

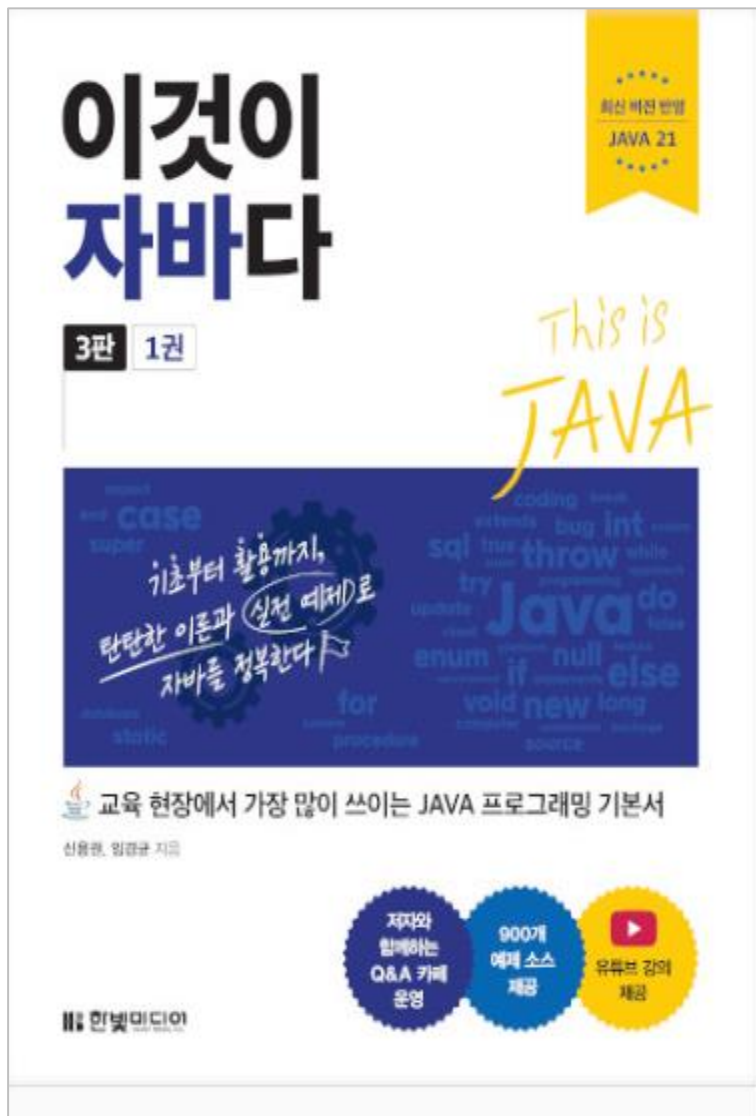
# 객체지향 프로그래밍



[교재6장 보충교재]  
클래스(class)

고제욱 교수

2024.10.04



## 이것이 자바다(3판)

교육 현장에서 가장 많이 쓰이는 JAVA 프로그래밍 기본서

한빛미디어 집필서 판매중



- 저자 : 신용권, 임경균
- 출간 : 2024-04-01
- 페이지 : 1112 쪽
- ISBN : 9791169212298
- 물류코드 : 11229
- 구판정보 : 이 도서는 <이것이 자바다(개정판)>의 개정판입니다.

구판 정보 보기



여기서 말하는 “교재”는 “이것이 자바다(3판)” (2024.04.01) - 신용권 지음, 한빛미디어(한빛아카데미)”을 의미함.

- 저자 : 신용권
- 최초출간 : 2024-04-01
- 페이지 : 1112 쪽
- ISBN : 9791169212298
- 물류코드 : 11229

[https://www.hanbit.co.kr/store/books/look.php?p\\_code=B1795688037](https://www.hanbit.co.kr/store/books/look.php?p_code=B1795688037)

<https://product.kyobobook.co.kr/detail/S000212853100>

# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재6장 보충교재  
클래스(class)

2개 이상의 클래스가 선언된  
소스파일 컴파일시 결과

C드라이브에서  
C:\Temp\test 디렉토리에  
Car.java 파일을 생성함.

The screenshot shows a Windows File Explorer window titled 'test'. The address bar shows the path '내 PC > Basecamp (C:) > Temp > test'. The left sidebar shows the 'Basecamp (C:)' folder selected. The main pane shows a single file named 'Car.java' with a size of 1KB, created on 2021-02-07. The file is highlighted with a red box.

Car.java 파일의 소스코드 내용(아래)

```
public class Car {  
  
}  
  
class Tire {  
  
}
```

<< 주의사항 >>

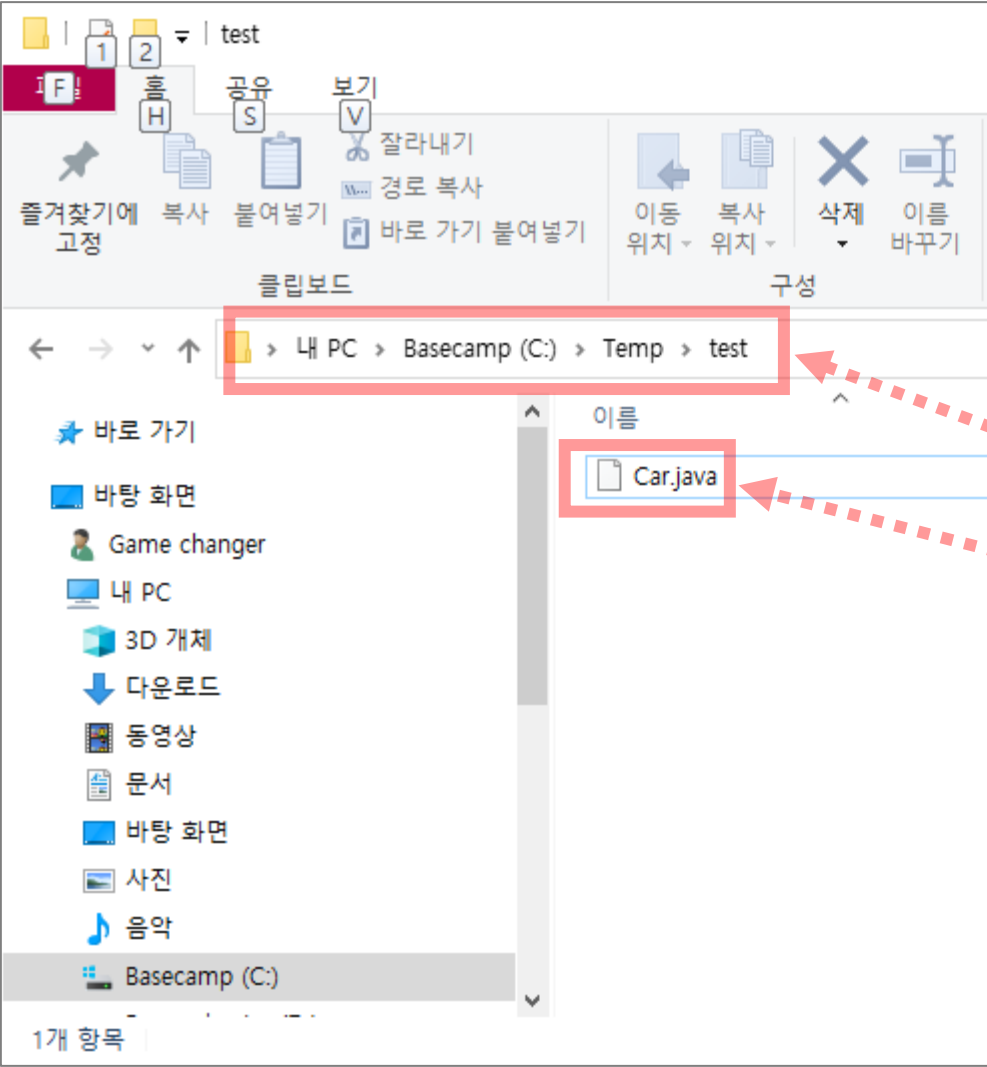
여기서, .java 파일확장자를 갖는 파일 이름과 이 파일 안에서 정의되는 클래스 이름은 대소문자가 모두 정확하게 동일해야 한다. (교재 p192)

