15강. 컴포넌트, 배포, 패키지 다이어그램

컴퓨터과학과 김희천 교수

목차

Chapter. 1

개요

1. 컴포넌트 다이어그램

컴포넌트

- 시스템의 빌딩블록으로 '관리되고 재사용 가능하며 대체 가능하고 조합될 수 있는 소프트웨어 부품'
- 컴포넌트는 자주 사용되는 주요 기능을 모듈화한 소프트웨어 부품

✚ 컴포넌트 다이어그램

- ✕ 컴포넌트의 구성과 컴포넌트 간 관계를 보여줌
- ✕ 시스템이 작은 부분들로 이루어져 있음을 설계/구현 관점에서 보여줌
- ✕ 4+1 뷰에서 프로그래머나 소프트웨어 관리자 관점의 개발 뷰에 해당

2. 배포 다이어그램

배포

- 소프트웨어 산출물(artifact)를 특정 노드에 배정하는 것

✚ 배포 다이어그램

- ✕ 소프트웨어 파일이 실제로 어떤 하드웨어에 배치되어 실행되는지를 보여줌
- ✕ 최종적 하드웨어 구성과 소프트웨어의 물리적 배치를 보여줌
- ✕ 4+1 뷰에서 물리적 뷰에 해당

3. 패키지 다이어그램

패키지

- 관련이 있는 UML 요소를 그룹화하기 위한 요소
- 클래스나 컴포넌트들을 그룹화시켜 구조화할 수 있음

✚ 패키지 다이어그램

- ✕ 패키지의 구성과 패키지 간의 의존성을 보여줌
- ✕ UML 4+1 뷰에서 개발 뷰에 해당

Chapter. 2

컴포넌트 다이어그램

1. 컴포넌트

- **캡슐화된 재사용 가능한 소프트웨어 부품**
- 자주 사용되는 주요 기능이 모듈화된 것
- 독립적으로 개발되고 배포될 수 있는 단위. 재사용 부품으로 대개 하나 이상의 클래스로 구현됨(C#의 dll 파일이나 Java의 jar 파일)
- 컴포넌트와 컴포넌트는 인터페이스를 통해 연결됨
- 컴포넌트 기반으로 소프트웨어를 만든다는 것은 작은 단위의 컴포넌트들을 묶어 조립하는 것
- 컴포넌트의 크기는 하나의 클래스에서 서브시스템에 이르기까지 다양함

2. 컴포넌트 표현

01

블랙박스 관점

- 컴포넌트의 인터페이스를 중심으로 한 사용 관계를 다룸
- 컴포넌트를 블랙박스로 다룰 수 있음
- 비교하여, 클래스 다이어그램은 클래스의 내부 속성이나 메소드를 자세히 다룸

02

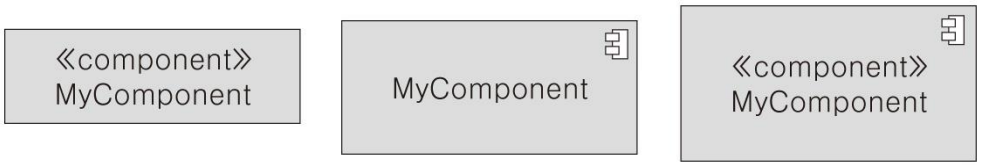
화이트박스 관점

- 특정 컴포넌트의 내부 구성을 보여줌
- 어떤 클래스나 서브 컴포넌트 및 인터페이스를 사용하여 기능이 구현되었음을 보여줌

3. 컴포넌트 표기

UML에서 컴포넌트 표기법

- 컴포넌트의 의미와 표기법은 클래스와 유사함
- 사각박스로 표시되며 《component》 스테레오 타입을 넣음
- 우상단에 컴포넌트 아이콘을 표시할 수 있음
- 컴포넌트가 서브시스템인 경우에는 《subsystem》을 사용



4. 컴포넌트의 인터페이스

+ 컴포넌트의 인터페이스는 다른 부분과 결합되는 부분

제공 인터페이스 (provided interface)

