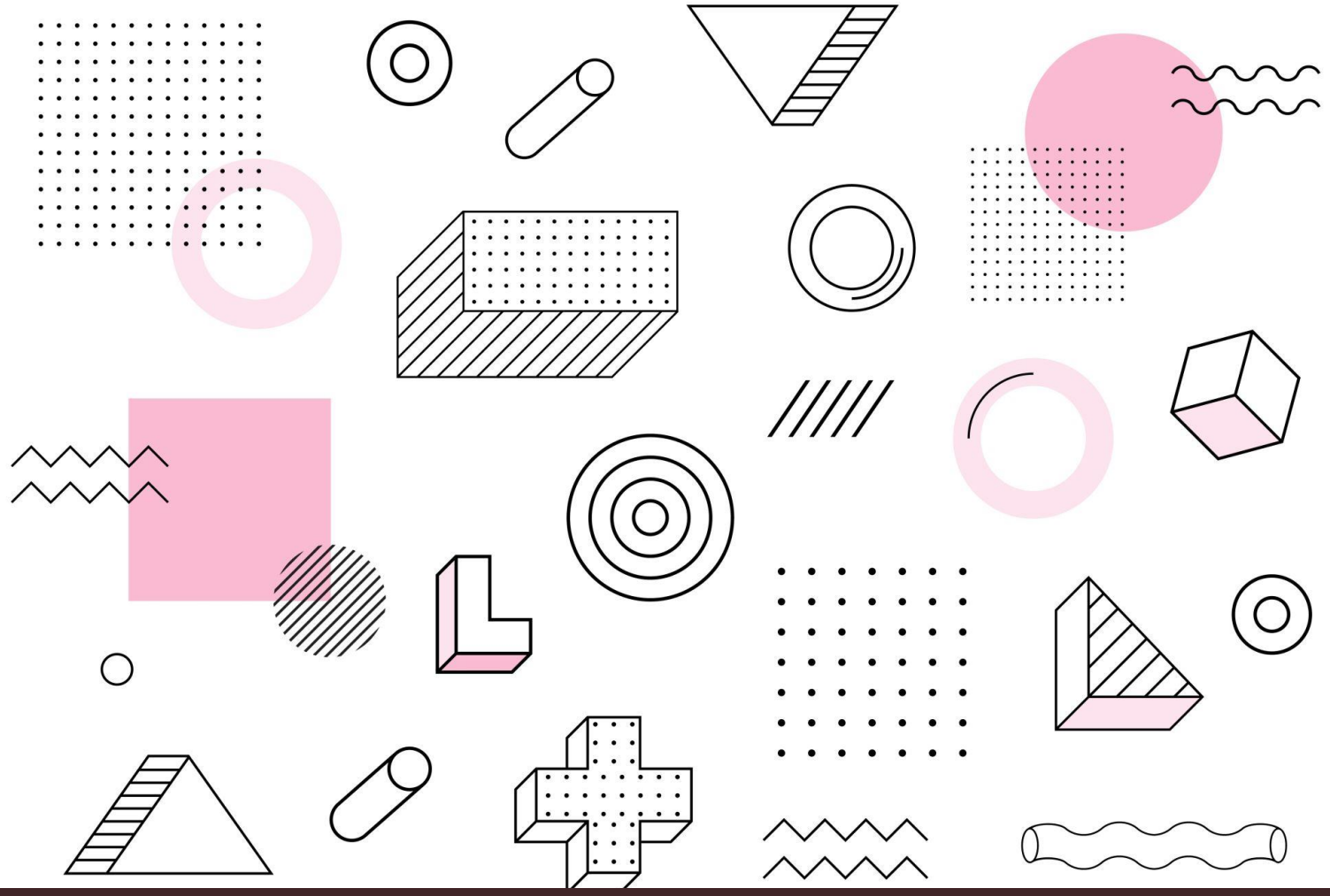

1학기 고급자바 실습

week 3-1

김민진(18)

김지희(18)

문의 메일 : genie02166@duksung.ac.kr

Part1. 제네릭

1) 제네릭 타입

2) 멀티 타입 파라미터

3) 제네릭 메소드

- 예제 1
- 예제 2

4) 제한된 타입 파라미터

5) 와일드카드 타입

6) 제네릭 타입의 상속과 구현

1) 제네릭 타입

Box.java

```java
package week3_1;

public class Box<T> {
    private T t;
    public T get() {return t;}
    public void set(T t) {this.t = t;}
}

```

BoxExample.java

```java
package week3_1;

public class BoxExample {

    public static void main(String[] args) {
        Box<String> box1 = new Box<String>();
        box1.set("hello");
        String str = box1.get();

        Box<Integer> box2 = new Box<Integer>();
        box2.set(6);
        int value = box2.get();
    }

}
```

