**객체지향 프로그래밍** | 최윤정 교수님 | 실습#9 Summary 이서현 2276219

Lab-Box, Main\_Test

텍스트, 번호, 라인, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Lab# : 와일드카드 타입 사용 예 미리보기 (Box에 담을 수 있는 것을 정의한 후, put메소드에서 형태를 제한.)

와일드카드 관련은 하단부에 정리

**List컬렉션: ArrayList**

가변 크기의 배열. 초기용량을 지정한 후에도 저장용량을 초과하면 자동적으로 늘어남(고정가능)

* ArrayList의 생성: List<E> list = new ArrayList<E>();

List<String> list = Arrays.asList(new String[size]); //일반 배열을 리스트로 변환

* 원소 추가: list.add( "aaa" );

**제네릭(Generic)** **프로그래밍**

 일반적인 코드를 작성하고, 이 코드를 다양한 타입의 객체에 대하여 재사용하는 프로그래밍 기법이다.

 타입을 파라미터로 가지는 클래스와 인터페이스를 정의한다.

- 선언 시 클래스 또는 인터페이스 이름 뒤에 “<T>". <> 사이에는 타입 파라미터\*를 명시한다.

\*타입 파라미터: 일반적으로 대문자 알파벳 한 문자로 표현. 두 개 이상의 타입 파라미터도 사용 가능.

(ex- <E, V> : element, value; class<K, V, …> { … }, interface<K, V, …>{…})

**제네릭을 사용하는 이유**(1)컴파일 시 강한 '타입 체크' 가능

클래스들의 구조와 연결관계가 복잡해지면서 TypeMiscasting 등의 타입오류들이 발생할 수 있음

컴파일 시에 미리 타입을 강하게 체크하여 에러를 발생시킨다.

(실행 시 런타임-타입 에러가 나는 것 방지.)

(2) 타입명시 / 타입변환을 제거 가능

list에 일관된 타입의 정보를 저장하도록 명시한다.

**제네릭 메소드** 매개변수 타입과 리턴 타입으로 타입 파라미터를 갖는 메소드.

제네릭 메소드 선언 방법: 리턴 타입 앞에 “<\_>” //\_는 타입 파라미터

**public** **class** Array {

**public** **static** <T> T getLast(T[] a){

**return** a[a.length-1];

}

}

제네릭 메소드 호출 - 두 가지 방법

1. 명시적으로 구체적 타입 지정 Box<Integer> box = <Integer>boxing(100);
2. 매개값을 보고 구체적 타입을 추정 Box<Integer> box = boxing(100);

Lab#: 제네릭 클래스/메소드의 예 : ArrayList 미리보기 (Box.java, Main\_Test.java 파일 참고)

**제한된 타입 파라미터** 상속 및 구현 관계 이용해 타입 제한

 메소드의 인자 또는 리턴타입에 제한을 둘 수 있다. 상위 타입은 클래스뿐만 아니라 인터페이스도 가능함.

**public** <T **extends** 동물> T getAnimal(){…} //상위타입제한

 타입 파라미터 대체할 구체적인 타입 명시. 같은 상속트리에 있거나, 인터페이스로 연결된 경우만 지정 가능.

Box<동물> box1= **new** Box();

Box<강아지> box2= **new** Box(); Box<오리> box3= **new** Box(); //강아지 클래스와 오리 클래스는 동물 클래스를 상속받음

**와일드카드 타입** 세가지 형태 – 매개변수(인자)에서 허용  
(1) 제네릭타입<?>: Unbounded Wildcards (제한없음)

구체적인 타입, 모든 클래스나 인터페이스 타입 가능

(2) 제네릭타입<? extends 상위타입>: Upper Bounded Wildcards (상위클래스 제한)  
 '상위타입 or 상위타입의 하위타입'만 올 수 있다

(3) 제네릭타입<? super 하위타입>: Lower Bounded Wildcards (하위클래스 제한)  
 '하위타입 or 하위타입의 상위타입'만 올 수 있다

**제네릭 타입의 상속과 구현**

제네릭 타입을 부모 클래스로 사용할 경우 타입 파라미터는 자식 클래스에도 기술해야.

* 추가적인 타입 파라미터 가질 수 있음. public class ChildProduct<T, M, C> extends Product<T, M> {…}
* 제네릭 인터페이스를 구현할 경우, 그걸 구현한 클래스도 제네릭 타입임.

**컬렉션Collection:** 자바에서 자료구조를 구현한 클래스

객체들을 효율적으로 추가, 삭제, 검색할 수 있도록 제공되는 라이브러리

* 리스트(list), 스택(stack), 큐(queue), 집합(set), 해쉬 테이블(hash table) 등.

 java.util 패키지에 포함됨

 컬렉션 클래스와 컬렉션 인터페이스로 제공

컬렉션 인터페이스: 주요 인터페이스와 메소드들 보기

List 컬렉션과 주요메소드

\*구현클래스: ArrayList, Vector, LinkedList

반복자 iterator: 컬렉션의 원소들을 하나씩 처리하는 데 사용