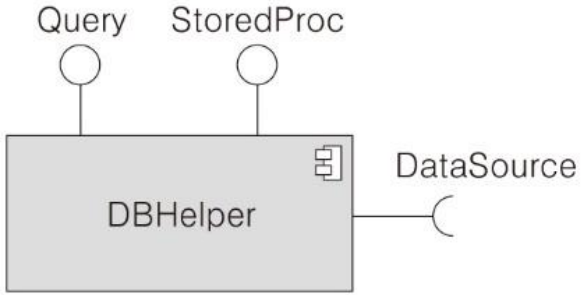
- 기능을 구현하고 외부에 서비스를 제공하기 위해 노출한 것
- 외부의 다른 컴포넌트가 이 인터페이스를 통해 기능을 사용하게 됨

요구 인터페이스 (required interface)

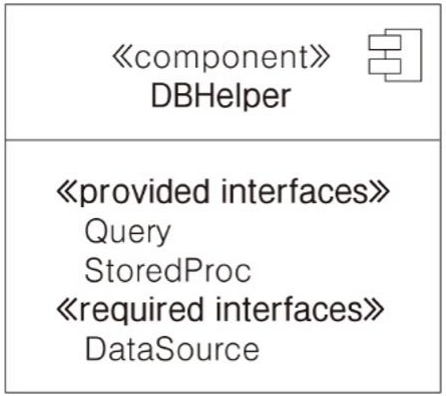
- 컴포넌트가 정상적으로 동작하기 위해 필요로 하는 기능
- 이것을 구현한 외부의 컴포넌트가 필요하며, 외부 컴포넌트 입장에서는 제공 인터페이스임

5. 컴포넌트의 인터페이스 표기 (1/2)

- 막대 사탕과 소켓 표기법 (Ball and Socket Notation)
  - 예) Query와 StoredProc은 제공 인터페이스이며 DataSource는 요구 인터페이스

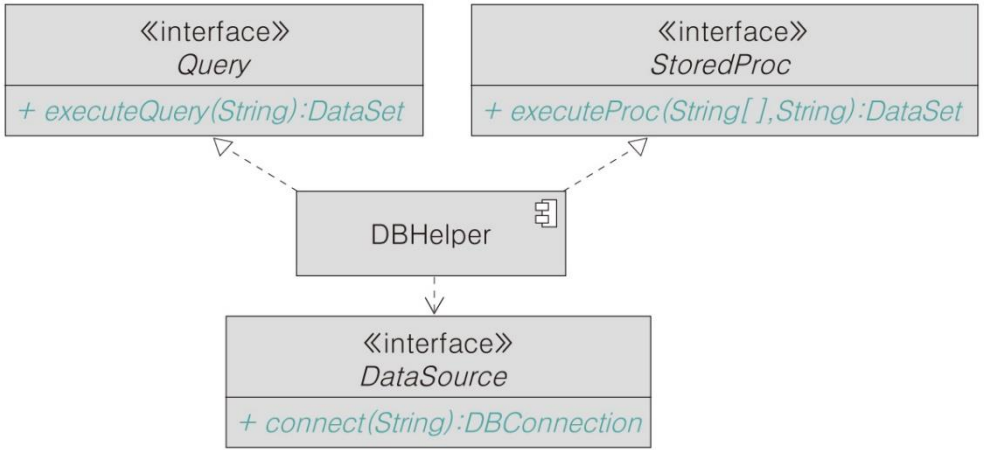


- 클래스 표기법과 유사한 방법



5. 컴포넌트의 인터페이스 표기 (2/2)

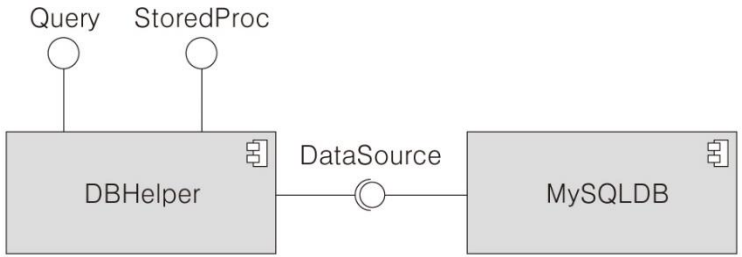
- 다른 표기법이 있으며 《interface》 스테레오 타입을 사용
  - 제공 인터페이스는 구현을 의미하는 점선의 상속 관계로 컴포넌트와 연결
  - 요구 인터페이스는 컴포넌트와 의존 관계의 점선 화살표로 연결