Car.java

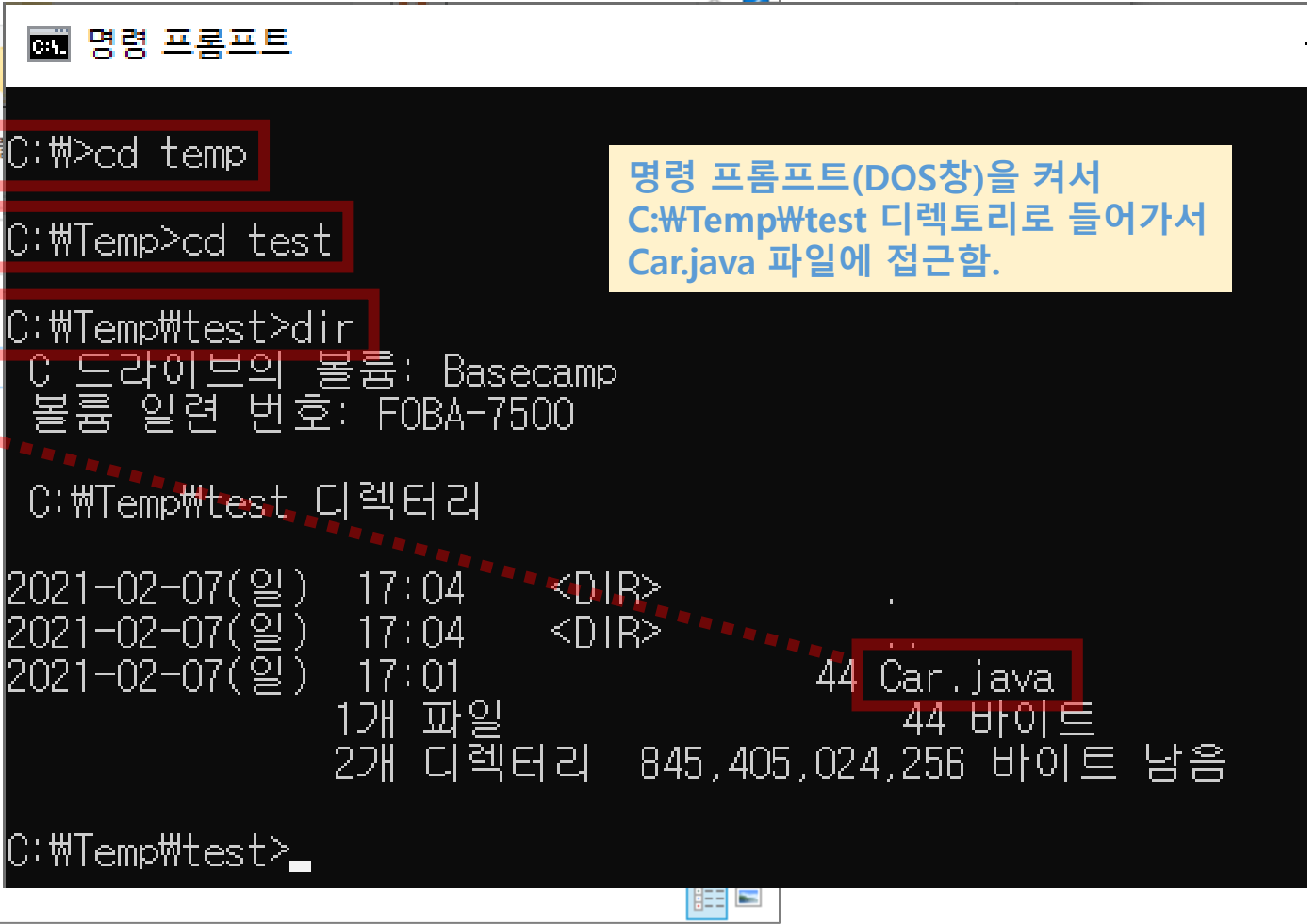
| 이름       | 수정한 날짜               | 유형      | 크기  |
|----------|----------------------|---------|-----|
| Car.java | 2021-02-07(일) 오후 ... | JAVA 파일 | 1KB |

```
public class Car {  
    .....  
}
```

Car.java 파일의 소소코드 내용(위)



File Explorer window showing the path: > 내 PC > Basecamp (C:) > Temp > test. The file Car.java is highlighted in the right pane.



명령 프롬프트

```
C:\>cd temp
C:\Temp>cd test
C:\Temp\test>dir
C 드라이브의 볼륨: Basecamp
볼륨 일련 번호: FOBA-7500

C:\Temp\test 디렉터리

2021-02-07(일)  17:04    <DIR>
2021-02-07(일)  17:04    <DIR>
2021-02-07(일)  17:01                44 Car.java
                        1개 파일                44 바이트
                        2개 디렉터리  845,405,024,256 바이트 남음

C:\Temp\test>
```

명령 프롬프트(DOS창)을 켜서 C:\Temp\test 디렉토리로 들어가서 Car.java 파일에 접근함.

파일 | 홈 | 공유 | 보기

즐거찾기에 고정 | 복사 | 붙여넣기 | 클립보드

잘라내기 | 경로 복사 | 바로 가기 붙여넣기

javac로 Car.java 파일을 컴파일한 결과,  
C:\Temp\test 디렉토리에  
Car.class 와 Tire.class 라는  
2개의 파일(2개의 바이트코드 파일)이  
생성된 것을 확인할 수 있다.

열기 | 편집 | 히스토리 | 기

모두 선택 | 선택 안 함 | 선택 영역 반전 | 선택

← → ↕ ↑

내 PC > Basecamp (C:) > Temp > test

test 검색

| 이름         | 수정한 날짜               | 유형       | 크기  |
|------------|----------------------|----------|-----|
| Car.class  | 2021-02-07(일) 오후 ... | CLASS 파일 | 1KB |
| Car.java   | 2021-02-07(일) 오후 ... | JAVA 파일  | 1KB |
| Tire.class | 2021-02-07(일) 오후 ... | CLASS 파일 | 1KB |

바로 가기

바탕 화면

Game changer

내 PC

3D 개체

다운로드

3개 항목

명령 프롬프트

```
C:\Temp\test> javac Car.java  
C:\Temp\test>
```



# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재6장 보충교재  
클래스(class)

패키지 선언이 포함된  
클래스 컴파일



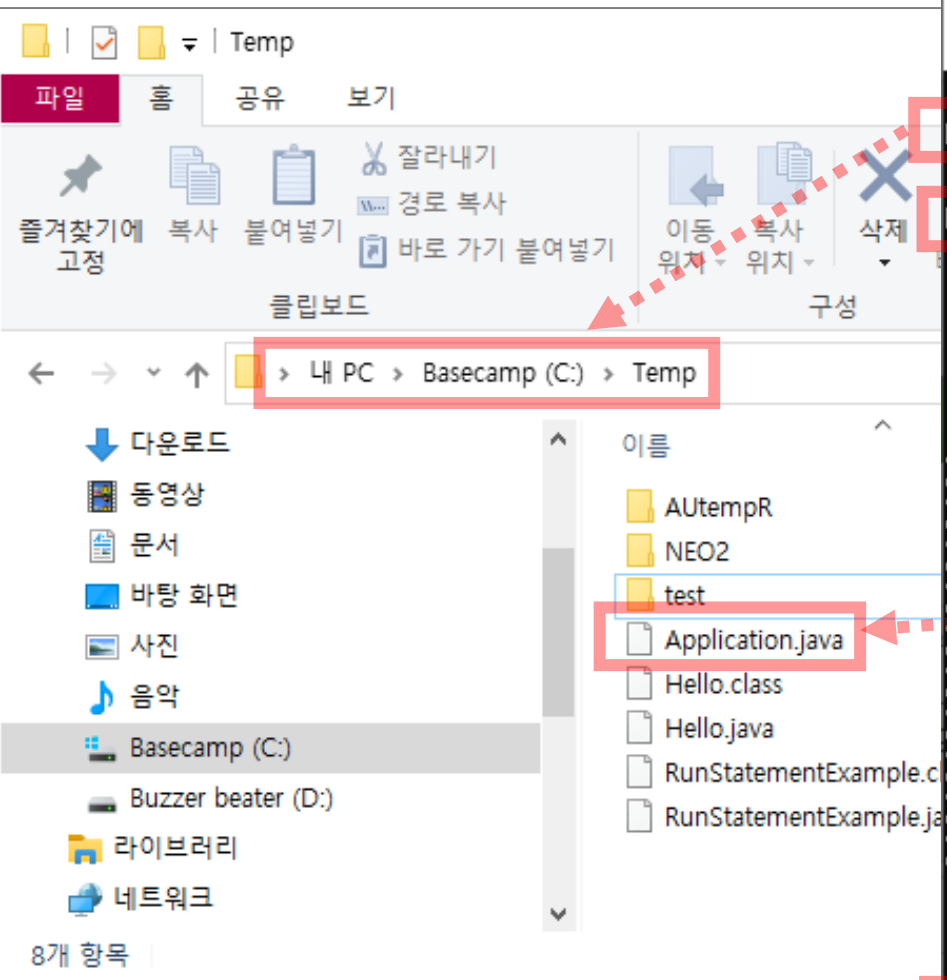
C:\Temp 디렉토리에 Application.java 라는 소스코드가 있다고 가정하자

| 이름               | 수정한 날짜               | 유형       | 크기  |
|------------------|----------------------|----------|-----|
| AUtempR          | 2020-12-28(월) 오후 ... | 파일 폴더    |     |
| NEO2             | 2020-12-29(화) 오후 ... | 파일 폴더    |     |
| test             | 2021-02-07(일) 오후 ... | 파일 폴더    |     |
| Application.java | 2013-10-26(토) 오후 ... | JAVA 파일  | 1KB |
| Hello.class      | 2021-01-31(일) 오전 ... | CLASS 파일 | 1KB |
| Hello.           |                      |          |     |
| RunSt            |                      |          |     |
| RunSt            |                      |          |     |

```
package sec12.exam01_package_compile;

public class Application {
    public static void main(String[] args) {
        System.out.println("애플리케이션을 실행합니다.");
    }
}
```

