제 26 강 : 패키지와 단언

※ 학습목표
✓ 자바 패캐지의 개념을 설명할 수 있다.
✓ 패키지 컴파일을 수행할 수 있다.
✓ 외부 라이브러리를 활용할 수 있다.
✓ 단언을 정의하고 활용할 수 있다.

1. 패키지
✓ 자바에서 이야기하는 패키지는 서로 관련 있는 클래스와 인터페이스를 하나의 단위로 묶는 것을 의미하며, 일종의 Library(자료실)라고 할 수 있다.

① 패키지 선언 방법
✓ 패키지(package)선언은 주석문을 제외하고 반드시 소스파일의 첫 줄에 와야 한다.
✓ package 패키지경로명;

② 패키지 사용법
✓ import [패키지경로.클래스명]; 또는 import [패키지경로.*];

③ 패키지 컴파일 방법
✓ javac [옵션] [작업위치] [소스 파일명]

옵션	설명
-d	디렉토리를 의미하며 컴파일되어 생기는 클래스 파일이 저장될 위치를 말하는 것이다. 즉 디렉토리(패키지) 작업을 의미하는 것이며 디렉토리(패키지)를 만들어야 한다면 만들라는 것이다.
.	디렉토리(패키지) 작업은 바로 현재 디렉토리에서 하라는 뜻이다.

[실습] : Editplus를 활용하여 실습해보자.

✓ MyPackOne 클래스 작성

```
1  package mypack.pack;
2
3  public class MyPackOne {
4          public void one() {
5                  System.out.println("MyPackOne클래스의 one메소드");
6          }
7  }
```

✓ MyPackTwo 클래스 작성

```
1  package mypack.pack;
2
3  public class MyPackTwo {
4          public void two() {
5                  System.out.println("MyPackTwo클래스의 two메소드");
6          }
7  }
```

✓ MyPackEx 클래스 작성

```
1  import mypack.pack.MyPackOne;
2  import mypack.pack.MyPackTwo;
3
4  public class MyPackEx {
5          public static void main(String[] args) {
6                  MyPackOne myOne = new MyPackOne();
7                  myOne.one();
8                  MyPackTwo myTwo = new MyPackTwo();
9                  myTwo.two();
10         }
11 }
```

2. static import

✓ JDK5.0이전에서는 상수 또는 static으로 선언된 것들을 사용하려면 해당 [클래스명.상수]등으로 접근해 왔었다.

✓ 하지만 JDK5.0에서부터는 static import를 사용하여 보다 쉽고 빠르게 static 상수 또는 메서드등을 호출할 수 있다.

✓ 이 때문에 작성된 소스 분석이 조금 어려워졌다는 단점도 있다

✓ import static [패키지경로.클래스명.*];

✓ import static [패키지경로.클래스명.상수필드명];

[실습]

```
 1  package tommy.java.exam04;
 2
 3  import static java.lang.Math.*;
 4  import static java.lang.System.out;
 5
 6  public class StaticImportEx {
 7          public static void main(String[] args) {
 8                  int i = (int) (random() * 26 + 65);
 9                  out.println((char) i);
10          }
11  }
```

3. 단언(Assertion)

✓ 프로그래머 자신이 전개하고 있는 코드 내용에서 프로그래머 자신이 생각하고 있는 움직임과 그리고 특정 지점에서의 프로그램상의 설정 값들이 일치하고 있는지를 검사할 수 있도록 하는 것이 바로 Assertion이다.

✓ 예를 들면 어느 특정 메서드의 인자 값은 10이상이어야 한다는 프로그래머의 확고함이 있다고 하자! 이럴 때 Assertion을 사용하여 프로그래머가 주장하는 확고함을 조건으로 명시하고 그 조건을 만족할 때만 코드가 실행할 수 있도록 하는 것이 Assertion이다.

① 단언의 문법

✓ assert [boolean식];

✓ [boolean식]은 항상 true아니면 false인 boolean형의 결과를 가지는 표현식이 되어야 한다. 만약 boolean식의 결과가 false일 경우에는 AssertionError가 발생하여 수행이 정상적으로 진행되지 않는다.

✓ assert [boolean식] : [표현식];

✓ [boolean식]의 결과값이 false일 경우에 [표현식]을 수행한다. 여기에는 일반 값일 수도 있고 특정 메소드를 호출하여 받은 반환 값일 수도 있다. [표현식]에는 문자열로 변환이 가능한 값이 무조건 와야 한다

② 간단한 예문

✓ assert var>10;

✓ assert var<10 : "10보다 작은 값이어야 함!";

✓ assert str.equals("");

✓ assert !str.equals("");

✓ assert str != null : "str에 null값이 들어오면 안됨!";

③ 컴파일 및 실행 법

✓ 실행 할 때 다음과 같이 옵션을 주면서 실행해야 한다.

✓ java -ea [클래스명]

✓ -ea : Enable Assertions라고 해서 단언기능을 사용 가능하게 하는 옵션이다.

✓ -da : Disable Assertions라고 해서 ea의 반대 옵션이다.

[실습] 커맨드 창에서 실행해보자.

```
1  package tommy.java.exam05;
2
3  import static java.lang.System.out;
4
5  public class AssertEx {
6          public void gugu(int dan) {
7                  assert dan > 1 && dan < 10 : "2~9단중 하나를 입력하세요";
8                  out.println(dan + "단");
9                  out.println("----------");
10                 StringBuffer sb = new StringBuffer();
11                 for (int i = 0; i < 9; i++) {
12                         sb.delete(0, sb.length());
13                         sb.append(dan);
14                         sb.append("*");
15                         sb.append(i + 1);
16                         sb.append("=");
17                         sb.append(dan * (i + 1));
18                         out.println(sb.toString());
19                 }
20         }
```

```
21
22          public static void main(String[] args) {
23                  AssertEx at = new AssertEx();
24                  try {
25                          int dan = Integer.parseInt(args[0]);
26                          at.gugu(dan);
27                  } catch (Exception e) {
28                          e.printStackTrace();
29                  }
30          }
31  }
```