6. 컴포넌트 연결

제공 인터페이스와 요구 인터페이스의 연결

- 인터페이스를 필요로 하는 컴포넌트와 그 인터페이스를 구현해 주는 다른 컴포넌트와 연결하는 것
- 예) DBHelper와 MySQLDB가 DataSource를 통해 연결됨. 컴포넌트가 DataSource 인터페이스를 제공한다면, 데이터베이스 종류와 상관없이 DBHelper가 사용할 수 있음. DBHelper를 사용하면 Query와 StoredProc를 이용하여 데이터베이스에 접근할 수 있음.



7. 컴포넌트의 내부 구조 명세 (1/3)

✚ 컴포넌트의 내부 구성 정보를 명세하는 것

– 컴포넌트의 구성 요소들을 포함시키고 그들의 관계를 표시

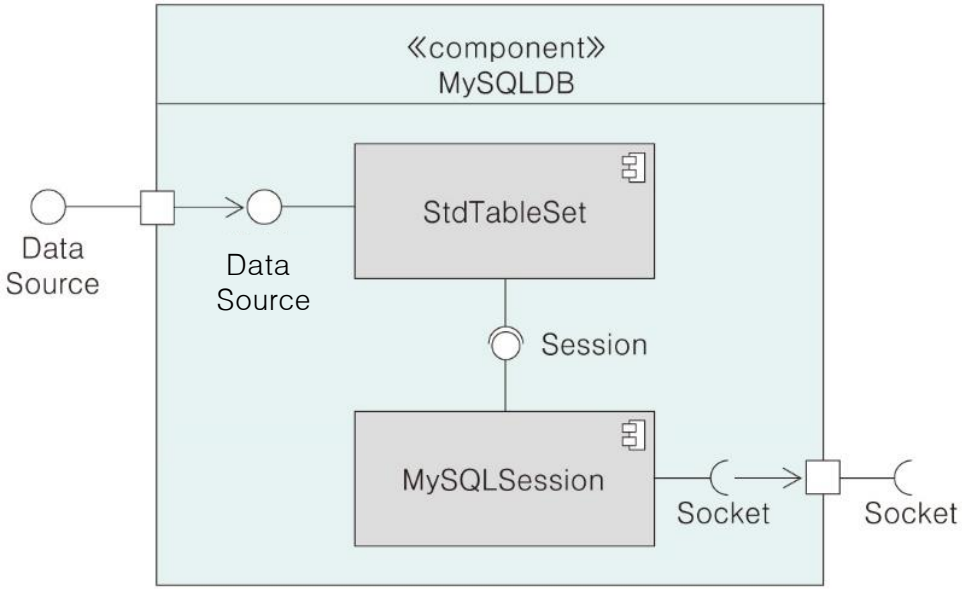
✕ 포트(port)

- 컴포넌트 내부에 존재하는 요소들과 컴포넌트의 제공/요구 인터페이스와의 관계를 맺어주는 지점
- 컴포넌트의 경계에 속인 빈 작은 사각형으로 표시