Application.java 파일의 소소코드 내용(위)



C:\>cd temp

C:\>Temp>dir

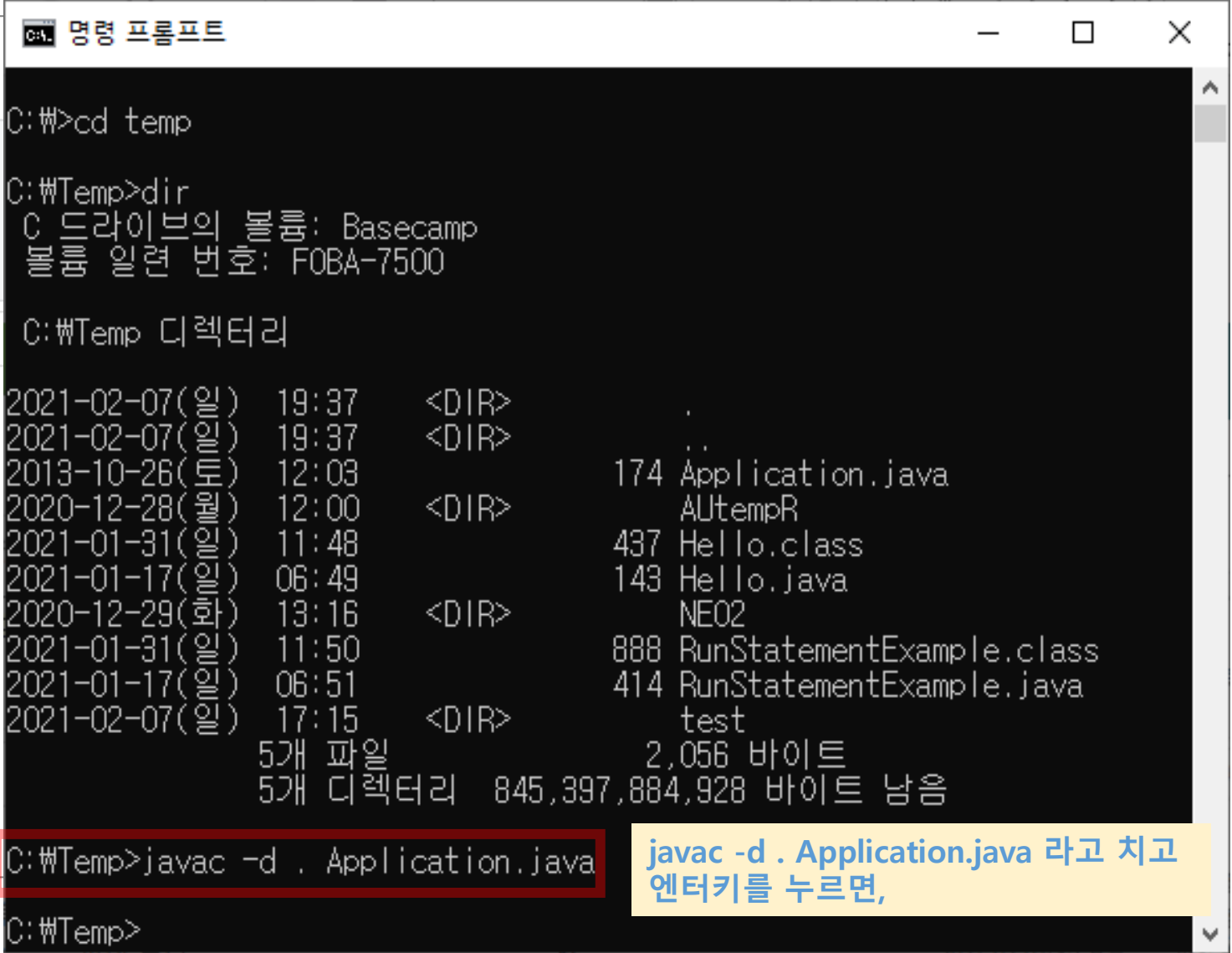
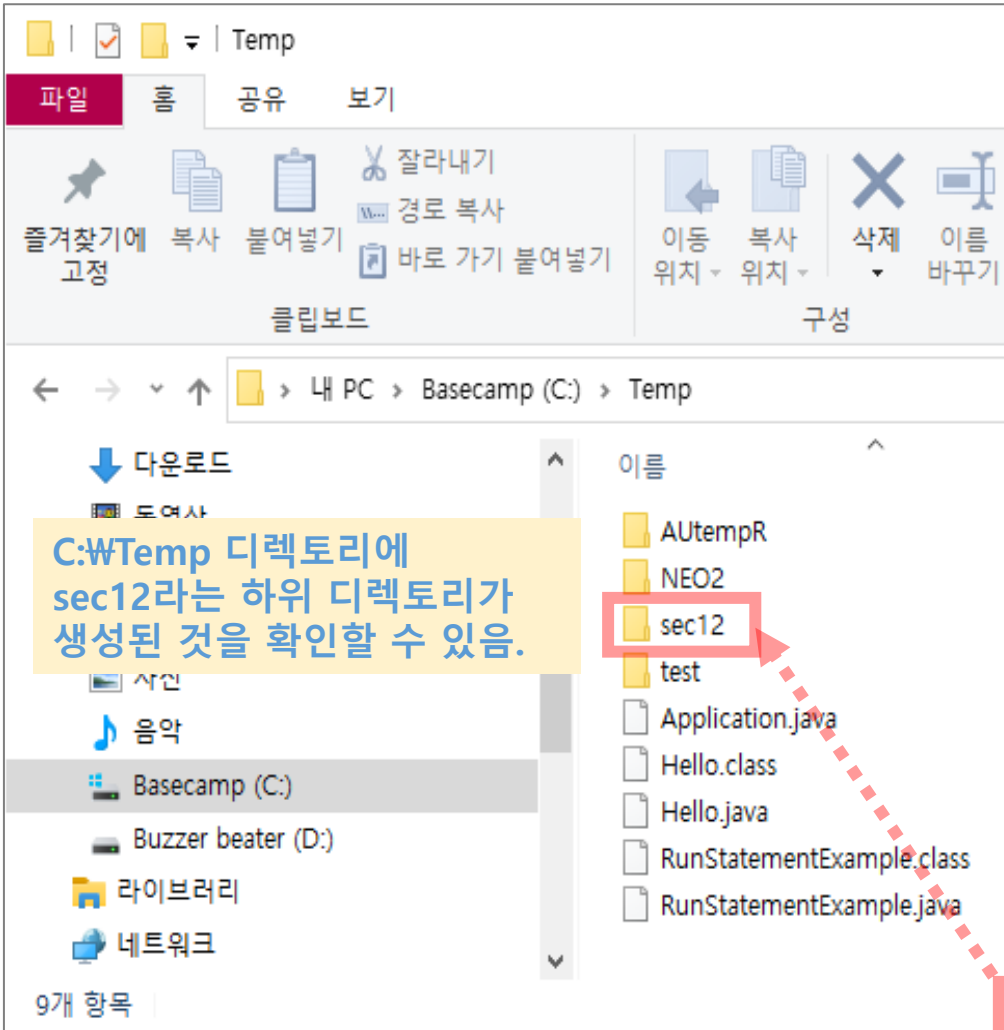
C:\Temp>javac -d . Application.java

