제네릭(Generic) 타입

제네릭(Generic) 타입

❖ 제네릭(Generic) 타입이란?

- ‘컴파일 단계’에서 ‘잘못된 타입 사용될 수 있는 문제’제거 가능
- 자바5부터 새로 추가 !
- 컬렉션, 람다식(함수적 인터페이스), 스트림, NIO에서 널리 사용
- 제네릭을 모르면 API 도큐먼트 해석 어려우므로 학습 필요

Class ArrayList<E>

```
default BiConsumer<T,U> andThen(BiConsumer<? super T,? super U> after)
```

제네릭(Generic) 타입

❖ 제네릭을 사용하는 코드의 이점

- ㅇ 컴파일 시 강한 타입 체크 가능
  - ▪ 실행 시 타입 에러가 나는 것 방지
  - ▪ 컴파일 시에 미리 타입을 강하게 체크해서 에러 사전 방지

- ㅇ 타입변환 제거 가능

```
List list = new ArrayList();
list.add("hello");
String str = (String) list.get(0);
```

```
List<String> list = new ArrayList<String>();
list.add("hello");
String str = list.get(0);
```

제네릭(Generic) 타입

❖ 제네릭 타입이란?

- 타입을 파라미터로 가지는 클래스와 인터페이스
- 선언 시 클래스 또는 인터페이스 이름 뒤에 "<>" 부호 붙임
- "<>" 사이에는 타입 파라미터 위치

- 타입 파라미터
  - 일반적으로 대문자 알파벳 한 문자로 표현
  - 개발 코드에서는 타입 파라미터 자리에 구체적인 타입을 지정해야

```
public class 클래스명<T> { ... }
public interface 인터페이스명<T> { ... }
```

제네릭(Generic) 타입

❖ 제네릭 타입 사용 여부에 따른 비교

- 제네릭 타입을 사용하지 않은 경우
- Object 타입 사용 → 빈번한 타입 변환 발생 → 프로그램 성능 저하

```
public class Box {
   private Object object;
   public void set(Object object) { this.object = object; }
   public Object get() { return object; }
}
```

```
Box box = new Box();
box.set("hello");                  //String 타입을 Object 타입으로 자동 타입 변환해서 저장
String str = (String) box.get();   //Object 타입을 String 타입으로 강제 타입 변환해서 얻음
```

제네릭(Generic) 타입

❖ Apple.java

```
public class Apple {

}
```

❖ Box.java

```
public class Box {
    private Object object;

    public void set(Object object) {
        this.object = object;
    }

    public Object get() {
        return object;
    }
}
```

제네릭(Generic) 타입

❖ BoxExample.java

```
public class BoxExample {

    public static void main(String[] args) {
        Box box = new Box();
        box.set("홍길동");
        String name = (String) box.get();

        box.set(new Apple());
        Apple apple = (Apple) box.get();
    }

}
```

제네릭(Generic) 타입

❖ 제네릭 타입 사용 여부에 따른 비교

- ㅇ 제네릭 타입 사용한 경우
  - 클래스 선언할 때 타입 파라미터 사용
  - 컴파일 시 타입 파라미터가 구체적인 클래스로 변경

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

Box<String> box = new Box<String>();

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

```
Box<String> box = new Box<String>();
box.set("hello");
String str = box.get();
```

Box<Integer> box = new Box<Integer>();

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

```
Box<Integer> box = new Box<Integer>();
box.set(6);
int value = box.get();
```

제네릭(Generic) 타입

❖ Box.java

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

    public T get() {
        return t;
    }

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

제네릭(Generic) 타입

❖ BoxExample.java

```
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();
    }
}
```

제네릭(Generic) 타입

❖ 제네릭 타입은 두 개 이상의 타입 파라미터 사용 가능

- 각 타입 파라미터는 콤마로 구분
  - Ex) class<K, V, …> { … }
  - interface<K, V, …> { … }

```
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; }
}
```

```
Product<Tv, String> product = new Product<Tv, String>();
```

- 자바 7부터는 다이아몬드 연산자 사용해 간단히 작성과 사용 가능

```
Product<Tv, String> product = new Product<>();
```

제네릭(Generic) 타입

❖ Car.java

```
public class Car {

}
```

❖ Tv.java

```
public class Tv {

}
```

제네릭(Generic) 타입

❖ Product.java

```
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;
    }
}
```

제네릭(Generic) 타입

❖ ProductExample.java

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

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