**패키지와 임포트**

|  |  |
| --- | --- |
| 1 | 다음 설명 중 틀린 것은?  (1) 같은 디렉토리내의 파일명은 OS차원에서도 중복을 허용하지 않으므로 클래스를 하나의 디렉토리에  계속 누적하다 보면 클래스 명의 중복을 피하기란 쉽지 않다. O  (2) 클래스 명의 중복을 원천적으로 방지하기 위해서는 서로 다른 디렉토리로 구분하여 저장하면 된다. O    (3) 관련성 있는 클래스들을 디렉토리 단위로 구분하여 모아놓는 관리방법을 패키지라고 한다.O  (4) 클래스 파일명이 동일하더라도 패키지가 다를 경우엔 중복의 문제는 발생하지 않는다.O  (5) 서로 다른 디렉토리에 존재하는 클래스간의 접근은 불가능하다.X  (6) 패키지화하여 관리하는 목적은 중복도 피하고, 관련성 있는 클래스들 간의 관리 목적 때문이다.O |
| 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.O  (2) 만일 이미 pack1 이라는 패키지가 존재한다면 중복 생성되지 않는다.O  (3) 이 클래스는 외부에서 절대로 접근이 불가능한 클래스이다.O    (4) price 멤버변수에 접근 제한자가 지정되지 않으면 자동으로 private 으로 지정된 것과 같다. |
| 3 | 위 2번의 클래스가 실제 하드디스크 D:\minjava\pack1 에 들어있다고 가정할 때 import 문으로 올바른 것을 모두 고르면?  (1) import D:/ minjava X(슬로건위배)  (2) import minjava /pack1 X  (3) import pack1; X(클래스명 미명시)  (4) import D:/minjava/pack/Car X(슬로건위배)  (5) import pack1.Car O  (6) import pack1.\*O(맞지만 개발시 지양대상..) |
| 4 | 다음 설명 중 올바르지 않은 것은?  (1) 클래스를 하나의 디렉토리에 몰아넣으면 클래스 명이 중복될 경우 중복 파일 문제가 발생할 수 있다.O  (2) 관련 있는 디렉토리로 비슷한 업무의 클래스들을 묶어 놓는 것은 관리상 효율성이 있다.O  (3) 패키지 명은 소문자로 하되, 주로 개발 기업의 도메인을 거꾸로 기재하여 배포한다.O  (4) 같은 패키지 내에 있는 클래스들 끼리는 언제든지 서로의 데이터를 접근할 수 있다.X |