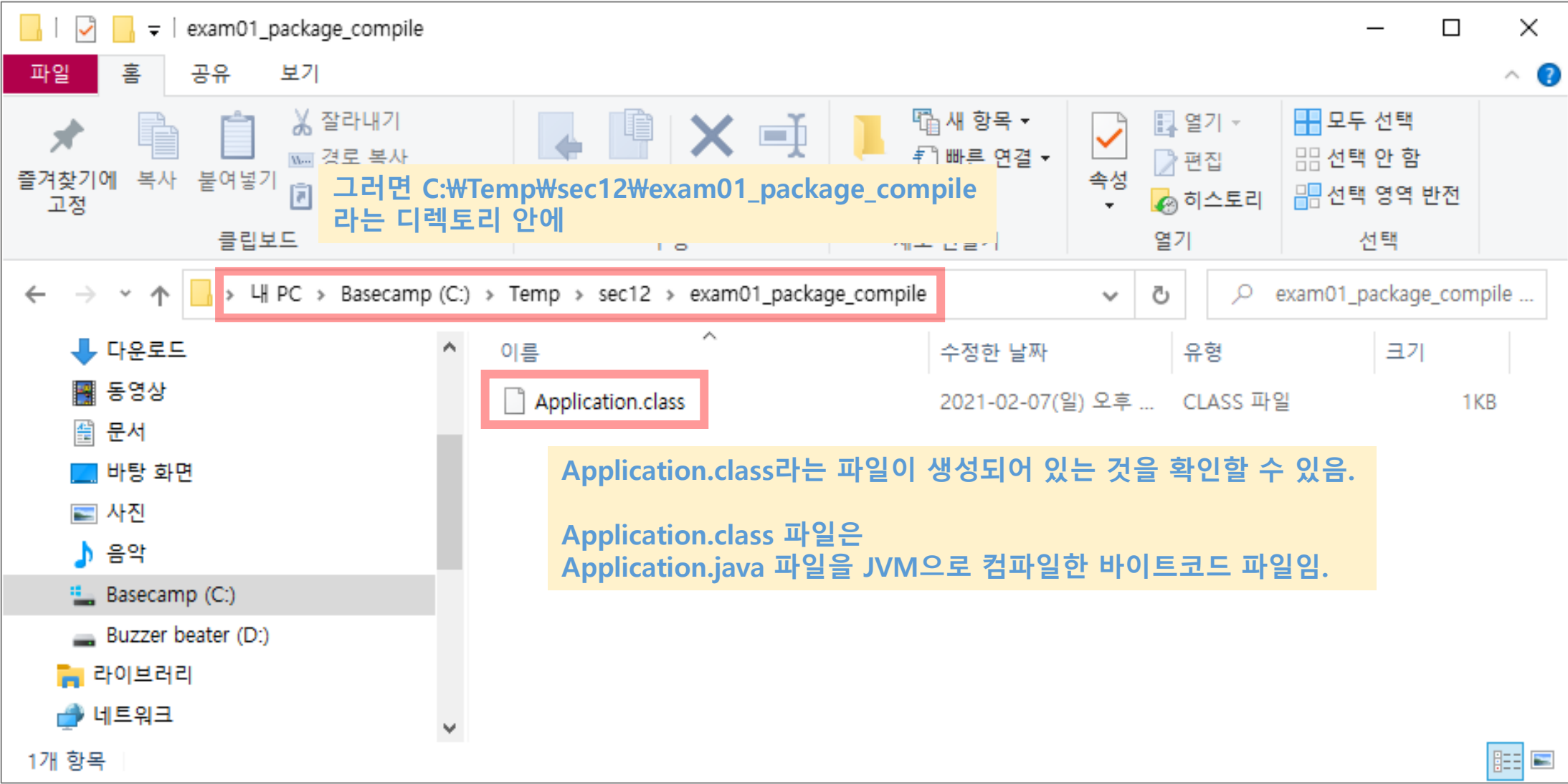
C:\Temp>

명령 프롬프트(DOS창)을 켜서  
C:\Temp 디렉토리로 들어가서

2021-02-07(일) 19:37 <DIR>  
2021-02-07(일) 19:37 <DIR>  
2013-10-26(토) 12:03 <DIR>  
2020-12-28(월) 12:00 <DIR>  
2021-01-31(일) 11:48 174 Application.java  
2021-01-17(일) 06:49 437 Hello.class  
2020-12-29(화) 13:16 143 Hello.java  
2021-01-31(일) 11:50 888 RunStatementExample.class  
2021-01-17(일) 06:51 414 RunStatementExample.java  
2021-02-07(일) 17:15 <DIR>  
5개 파일 2,056 바이트  
5개 디렉터리 845,397,884,928 바이트 남음

javac -d . Application.java 라고 치고  
엔터키를 누름





패키지에 소속된 클래스를 명령 프롬프트로 실행하려면,  
바이트코드 파일(.class)이 있는 폴더에서 java 명령어를 실행하면 안된다.

패키지는 클래스의 일부분이므로,  
패키지가 시작하는 폴더에서 java 명령어를 실행해야 한다.

Application 클래스의 패키지 sec12는 C:\Temp 폴더에 있기 때문에,  
C:\Temp 폴더에서, 빨간색 사각형 표시 부분과 같이 java 명령어를 실행한다.

java 명령어를 실행한 결과는  
초록색 사각형 표시 부분과 같이 나타난다.

