# 다른 소스 파일의 코드 참조

관련 파일 : Test06.java, Test06\_other.java

## Ⅰ. 다른 소스 파일의 코드를 참조 시 컴파일 방법

- 다른 소스 파일의 코드를 참조할 때, 컴파일 하는 방법을 알아보자!

- 다른 소스 파일의 코드를 참조하는 이유는?

## Ⅱ. 실습 코드

class Test06 {

public static void main(String[] args) {

Test06\_other.printHello();

}

}

## Ⅲ. 일반적인 컴파일 방법

$프로젝트폴더> javac -d bin -encoding utf8 src/main/java/Test06.java

=> Test06 블록에서는 Test06\_other라는 블록의 코드 사용

=> 그러나 컴파일러가 Test06\_other라는 블록을 못 찾기 때문에 오류 발생

## Ⅳ. 다른 소스 파일을 참조하는 경우 컴파일 하는 방법

-sourcepath [참조하는 소스 파일이 들어 있는 폴더]

$프로젝트폴더> javac ... -sourcepath src/main/java src/main/java/Test06.java

# ■ .java vs class {} vs .class

관련 파일 : Test07\_1.java

## Ⅰ. class {} vs.java vs .class

- .java 파일과 class {}과 .class 파일의 관계를 알아보자!

- .java 파일에 class 블록을 한 개만 두는 이유는?

## Ⅱ. 실습 코드

class Test07\_A {}

class Test07\_B {}

class Test07\_C {}

## Ⅲ . .java 소스 파일의 컴파일 결과

- 각각의 class 블록 별로 .class 파일이 생성

- .class 파일의 이름은 class {} 블록 이름과 동일

# ■ 공개하는 클래스 블록

관련 파일 : Test07\_2.java

## Ⅰ. 공개하는 클래스 블록

- 클래스 블록을 공개하는 방법과 컴파일 하는 방법을 알아보자!

- 클래스 블록을 공개하는 이유?

### Ⅱ. 실습 코드

- 클래스 블록을 공개할 때는 앞에 public 추가

- 에러 발생하는 public class 예 :

Test07\_2.java의 public 클래스명 : public class Test07\_D {}

public class Test07\_2 {}

class Test07\_E {}

class Test07\_F {}

class Test07\_G {}

## Ⅲ. .java 소스 파일의 컴파일 결과

- 컴파일 오류 발생!

- 공개 클래스 블록의 파일명은 반드시 클래스 이름과 같아야 함

## Ⅳ. 결론

- 한 소스 파일에는 한 개의 공개 클래스만 생성 가능

- 한 소스 파일에 비공개 클래스는 여러 개 생성 가능

- 그러나 유지보수를 쉽게 하기 위해서는

한 소스 파일 당 한 개의 클래스를 만드는 것이 좋다.

- 파일 이름으로 어떤 클래스인지 바로 구분할 수 있기 때문

# ■ 코드와 주석

관련 파일 : Test08.java

## Ⅰ. 코드와 주석

- 프로그램 명령을 작성할 때 주석을 다는 방법을 알아보자.

- 주석의 종류와 그 쓰임새를 알아보자!

## Ⅱ. 실습 코드

public class Test08 {

/\*\*

이 변수는 출력하려는 문자열을 저장하고 있다.

\*/

static String message = "Hello!";

/\*\* 프로그램 시작 시, 처음 시작되는 명령어 블록

<b>"프로그램 엔트리(entry)"</b>라 부른다.

\*/

public static void main(String[] args) {

/\* 표준 출력 장치(모니터)로 문자열을 출력하려면,

다음 명령을 작성해야 한다.\*/

System.out.println(message); // 명령어 끝에 반드시 세미콜론(;)을 붙여라!

}

}

## Ⅲ. 주석의 특징

- 주석은 컴파일할 때 제거된다.

