03-07 [Java]

소유자	종수 김
태그	

[Package]

만약 프로그램이 커지고 이를 종류 별로 구분해야 한다면 ?

- user
  - UserController
  - UserService
  - UserRepository
- product
  - ProductController
  - ProductService
  - ProductRepository

비슷한 종류의 파일을 패키지로 묶는 트리 형태의 구조

- 패키지가 계층 구조를 이루더라도, 각각의 패키지는 각각 다른 관계없는 패키지.

- 같은 패키지 레벨만 패키지 구조를 생략할 수 있고, 그 외에 경우에는 패키지 구조를 명시해줘야 함.

```
package pack;


public class PackageMain1 {
    public static void main(String[] args) {
        Data data = new Data();
        pack.a.User user = new pack.a.User();

    }
}
```

import

패키지 구조를 일일이 명시하는게 너무 불편해.
import를 사용하면, 다른 패키지에 있는 클래스를 패키지 구조 명시 없이 사용 가능.

```
package pack;


import pack.a.User;
import pack.a.User2;

public class PackageMain1 {
    public static void main(String[] args) {
        Data data = new Data();
        User user = new User();
        User2 user2 = new User2();
    }
}
```

- *
  - 해당 패키지 하위 레벨의 모든 패키지를 선언하는 것.
- 동일한 클래스 이름을 가진 객체가 여러 패키지에 나눠져 있다면 ?
  - import가 선언된 클래스만 패키지 구조를 명시하지 않아도 됨.
  - 그 외에 모든 동일한 이름 객체는 패키지 구조를 명시해줘야 함.

```
package pack;

import pack.a.User;

public class PackageMain2 {
    public static void main(String[] args) {
        User userA = new User();
        pack.b.User userB = new pack.b.User();
    }
}
```

패키지 규칙

1. 패키지 이름과 위치는 폴더(디렉토리) 위치와 같아야 한다. ⇒ 필수
2. 패키지 이름은 모두 소문자를 사용한다 ⇒ 관례
3. 패키지 이름의 앞 부분에는 일반적으로 회사의 도메인 이름을 거꾸로 사용한다. ⇒ 관례
   Ex) com.company.myapp
   - 많은 라이브러리가 함께 사용되면 같은 패키지에 같은 클래스 이름이 존재할 수도 있음.
   - 도메인 이름을 거꾸로 사용하면 이런 문제를 방지 가능.
   - 오픈소스나, 라이브러리를 만들어 제공한다면 꼭 지키는 것이 좋음.

패키지 활용

Ex) src 하위.

- com.helloshop
  - user
    - User
    - UserService
  - product
    - Product
    - ProductService
  - order
    - Order