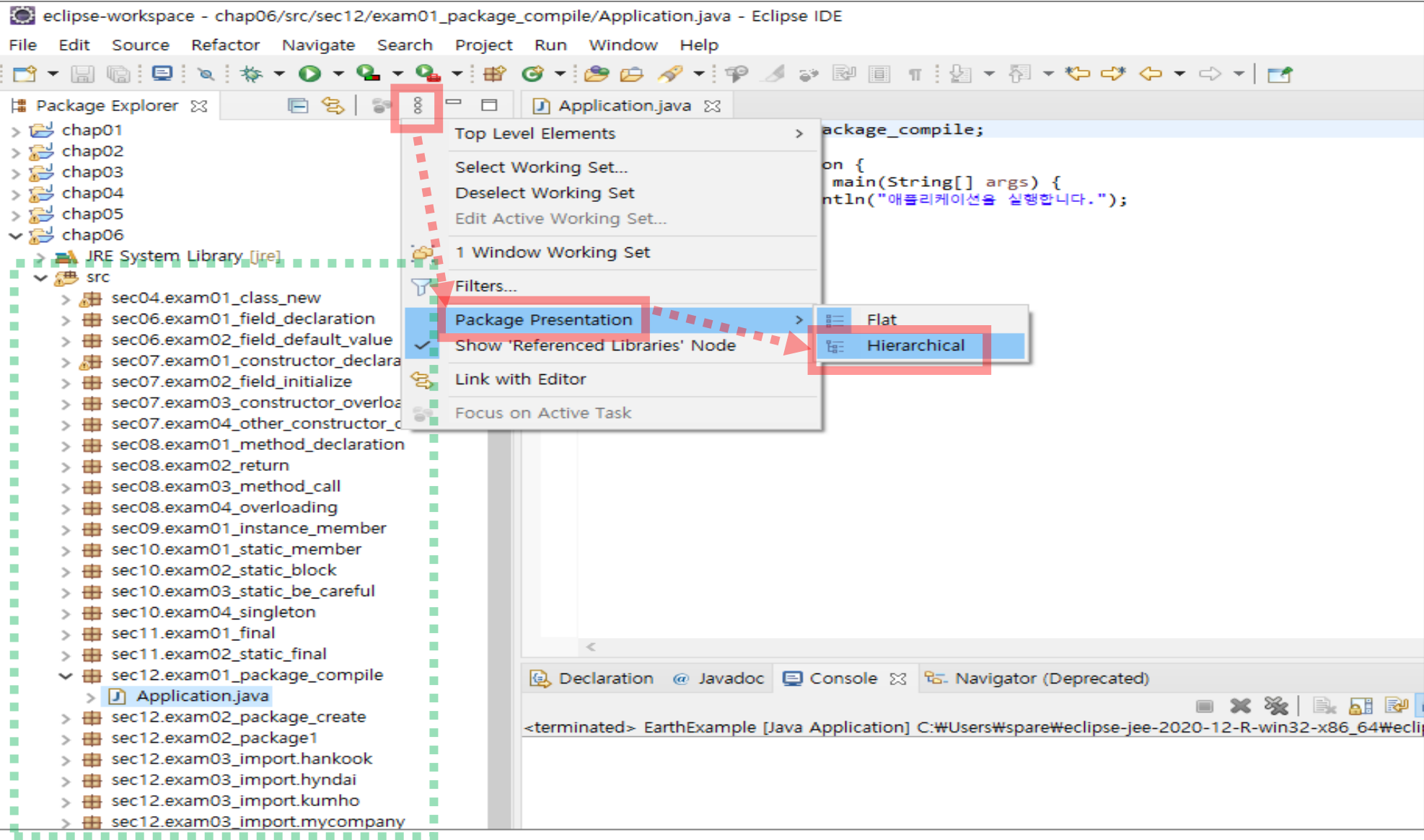
# 객체지향 프로그래밍



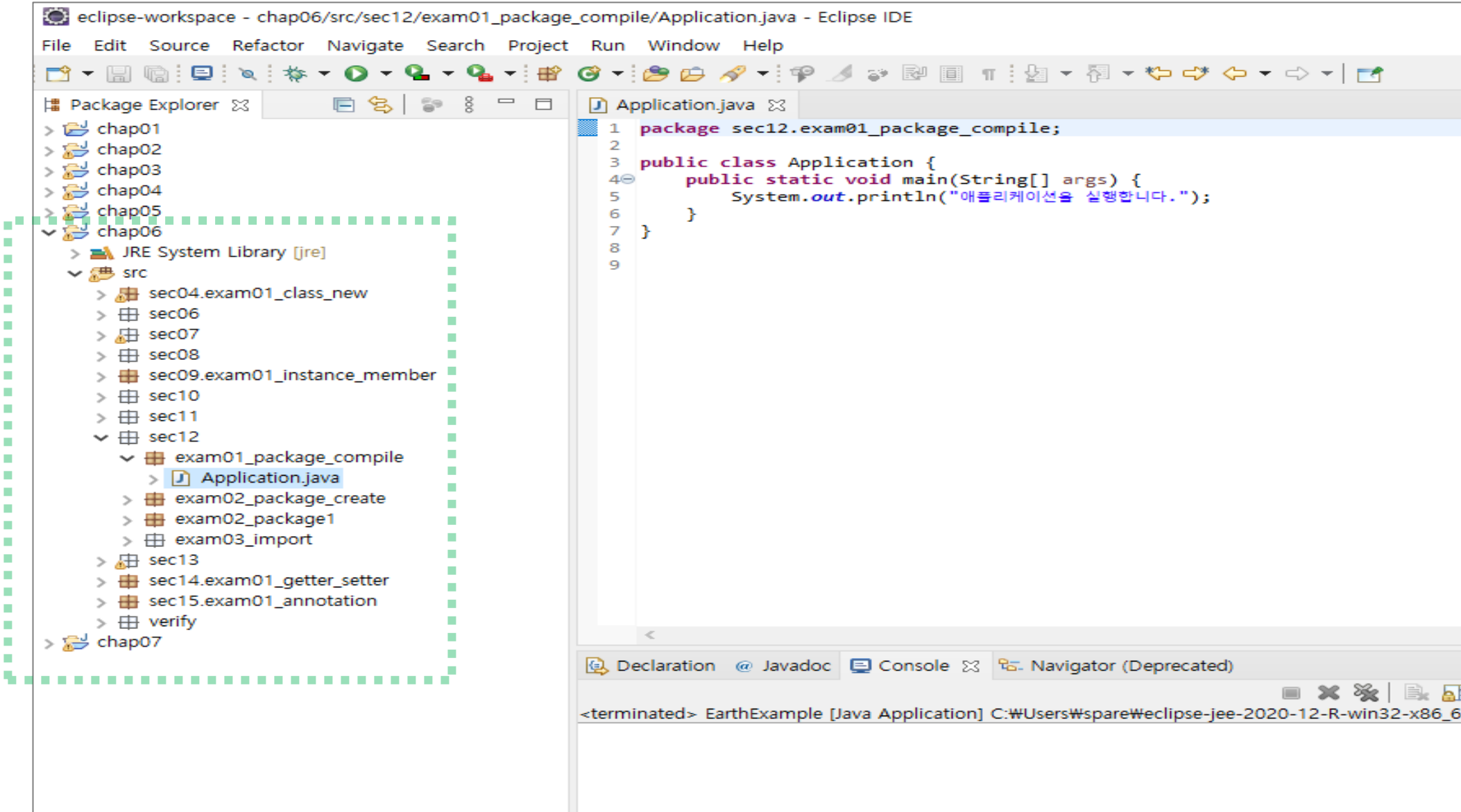
이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재6장 보충교재  
클래스(class)

상하위 패키지를 계층적으로  
보기 원할때의 설정법







# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재6장 보충교재  
클래스(class)

this 키워드 &  
자바(JAVA) 언어에서의 관례

```
public class Korean {  
    //필드  
    String nation = "대한민국";  
    String name;  
    String ssn;  
    //생성자  
    /* public Korean(String n, String s) {  
        name = n;  
        ssn = s;  
    } */  
  
    public Korean(String name, String ssn) {  
        this.name = name;  
        this.ssn = ssn;  
    }  
}
```

필드명

매개변수명

필드명

매개변수명

자바 언어에서는, 어떤 클래스의 생성자의 매개변수의 이름을 지을 때, 관례적으로 이 클래스의 필드의 이름과 동일한 이름을 매개변수명으로 사용하는 경우가 많다.

이런 경우, 클래스의 생성자의 매개변수명과 클래스의 필드명이 동일하기 때문에 어느 것이 무엇인지 헷갈릴 수가 있다. 따라서 혼동되지 않도록 이 둘을 구별하기 위해서, 자바는 **this** 라는 키워드를 사용한다. **this**라는 키워드가 의미하는 것은 **클래스의 생성자의 매개변수와 클래스의 필드가 속한 바로 그 클래스 자기 자신을 의미**한다.  
(여기서의 this는 (자기 자신의 클래스인) Korean 클래스를 지칭한다)

```
public class KoreanExample {  
    public static void main(String[] args) {  
        Korean k1 = new Korean("박자바", "011225-1234567");  
        System.out.println("k1.name : " + k1.name);  
        System.out.println("k1.ssn : " + k1.ssn);  
  
        Korean k2 = new Korean("김자바", "930525-0654321");  
        System.out.println("k2.name : " + k2.name);  
        System.out.println("k2.ssn : " + k2.ssn);  
    }  
}
```

<< 실행 결과 >>

```
k1.name : 박자바  
k1.ssn : 011225-1234567  
k2.name : 김자바  
k2.ssn : 930525-0654321
```

# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재6장 보충교재  
클래스(class)

메소드의 가변 매개변수  
처리 방법

메소드의 매개변수가 가변적인 경우에,

(1) 메소드의 매개변수를 배열로 받아서 처리하는 방법, 또는

```
public class Computer {
    int sum1(int[] values) {
        int sum = 0;
        for(int i=0; i<values.length; i++) {
            sum += values[i];
        }
        return sum;
    }

    int sum2(int ... values) {
        int sum = 0;
        for(int i=0; i<values.length; i++) {
            sum += values[i];
        }
        return sum;
    }
}
```

(2) “데이터타입 ... 매개변수명”의 형태로  
메소드의 매개변수를 설정하여  
처리하는 방법중에서

하나를 선택하여 처리한다.

```
public class ComputerExample {
    public static void main(String[] args) {
        Computer myCom = new Computer();

        int[] values1 = {1, 2, 3};
        int result1 = myCom.sum1(values1);
        System.out.println("result1: " + result1);

        int result2 = myCom.sum1(new int[] {1, 2, 3, 4, 5});
        System.out.println("result2: " + result2);

        int result3 = myCom.sum2(1, 2, 3);
        System.out.println("result3: " + result3);

        int result4 = myCom.sum2(1, 2, 3, 4, 5);
        System.out.println("result4: " + result4);
    }
}
```