- 코드에 대한 개발자의 의견이나 설명을 붙일 때 사용

## Ⅳ. 주석의 종류

### 1) // : 한 줄 주석

- 주석 형태 :

// 주석 내용

- 주석 사용 목적 : 한 줄의 주석을 작성하려 할 때

### 2) /\*\*/ : 여러 줄 주석

- 주석 형태 :

/\*

\*주석 내용

\*/

※ \*란? : wildcard

주석 내용 앞의 \*는 생략 가능

- 주석 사용 목적 : 여러 줄의 주석을 작성하려 할 때

### 3) /\*\*\*/ : javadoc(java document) 주석

- 주석 형태 :

/\*\*

\*주석 내용

\*/

※ \*란? : wildcard

주석 내용 앞의 \*는 생략 가능

- 주석 사용 목적

· 프로그램 개발 완료 후, 각 클래스와 메서드의 설명을 HTML 문서로 생성

· 클래스 선언, 메서드 정의, 변수 선언 앞에 붙임

# ■ 클래스와 패키지

관련 파일 : Test09\_1.java, Test09\_2.java, Test09\_3.java, Test09\_4.java

## Ⅰ. 클래스와 패키지

- 패키지를 사용하여 클래스를 분류하는 방법을 알아보자!

- 패키지를 사용하는 이유?

## Ⅱ. 실습 코드

public class Test09\_1 {

public static void main(String[] ags) {

System.out.println("Hello!");

}

}

## Ⅲ. 패키지 무소속 클래스

- 위와 같이 특정 패키지에 종속되지 않은 클래스를 말한다.

## Ⅳ. 패키지 선언 시점

- 소스 파일 첫 부분

- 이후 선언되는 모든 클래스는 선언된 패키지에 소속

## Ⅴ. 패키지 선언 목적

- 소스 파일의 관리의 용이성 확보

- 패키지 소스 파일 저장용 폴더 생성 후 해당 폴더에 소스파일 저장

**1) 실습 코드**

package a;

public class Test09\_2 {

public static void main(String[] ags) {

System.out.println("Hello!");

}

}

## Ⅵ. 패키지 클래스

- 특정 패키지에 소속된 클래스

- 컴파일러가 패키지 클래스 컴파일 시, 패키지 해당 폴더 자동 생성

### Ⅶ. 패키지 클래스를 실행하는 방법

1) 패키지 무소속 클래스

> java -cp ./build/classes/java/main Test09\_1

2) 패키지 소속 클래스

패키지명을 포함하여 클래스명을 지정한다.

> java -cp ./build/classes/java/main a.Test09\_2

3) CLASSPATH 등록 후 실행하기

(Windows)

> set CLASSPATH=build\classes\java\main

> java a.Test09\_2

(linux & unix)

> export CLASSPATH=./build/classes/java/main

> java a.Test09\_2

### Ⅷ. 주의 사항

- 패키지를 클래스를 실행할 때 패키지 이름까지 포함해야 함

- "-cp" 또는 "-classpath"에 패키지명을 포함하지 말아야 함

ex.

> java -cp ./build/classes/java/main/a Test09\_2 <== 실행 오류!

**1) 실습 코드**

package a.b.c; // 하위 패키지는 점(.)으로 구분한다.

public class Test09\_3 {

public static void main(String[] ags) {

System.out.println("Hello!");

}

}

### Ⅸ. 패키지 클래스 실행

> java -cp ./build/classes/java/main a.b.c.Test09\_3

# ■ 패키지 명명 규칙

관련 파일 : Test09\_1.java, Test09\_2.java, Test09\_3.java, Test09\_4.java

## Ⅰ. 패키지 명명 규칙

- 패키지 이름이 겹치지 않도록 짓는 방법!

- 패키지 이름으로 왜 도메인명을 사용하는가?

- 왜 도메인명을 꺼꾸로 사용하는가?

### 1) 실습 코드

package kr.co.bitcamp.java100;

public class Test09\_5 {

public static void main(String[] ags) {

System.out.println("Hello!");

}

}

### Ⅱ. 명명 양식

1) 보통 패키지명이 겹치지 않도록 하기 위해 도메인명을 사용한다.

2) 폴더 관리를 쉽게 하기 위해

상위 도메인명이 상위 폴더가 될 수 있도록

도메인명을 꺼꾸로 사용한다.

예) com.xxx.제품명.하위패키지