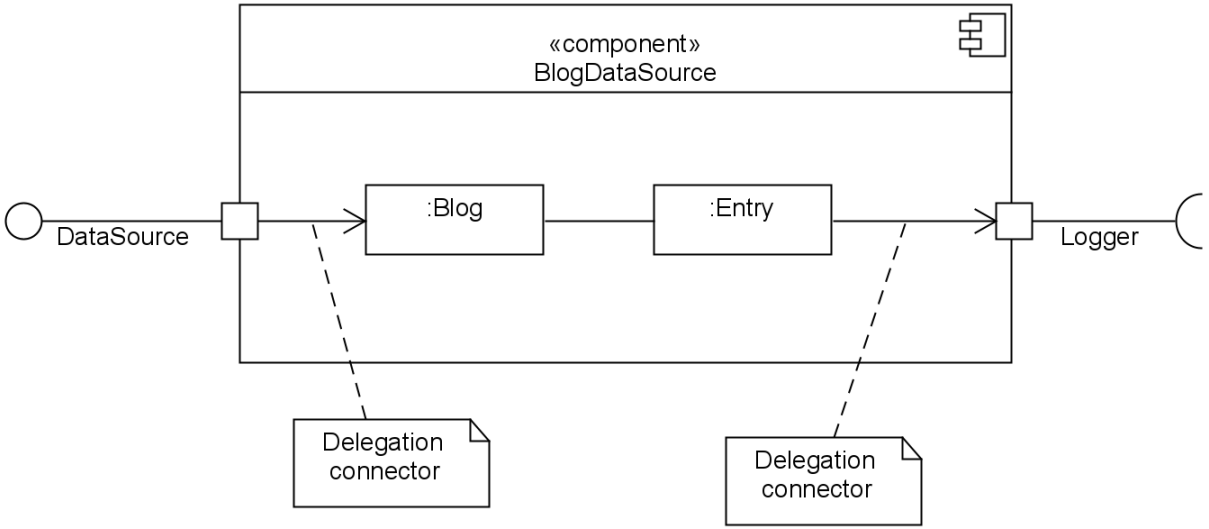
✕ 위임 연결자(delegation connector)

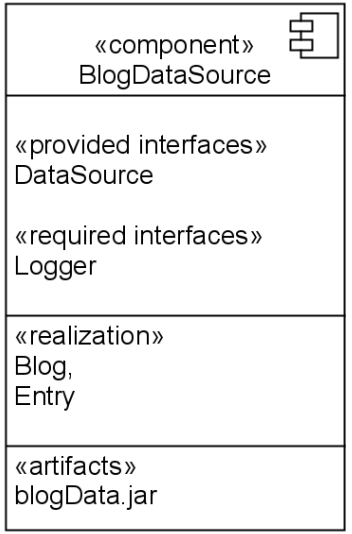
- 컴포넌트의 내부 요소와 포트를 연결해 주는 화살표
- 컴포넌트를 사용하는 과정에서 발생하는 메시지의 흐름을 표현
- 제공 인터페이스의 사용 요청이 오면, 포트를 통해 내부 구현 요소로 연결됨
- 외부 컴포넌트를 사용하는 요구 인터페이스의 경우, 내부 요소로부터 포트를 통해 요청 메시지가 나감

7. 컴포넌트의 내부 구조 명세 (2/3)



7. 컴포넌트의 내부 구조 명세 (3/3)





Chapter. 3

배포 다이어그램

1. 배포 다이어그램

- **하드웨어 요소와 소프트웨어 산출물의 배포를 표현함**
  - 실행 상황의 물리적 시스템 아키텍처를 보여줌
  - 실행 파일이나 라이브러리 파일과 같은 소프트웨어 산출물과 이것을 실행하는 하드웨어를 시각적으로 보여줌
  - 소프트웨어 산출물의 물리적 배치를 모델링하기 위해 사용됨
- **시스템 설계 단계에서는 물리적 배치를 결정하기 어려우나 개략적인 시스템 구조는 고려할 수 있음**
  - 하드웨어 구성 계획에 배포 다이어그램을 사용

2. 노드 (1/2)

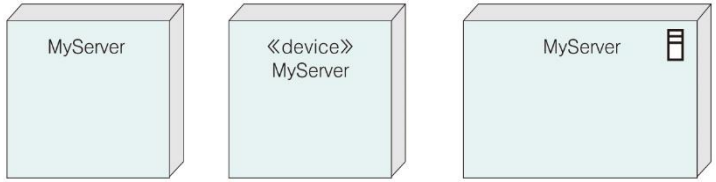
✚ 구성 요소들을 동작시키기 위해 필요한 환경을 제공하는 하드웨어 또는 소프트웨어

- 노드는 실행 시간에 존재하는 물리적 요소로 실행에 필요한 자원을 표현
- 대부분의 경우 노드는 하드웨어지만 운영체제, 웹 서버, 응용 서버 등은 실행 환경을 제공하는 노드가 될 수 있음