<< 실행 결과 >>

```
result1: 6
result2: 15
result3: 6
result4: 15
```

```

1 package ch06.sec08.exam02;
2
3 public class ComputerExample {
4     public static void main(String[] args) {
5         //Computer 객체 생성
6         Computer myCom = new Computer();
7
8         //sum() 메소드 호출 시 매개값 1, 2, 3을 제공하고
9         //합산 결과를 리턴 받아 result1 변수에 대입
10        int result1 = myCom.sum(1, 2, 3);
11        System.out.println("result1: " + result1);
12
13        //sum() 메소드 호출 시 매개값 1, 2, 3, 4, 5를 제공하고
14        //합산 결과를 리턴 받아 result2 변수에 대입
15        int result2 = myCom.sum(1, 2, 3, 4, 5);
16        System.out.println("result2: " + result2);
17
18        //sum() 메소드 호출 시 배열을 제공하고
19        //합산 결과를 리턴 받아 result3 변수에 대입
20        int[] values = { 1, 2, 3, 4, 5 };
21        int result3 = myCom.sum(values);
22        System.out.println("result3: " + result3);
23
24        //sum() 메소드 호출 시 배열을 제공하고
25        //합산 결과를 리턴 받아 result4 변수에 대입
26        int result4 = myCom.sum(new int[] { 1, 2, 3, 4, 5 });
27        System.out.println("result4: " + result4);
28    }
29 }
    
```

```

1 package ch06.sec08.exam02;
2
3 public class Computer {
4     //가변길이 매개변수를 갖는 메소드 선언
5     int sum(int ... values) {
6         //sum 변수 선언
7         int sum = 0;
8
9         //values는 배열 타입의 변수처럼 사용
10        for (int i = 0; i < values.length; i++) {
11            sum += values[i];
12        }
13
14        //합산 결과를 리턴
15        return sum;
16    }
17 }
    
```

<< 실행 결과 >>

```

result1: 6
result2: 15
result3: 15
result4: 15
    
```

# 객체지향 프로그래밍



이미지 출처 : <https://www.brickowl.com/catalog/lego-computer-programmer-minifigure>

교재6장 보충교재  
클래스(class)

접근변경자  
접근제한자(access modifier)



## 클래스의 접근 제한

package1

클래스 A

```
package
sec13.exam01_class
_access.package1;
```

```
class A {
}
```

default 접근제한을  
갖는 클래스 A  
(default 지정자가  
생략된 것임)  
  
default class A { }

클래스 B

```
package
sec13.exam01_class_
access.package1;
```

```
public class B {
    A a;
}
```

클래스 A와 B는 같은 패키지  
안에 있기 때문에, 클래스 A가  
default 제한접근을 하고 있어도  
클래스 B에서 클래스 A로  
접근 가능함.

클래스 A와 C는 서로 다른 패키지에 속해  
있고, 클래스 A의 제한접근자가 default이기  
때문에, 클래스 C가 클래스 A로 접근할 수  
없음(불가능함)

package2

클래스 C

```
package
sec13.exam01_class_access
.package2;
```

```
import
sec13.exam01_class_access
.package1.*;
```

```
public class C {
    // A a;
    B b;
}
```

클래스 B와 C는 서로 다른 패키지에 속해  
있지만, 클래스 B의 제한접근자가 public  
이기 때문에, 클래스 C가 클래스 B로 접근  
가능함.

\* default 는 접근제한자가 아니라, 접근제한자가 붙지 않은 상태를 의미함.

## 생성자의 접근 제한 – 기본 생성자(default constructor)의 경우

package1

클래스 A

```
package sec13.exam01_
class_access.package1;

public class A {

    //생성자
    public A() {}

}
```

이전 교재  
p203~204  
6.7, 6.7.1 참고

클래스에서 생성자를  
선언하지 않은 경우,

클래스 A

```
package sec13.exam01_
class_access.package1;

public class A {

    //생성자
    public A() {}

}
```

컴파일러에 의해 자동적으로  
기본 생성자가 추가되는데,  
이때, 기본 생성자의 접근권한은  
클래스의 접근권한과 동일하다.

여기서는 클래스의 접근제한자가  
public이므로 기본 생성자의  
접근권한도 동일하게 public 이다.

package2

클래스 B

```
package
sec13.exam01_class_access
.package2;

class B {

    //생성자
    B() {}

}
```

이전 교재  
p203~204  
6.7, 6.7.1 참고

클래스에서 생성자를  
선언하지 않은 경우,

클래스 B

```
package
sec13.exam01_class_access
.package2;

class B {

    //생성자
    B() {}

}
```

default 접근제한을 갖는  
클래스 B  
(default 지정자가  
생략된 것임)  
default class B { }

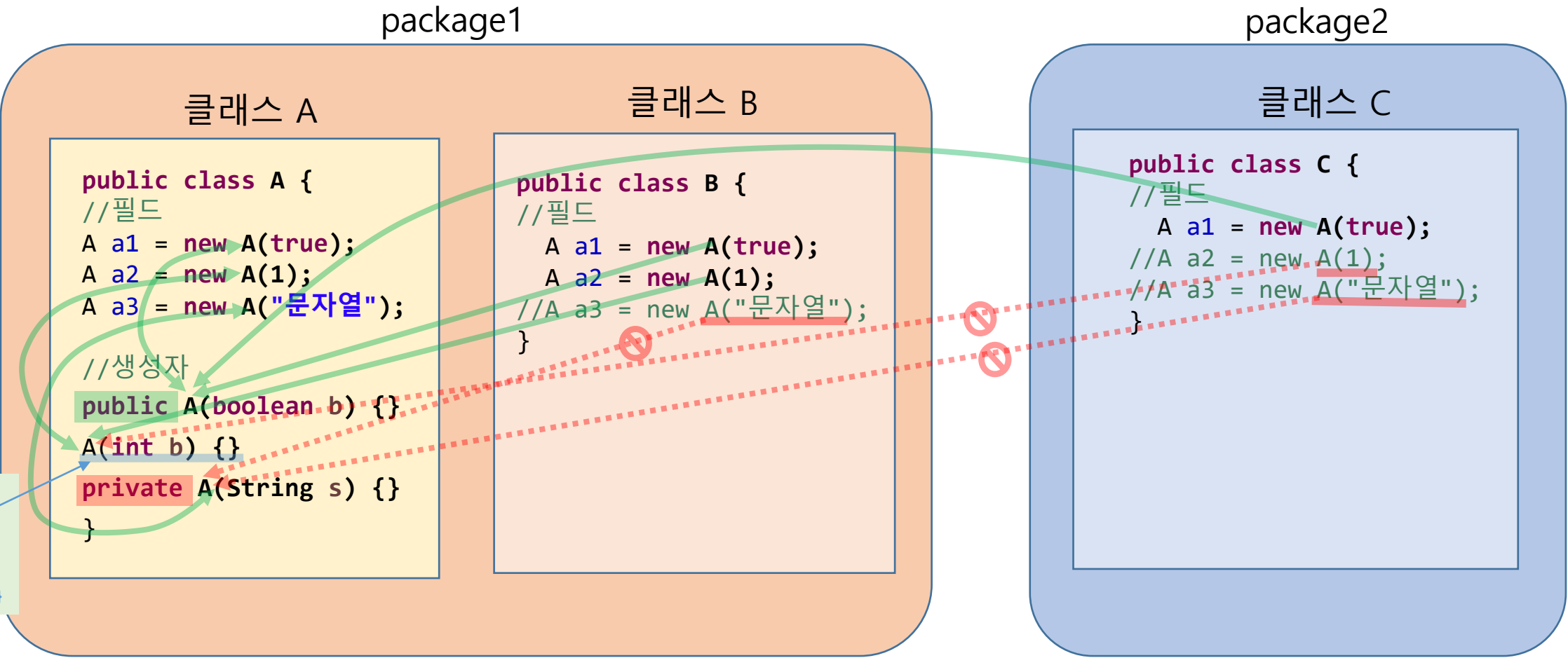
default 접근제한을 갖는  
생성자 B()  
(default 지정자가  
생략된 것임)  
default B ( ) { }

컴파일러에 의해 자동적으로  
기본 생성자가 추가되는데,  
이때, 기본 생성자의 접근권한은  
클래스의 접근권한과 동일하다.

여기서는 클래스의 접근제한자가  
default 이므로 기본 생성자의  
접근권한도 동일하게 default 이다.

