표준 & 메타 어노테이션

Annotation

=〉 소스코드에 메타데이터를 추가

- metadata : 데이터에 대한 정보를 나타내는 데이터
- 컴파일 또는 런타임 시에 사용될 수 있음
- 컴퓨터가 보라고 쓰는 주석

〈〈 자바의 표준 어노테이션들 예시 〉〉

어노테이션	설명
@Deprecated	더 이상 사용되지 않음
@Override	오버라이드되는 메소드(컴파일러가 검사)
@FunctionalInterface	함수형 인터페이스임
@SuppressWarnings	컴파일러의 경고 무시

☐ ex01

☕ Main.java

```
 //  ☐ Deprecated : 앞으로 사용하지 말 것
//  - 더 좋은 / 기존 문제를 해결한 새 기능으로 대체
//  - 기존의 메소드를 삭제할 수는 없으므로...새기능으로 쓰세요
@Deprecated
public static void deprecatedMethod () {
    System.out.println("옛날에 이렇게 작성했어...");
}

// ☐SuppressWarnings : 컴파일러의 경고 무시
//@SuppressWarnings("unchecked") //
public static void warnedMethod () {

    //  제네릭을 사용하지 않는 컬렉션
    //  - unchecked 경고 발생
    //  - 실행시 에러가 발생하지는 않음
    //  - 위의 어노테이션 활성화하고 다시 확인

    ArrayList list = new ArrayList();
    list.add("감자");
    list.add("고구마");
    list.add("호박");
    System.out.println(list);
}
```

사용자 정의 어노테이션 만들기
□ ex02
CustomAnnt.java

```
//  □ 어노테이션 정의 문법
//  – interface 앞에 @
public @interface CustomAnnt { }
```

☕ MyClass.java

```
@CustomAnnt
public class MyClass {

    @CustomAnnt
    int field;
    //  아래에 이후의 코드들 작성
}
```

메타 어노테이션
사용자 정의 어노테이션의 속성들을 정의하는 어노테이션
어노테이션의 어노테이션

@Retention
어노테이션이 유지되는 범위
정책 종류
SOURCE : 소스 파일에만 적용
예시: @Override
CLASS : 클래스 파일까지 사용됨 – 기본
실행시에는 사용 불가
컴파일 타임에 읽힐 목적으로 사용
외부 라이브러리인 Lombok 등에서 코드 생성 등의 목적으로 사용
RUNTIME : 실행시에 사용 가능
리플렉션에서 접근되려면 이 정책이어야 함

RetSource.java

```
@Retention(RetentionPolicy.SOURCE)
public @interface RetSource { }
```

RetClass.java

```
@Retention(RetentionPolicy.CLASS)
public @interface RetClass { }
```

RetRuntime.java

```
@Retention(RetentionPolicy.RUNTIME)
public @interface RetRuntime { }
```

MyClass.java

```
@RetSource
int retSource;

@RetClass
int retClass;

@RetRuntime
int retRuntime;

public static void main(String[] args) throws ClassNotFoundException {

    Class  myClass = Class.forName("com.acorn.anno.MyClass");
    for ( Field  f : myClass.getDeclaredFields()) {
        if (f.getAnnotations().length > 0) {
            Arrays.stream(f.getAnnotations() ).forEach( s ->System.out.println(s));
        }
    }
}
```

!! MyClass.class 파일 살펴볼 것
RetentionPolicy.SOURCE 인 어노테이션 없음 확인
실행하여 결과 살펴볼 것
RetentionPolicy.RUNTIME 인 어노테이션만 출력됨 확인
@Target
어노테이션이 적용될 수 있는 대상 지정
종류
CONSTRUCTOR : 생성자
FIELD : 필드
LOCAL_VARIABLE : 지역 변수
METHOD : 메소드
PACKAGE : 패키지
PARAMETER : 매개변수
TYPE : 클래스, 인터페이스, enum 등의 타입

● TargConstr.java

```
@Target(ElementType.CONSTRUCTOR)
public @interface TargConstr {
}
```

● TargField.java

```
@Target(ElementType.FIELD)
public @interface TargField { }
```

● TargMulti.java

```
@Target({
        ElementType.FIELD,
        ElementType.METHOD,
        ElementType.LOCAL_VARIABLE
})
public @interface TargMulti { }
```

MyClass.java

```
    @TargConstr
    public MyClass() { }

    @TargField
    @TargMulti
    int targField;

    @TargMulti
    public void targMethod () {}

    @TargMulti
    boolean targMulti = true;
```

@Inherited

클래스에 사용될 시 자식 클래스도 이를 물려받음

● InheritF.java

```
@Retention(RetentionPolicy.RUNTIME)
public @interface InheritF { }
```

● InheritT.java

```
@Inherited //    자식클래스로 상속됨
@Retention(RetentionPolicy.RUNTIME)
public @interface InheritT { }
```

● MySubclass.java

```
public class MySubclass extends MyClass { }
```

● MyClass.java

```
// ...
@InheritF
@InheritT
public class MyClass {
// ...

        System.out.println("\n- - - - -\n");
        Class<?> mySubclass = Class.forName("sec13.chap02.ex02.MySubclass");
        System.out.println(
                "MySubclass의 어노테이션 : " +
                Arrays.stream(mySubclass
                        .getAnnotations())
                        .map(Annotation::toString)
                        .collect(Collectors.joining(","))
        );
```

● 어노테이션 만들기

```
@Retention(RetentionPolicy.RUNTIME)
public @interface Count {
    //int value(); // 기본값이 없을 때
    int value() default 1; // □ 기본값 설정
}
```

Main.java

```
    //  괄호 안에 {필드명} = {값}
    @Count(value = 3)
    private int apples;

    //  default가 있을 시 생략 가능
    @Count
    private int bananas;

    //  필드가 하나고 필드명이 value일 시   값만 넣을 수 있음
    @Count(5)
    private int cacaos;
```

PersonName.java

```
@Retention(RetentionPolicy.RUNTIME)
public @interface PersonName {
    String first();
    String last();
}
```

Main.java

```
    //   값이 여럿일 시. 순서 무관
    @PersonName(last = "홍", first = "길동")
    private Object seller;
```

```
@Retention(RetentionPolicy.RUNTIME)
public @interface LocsAvail {
    String[] visit();
    String[] delivery();
    String[] quick();
}
```

Main.java

```
    @LocsAvail(
            quick = {"서울", "대전", "강원"}, // {} 안에 작성
            visit = "판교", // 하나만 있을 시 {} 생략 가능
            delivery = {} // 요소가 없을 시 {} 필요
    )
    private Object store;
```

실습

LimitType.java

```
public enum LimitType {
    MAX, MIN
}
```