✕ 표기법

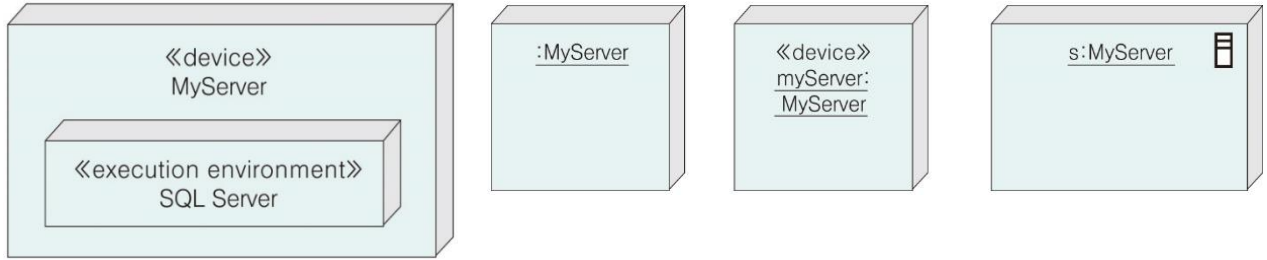
- 육면체로 표시하며 크게 《device》와 《executionEnvironment》로 구분하여 표시함
- 《computer》, 《server》, 《printer》 등을 사용할 수 있음
- 적당한 아이콘을 우상단에 표시할 수 있음

2. 노드 (2/2)



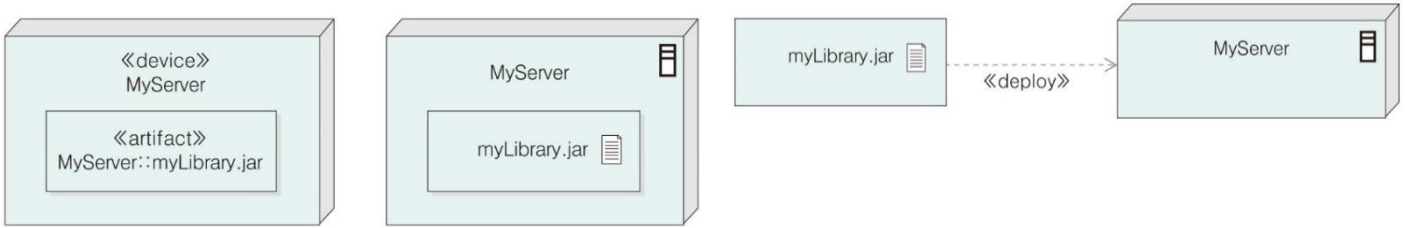
예

- SQL Server라는 실행 환경이 MyServer 노드에 배치된 모습(좌측 그림)
- 같은 종류의 여러 인스턴스 노드를 표현. 노드 이름에 밑줄을 그어 노드 인스탄스를 표현함(우측 그림)



3. 소프트웨어 산출물(artifact)의 배포

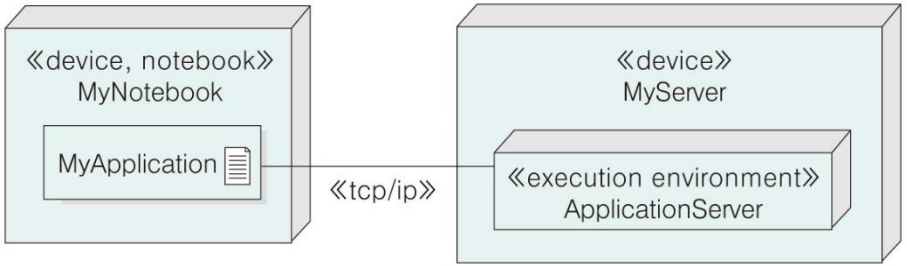
- 노드는 실행 파일이나 라이브러리와 같은 물리적 소프트웨어 산출물(artifact)을 포함할 수 있음
  - 물리적 소프트웨어 산출물이란 exe, dll, jar, class 파일 및 Java와 C의 소스 파일 그리고 xml, txt 파일 등의 모든 물리적 파일
- 소프트웨어 산출물을 표시하기 위해《artifact》스테레오 타입을 가진 사각형을 사용하고 아이콘을 표시할 수 있음
  - 산출물을 노드에 포함시켜 표현함(좌측 그림)
  - 《deploy》스테레오 타입을 가지는 의존 관계로 나타낼 수도 있음(우측 그림)