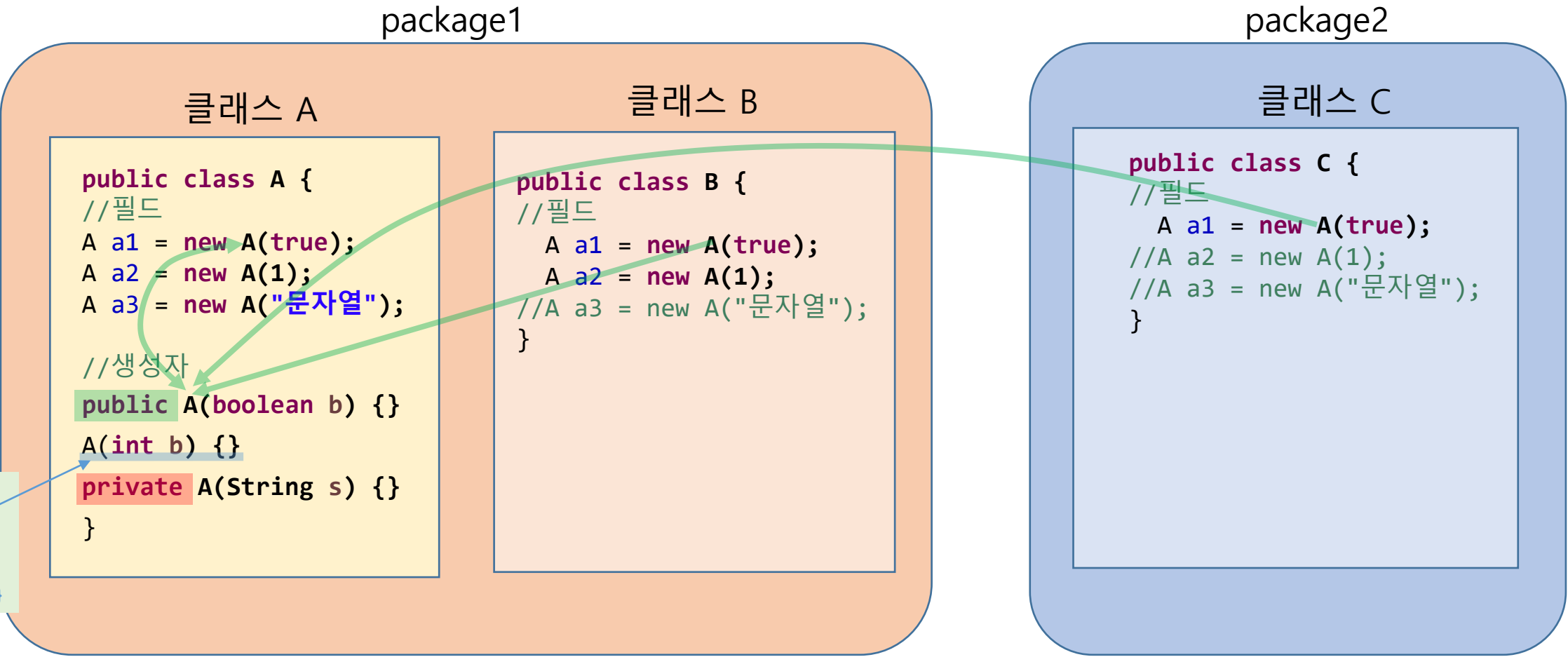
\* default 는 접근제한자가 아니라, 접근제한자가 붙지 않은 상태를 의미함.

생성자의 접근 제한 – 생성자를 명시적으로 정의한 경우



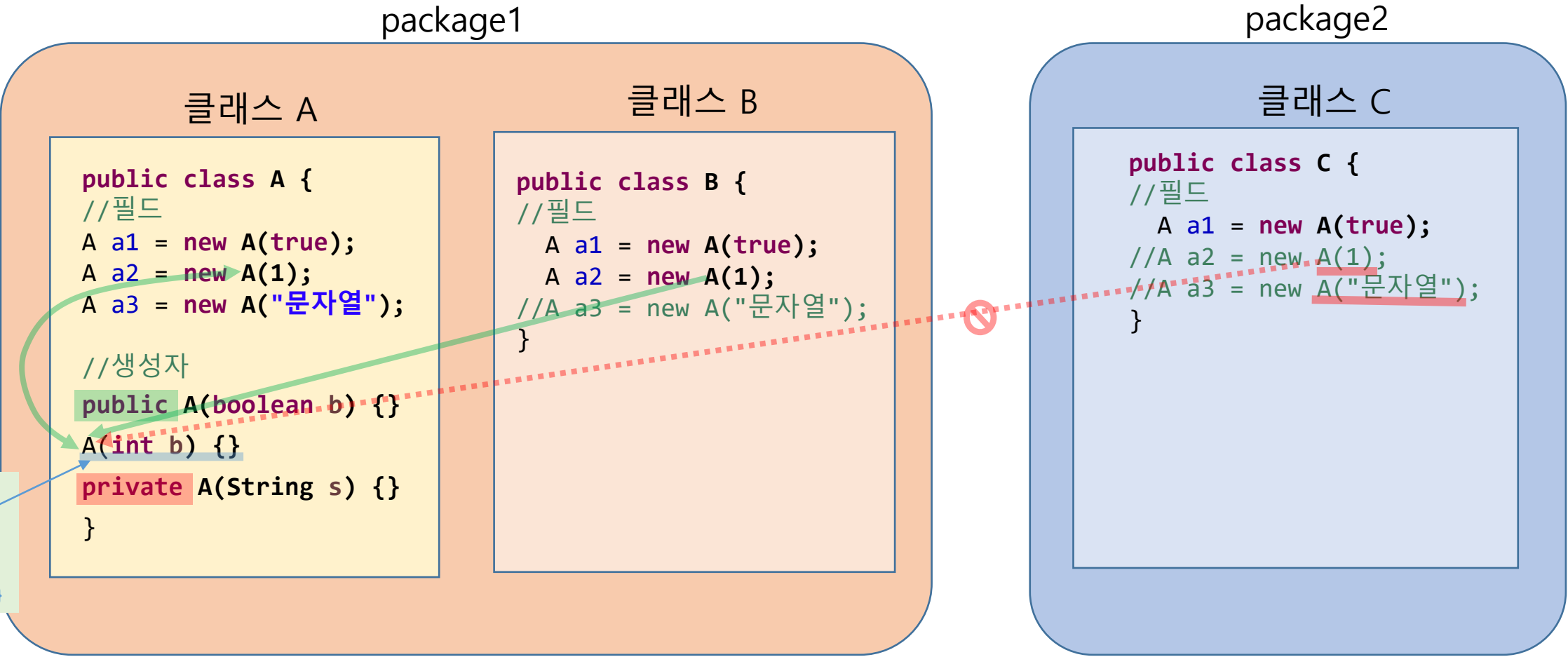
\* default 는 접근제한자가 아니라, 접근제한자가 붙지 않은 상태를 의미함.

## 생성자의 접근 제한 – 생성자를 명시적으로 정의한 경우



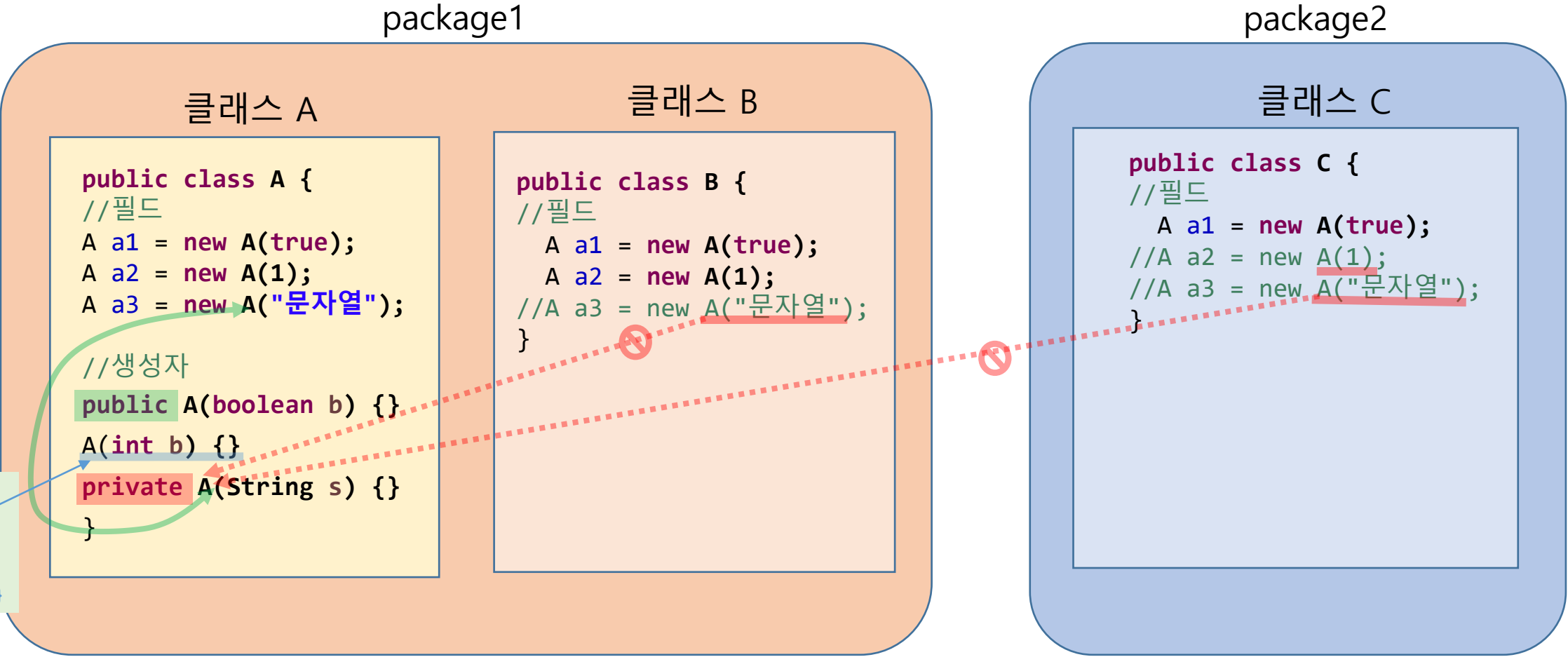
\* default 는 접근제한자가 아니라, 접근제한자가 붙지 않은 상태를 의미함.