2) 멀티 타입 파라미터

Product.java | ProductExample.java

```java
package week3_1;

public class Product<T, M> {

    private T kind;
    private M model;

    public T getKind() {return this.kind;}
    public M getModel() {return this.model;}

    public void setKind(T kind) {this.kind = kind;}
    public void setModel(M model) {this.model = model;}
}

class Tv{
    private String name;
    public Tv(String name) {
        this.name=name;
    }
    public String getName() {return this.name;}
}
class Car{
    private String name;
    public Car(String name) {
        this.name=name;
    }
    public String getName() {return this.name;}
}
```

2) 멀티 타입 파라미터

Product.java | ProductExample.java

```java
package week3_1;

public class ProductExample {
    public static void main(String[] args) {
        Product<Tv, String> product1 = new Product<>();
        product1.setKind(new Tv("Tv"));
        product1.setModel("SmartTv");
        Tv tv = product1.getKind();
        String tvModel = product1.getModel();

        System.out.println(tv.getName());
        System.out.println(tvModel);

        Product<Car, String> product2 = new Product<Car, String>();
        product2.setKind(new Car("Car"));
        product2.setModel("디젤");
        Car car = product2.getKind();
        String carModel = product2.getModel();

        System.out.println(car);
        System.out.println(carModel);
    }
}

```

실행 결과

Problems | Javadoc | Declar

<terminated> ProductExample [Jav

```
Tv
SmartTv
Car
디젤
```

3) 제네릭 메소드

- 예제 1

Util.java

```
1  package week3_1;
2
3  public class Util {
4      public static <T> Box<T> boxing(T t) {
5          Box<T> box = new Box<T>();
6          box.set(t);
7          return box;
8      }
9  }
10
```

BoxingMethodExample.java

```
1  package week3_1;
2
3  public class BoxingMethodExample {
4      public static void main(String[] args) {
5          Box<Integer> box1 = Util.<Integer>boxing(100);
6          int intValue = box1.get();
7
8          Box<String> box2 = Util.boxing("홍길동");
9          String strValue = box2.get();
10     }
11 }
12
```

3) 제네릭 메소드

- 예제 2

Util2.java | Pair.java | CompareMethodExample.java

```java
package week3_1;

public class Util2 {
    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;
    }
}

```

Util2.java | Pair.java | CompareMethodExample.java

```java
package week3_1;

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; }
}

```

3) 제네릭 메소드

- 예제 2

Util2.java | Pair.java | CompareMethodExample.java

```
1  package week3_1;
2
3  public class CompareMethodExample {
4      public static void main(String[] args) {
5          Pair<Integer, String> p1 = new Pair<Integer, String>(1, "사과");
6          Pair<Integer, String> p2 = new Pair<Integer, String>(1, "사과");
7          boolean result1 = Util2.<Integer, String>compare(p1, p2);
8          if(result1) {
9              System.out.println("논리적으로 동등한 객체입니다.");
10         } else {
11             System.out.println("논리적으로 동등하지 않는 객체입니다.");
12         }
13
14         Pair<String, String> p3 = new Pair<String, String>("user1", "홍길동");
15         Pair<String, String> p4 = new Pair<String, String>("user2", "홍길동");
16         boolean result2 = Util2.compare(p3, p4);
17         if(result2) {
18             System.out.println("논리적으로 동등한 객체입니다.");
19         } else {
20             System.out.println("논리적으로 동등하지 않는 객체입니다.");
21         }
22     }
23 }
```

실행 결과

Console | Problems | Javadoc

```
<terminated> CompareMethodExample
논리적으로 동등한 객체입니다.
논리적으로 동등하지 않는 객체입니다.
```

4) 제한된 타입 파라미터

Util.java

```java
package week3_1;

public class Util {

    public static <T extends Number> int compare(T t1, T t2) {
        double v1 = t1.doubleValue();
        double v2 = t2.doubleValue();
        return Double.compare(v1,  v2);
    }
}

```

BoundedTypeParameterEx.java

```java
package week3_1;

public class BoundedTypeParameterEx {

    public static void main(String[] args) {
        //String str = Util.compare("a", "b");  (x)

        int result1 = Util.compare(10, 20);
        System.out.println(result1);

        int result2 = Util.compare(4.5, 3);
        System.out.println(result2);
    }
}
```

실행 결과

Problems @ Javadoc Declaration Co

<terminated> BoundedTypeParameterEx [Java

```
-1
1
```

출석 과제 (3/29 월 오후 11:55 마감)

Q. 제네릭에 대한 설명으로 틀린 것은 무엇입니까?

1) 컴파일 시 강한 타입 체크를 할 수 있다.
2) 타입 변환(casting)을 제거한다.
3) 제네릭 타입은 타입 파라미터를 가지는 제네릭 클래스와 인터페이스를 말한다.
4) 제네릭 메소드는 리턴 타입으로 타입 파라미터를 가질 수 없다.