4. 노드 연관

노드들 간의 통신을 표현

- 통신 경로를 의미함
- 예) 노트북 컴퓨터 안의 응용 프로그램이 서버로부터 TCP/IP를 이용하여 자료를 요청하는 경우



Chapter. 4

패키지 다이어그램

1. 패키지 다이어그램

✚ 패키지

- 관련 있는 UML 요소들을 그룹으로 만들어 구조화한 요소
- 시스템, 서브시스템, 모듈 간 관계와 구조를 보일 때 사용
- 대부분의 UML 요소들에 적용할 수 있음
- 논리적 연관성이 있는 클래스들을 묶어 패키지로 관리(Java의 패키지나 C#의 네임스페이스)

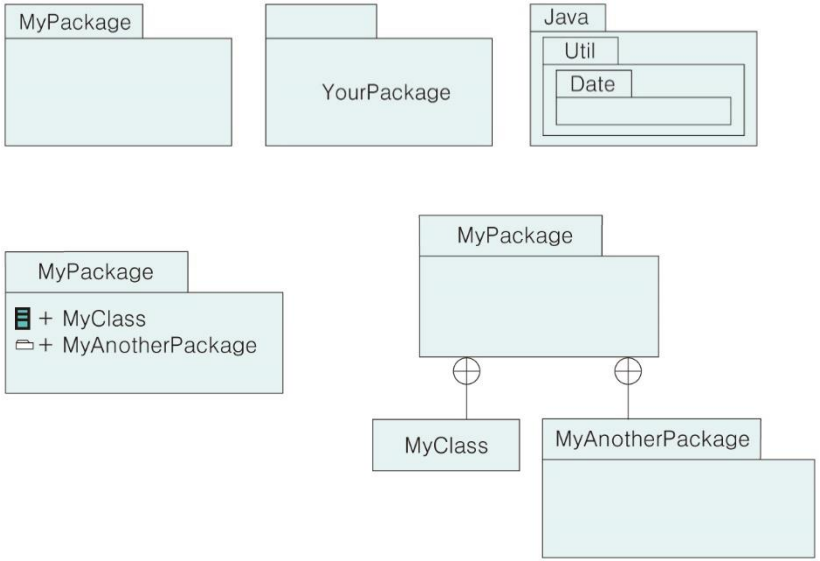
✕ 패키지 다이어그램

- 패키지 자체와 패키지 간의 의존 관계를 표현함

2. 패키지 표기 (1/2)

탭이 달린 사각형으로 표시

- 파일 폴더와 유사함
- 구성 요소들을 패키지 안에 포함시키거나 포함 관계를 표시하여 표현함



2. 패키지 표기 (2/2)

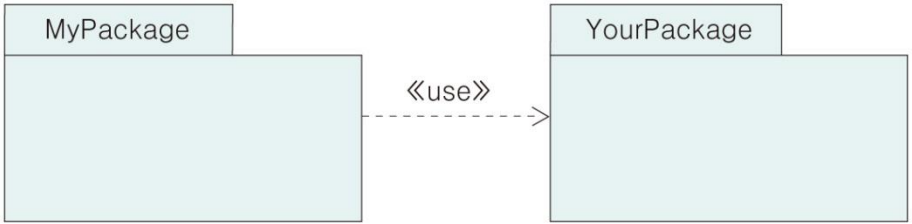
- **특정 패키지에 존재하는 요소를 표현하는 방법**
  - 패키지 내부에서는 이름만을 사용하여 요소를 표현할 수 있음
  - 동일 패키지 내에서 두 요소의 이름은 같을 수 없음
  - 패키지 안에 다른 패키지를 포함할 수 있음
  - 하나의 패키지 안에서 다른 패키지를 참조하려면 절대 경로를 사용해야 함
    - 예) MyPackage::MyClass
    - 예) MyPackage::MyAnotherPackage::MyAnotherClass

3. 접근 제어

- 패키지에 포함된 요소들에 대해 접근이 가능한 범위를 표시
- private 요소는 패키지 내부에서만 사용가능하며,
  public 요소는 어디서나 사용 가능

