**패키지와 임포트**

|  |  |
| --- | --- |
| 1 | 다음 설명 중 틀린 것은?  (1) 클래스를 하나의 디렉토리에 계속 누적하다 보면 클래스명의 중복을 피하기란 어렵다  (2) 클래스 명의 중복을 원천적으로 방지하기 위해서는 서로 다른 디렉토리로 구분하여 저장하면 된다.  (3) 관련성 있는 클래스들을 디렉토리 단위로 구분하여 관리하는 방법을 자바에서는 패키지라 한다.  (4) 클래스 파일명이 동일하더라도 패키지가 다를 경우, 중복 문제는 발생하지 않는다.  (5) ~~서로 다른 패키지에 존재하는 클래스간의 접근은 불가능하다.~~  (6) **패키지명은 소문자로 하되, 주로 개발 회사의 도메인을 거꾸로 표기하는 것이 관례이다**  (7) ~~같은 패키지 내에 있는 클래스들 끼리는 언제든지 서로의 데이터를 접근할 수 있다.~~  (8) 패키지의 주요 목적은 파일명의 중복을 피하며, 관련성 있는 클래스들끼리 묶어 관리하기 위함이다. |
| 2 | 아래 클래스에 대한 설명 중 올바르지 않은 것은?  **package** pack1;  **class** Car{  **int** price=300;  }  (1) javac –d 옵션으로 컴파일을 수행하면 pack1 이라는 패키지가 자동으로 생성되면서 그 안에  Car.class를 생성하여 준다.  (2) ~~Car 클래스 자체는 현재 외부에서 접근이 불가능한 상태이다.~~ (default: 같은 패키지만 접근 허용)  (3) ~~price 멤버변수에 접근 제한자가 지정되지 않으면 자동으로 private 으로 지정된 것과 같다.~~  (4) 클래스란 결국 쓰려고 정의한 것이기 때문에 패키지 안에 둔 클래스를 외부에서 사용하려면 클래스를  public 으로 공개해야 한다. |
| 3 | 다음 코드에 대한 설명 중 틀린 것은?  **package** test;  **import** hardware.Phone; (A)  **class** UsePhone{  **public** **static** **void** main(String[] args) {  Phone phone =**new** Phone();  }  }  (1) ~~(A)에서의 Phone 은 Phone.java 를 임포트하는 명령어이다.~~ (.class)  (2) (A)는 시스템상의 완전한 경로가 아니므로, hardware 패키지가 존재하는 경로는 환경변수에 등록  되어 있어야 한다.  (3) 위(2)번에서 사용되는 환경변수명은 classpath 이며 ~~개발 시 사용중인 OS가 직접 클래스 파일을~~  ~~해석 및 실행하는 데 사용된다.~~  (4) 자바의 .class 파일은 오직 JVM만이 해석 및 실행할 수 있는 바이트 코드라 불리는 특별한 파일이므로  OS가 직접 해석 및 실행하지 못한다.  (5) classpath 란 JVM이 사용할 .class 클래스 파일의 위치를 등록해놓기 위한 환경변수이며, classpath  를 사용하지 않을 경우 자바 소스 코드에 경로를 명시하게 되어 Write Once Run Anywhere 슬로건  에 위배된다 |
| 4 | Car.class 파일이 D:\study\pack1 에 들어있다고 가정할 때 import 문으로 올바른 것을 모두 고르면?  (1) ~~import D:/study~~  (2) ~~import study /pack1~~  (3) ~~import pack1;~~  (4) **import pack1.Car;**  (5) **import pack1.\*** |