {
        Pair<Integer, String> p1 = new Pair<Integer, String>(1, "사과");
        Pair<Integer, String> p2 = new Pair<Integer, String>(1, "사과");
        boolean result1 = Util.<Integer, String> compare(p1, p2);
        if (result1) {
            System.out.println("논리적으로 동등한 객체입니다.");
        } else {
            System.out.println("논리적으로 동등하지 않는 객체입니다.");
        }

       Pair<String, String> p3 = new Pair<String, String>("user1", "홍길동");
       Pair<String, String> p4 = new Pair<String, String>("user2", "홍길동");
        boolean result2 = Util.compare(p3, p4);
        if (result2) {
            System.out.println("논리적으로 동등한 객체입니다.");
        } else {
            System.out.println("논리적으로 동등하지 않는 객체입니다.");
        }
    }
}
```