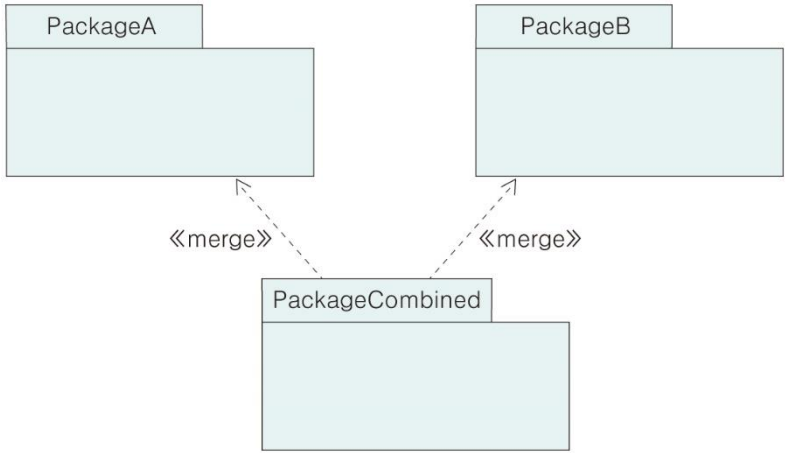
4. 패키지 간의 의존 관계(1/2)

- 《use》
  - 패키지 내 한 요소가 다른 패키지의 요소에 의존하는 경우 발생함
  - 한 패키지가 다른 패키지를 사용하는 관계
  - 패키지의 변경은 이것에 의존하는 다른 패키지에 영향을 줌
  - 유지보수성을 고려하면 순환 고리 형태의 의존 관계를 제거해야 함



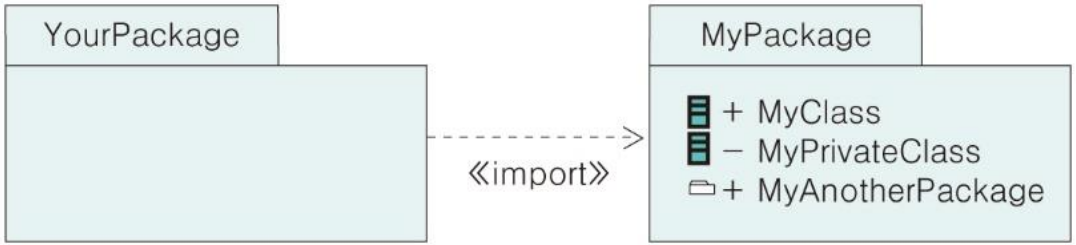
4. 패키지 간의 의존 관계(2/2)

- 《merge》
  - 한 패키지의 내용을 확장하여 새로운 패키지를 정의할 때 발생하는 의존 관계
  - 두 패키지의 내용을 합쳐서 하나의 새로운 패키지를 만들 때도 사용

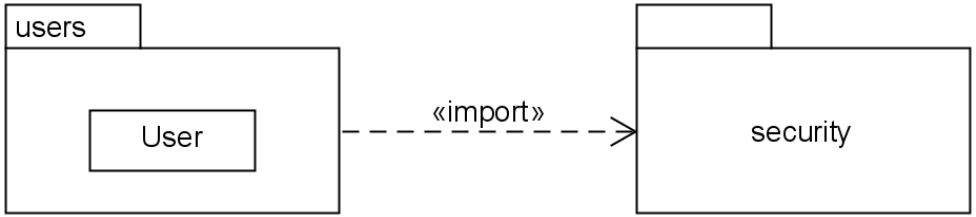


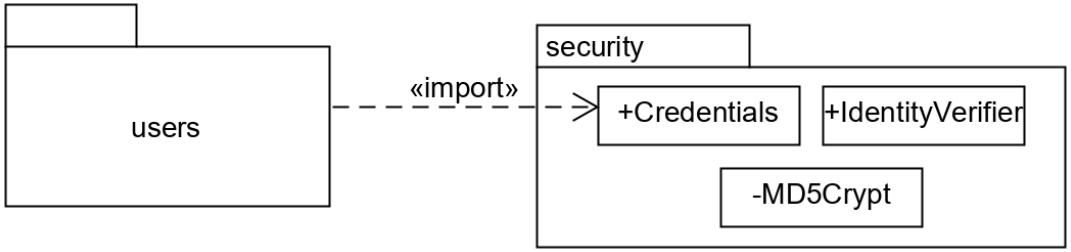
5. 특수한 의존 관계 (1/4)

- 《import》
  - 패키지가 다른 패키지(또는 개별 요소)를 import하는 것
  - import하는 네임스페이스에 import되는 패키지에 속한 요소의 이름이 추가됨
  - import가 선언되면, import 하는 패키지의 요소들에서 import 되는 패키지의 요소들을 사용할 때, 절대 경로를 사용하지 않고, 요소의 이름만으로 참조할 수 있음
  - 예 : YourPackage의 요소들에서 MyPackage에 속한 요소를 사용할 때 이름만으로 참조 가능



5. 특수한 의존 관계 (2/4)



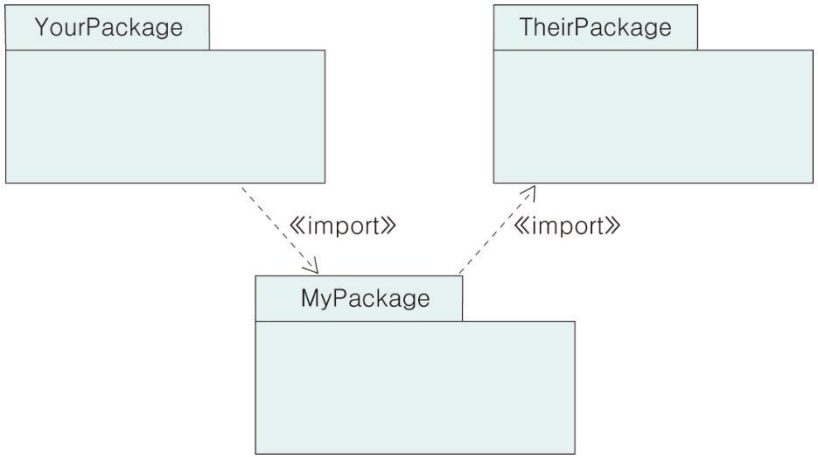



```
import security.*;

class User {
        Credentials credentials;
        ...
}
```

```
import security.Credentials;

class User {
        Credentials credentials;
        ...
}
```

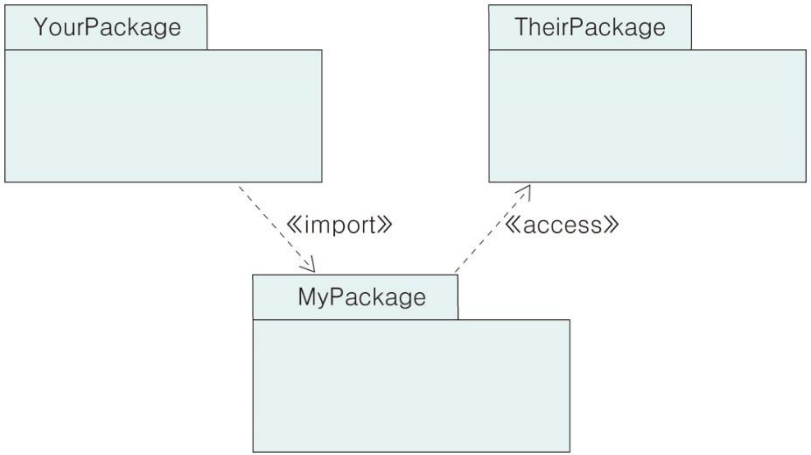
5. 특수한 의존 관계 (3/4)

- 《import》의 성질
  - 《import》 관계가 연속으로 발생하는 경우에 《import》는 public 성격의 import
  - import 되는 요소들은 import 하는 패키지에서 public 요소가 됨.
    즉 import하는 네임스페이스의 외부에서 접근 가능
  - 예) YourPackcage는 MyPackage를 통해 TheirPackage의 요소를 볼 수 있음



5. 특수한 의존 관계 (4/4)

- 《access》
  - 《access》는 private 성격의 패키지 import
  - import 되는 요소들은 import 하는 패키지에서 private 요소가 됨
  - 예) YourPackcage는 MyPackage를 통해 TheirPackage의 요소를 볼 수 없음



다음강의

한 학기 동안 수고 하셨습니다