## 생성자의 접근 제한 – 생성자를 명시적으로 정의한 경우



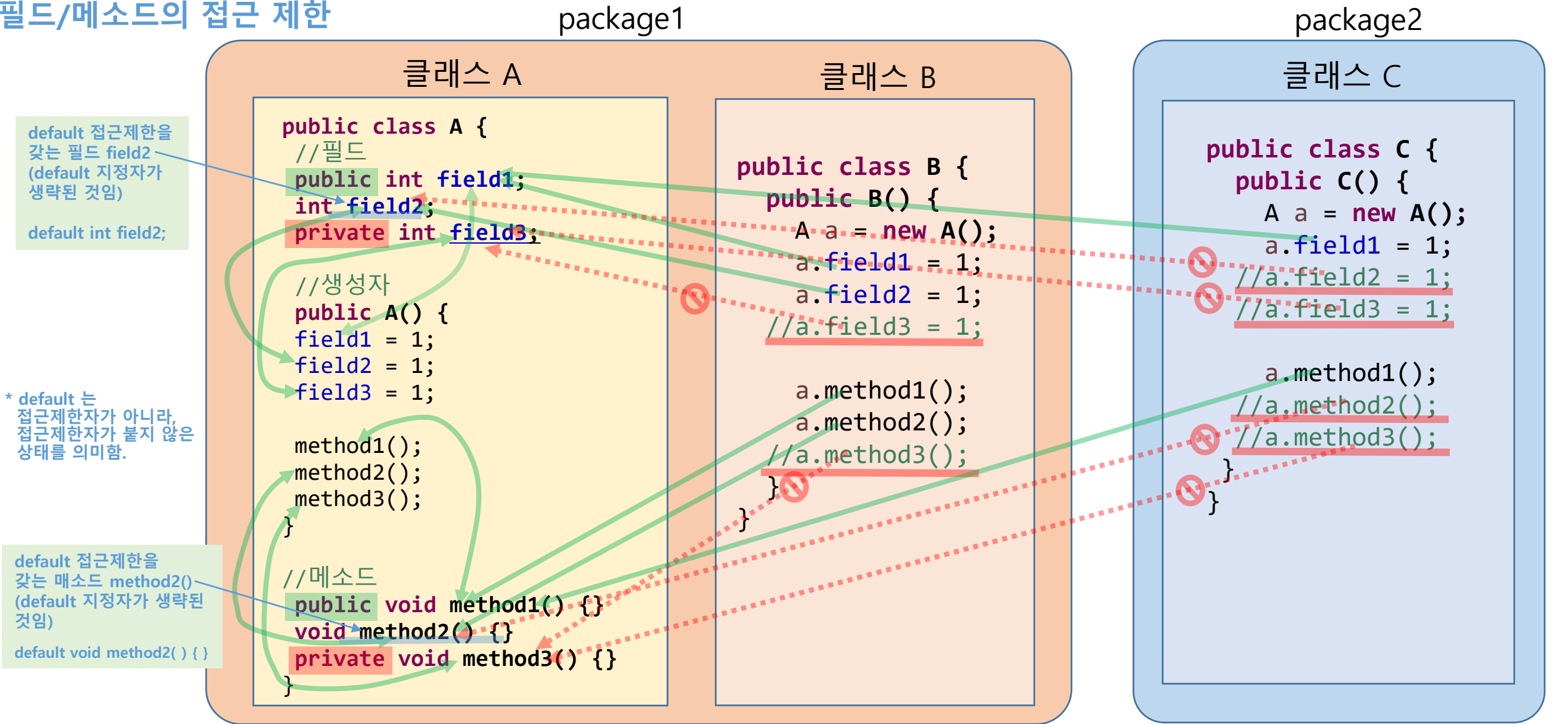
\* default 는 접근제한자가 아니라, 접근제한자가 붙지 않은 상태를 의미함.

## 생성자의 접근 제한 – 생성자를 명시적으로 정의한 경우



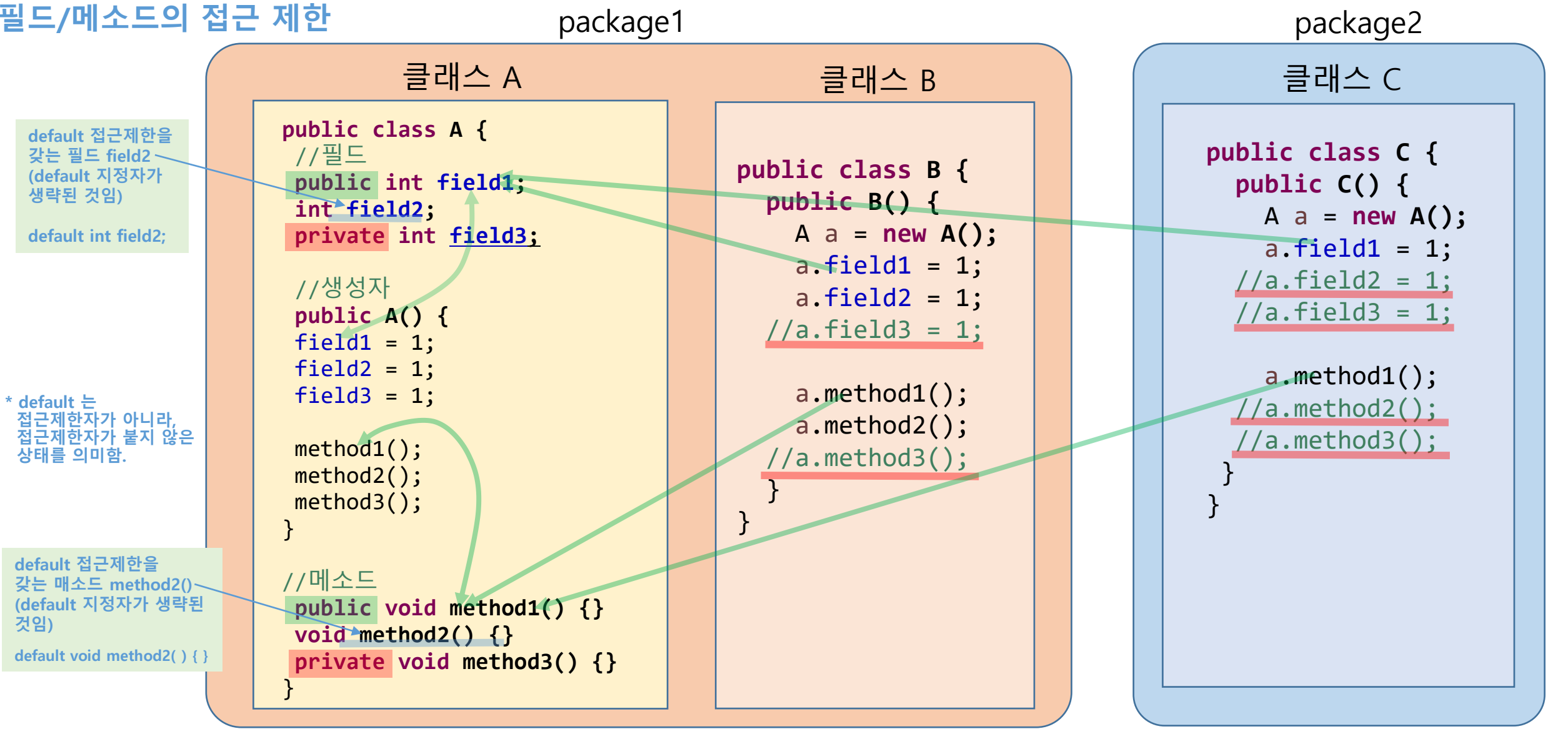
\* default 는 접근제한자가 아니라, 접근제한자가 붙지 않은 상태를 의미함.

필드/메소드의 접근 제한

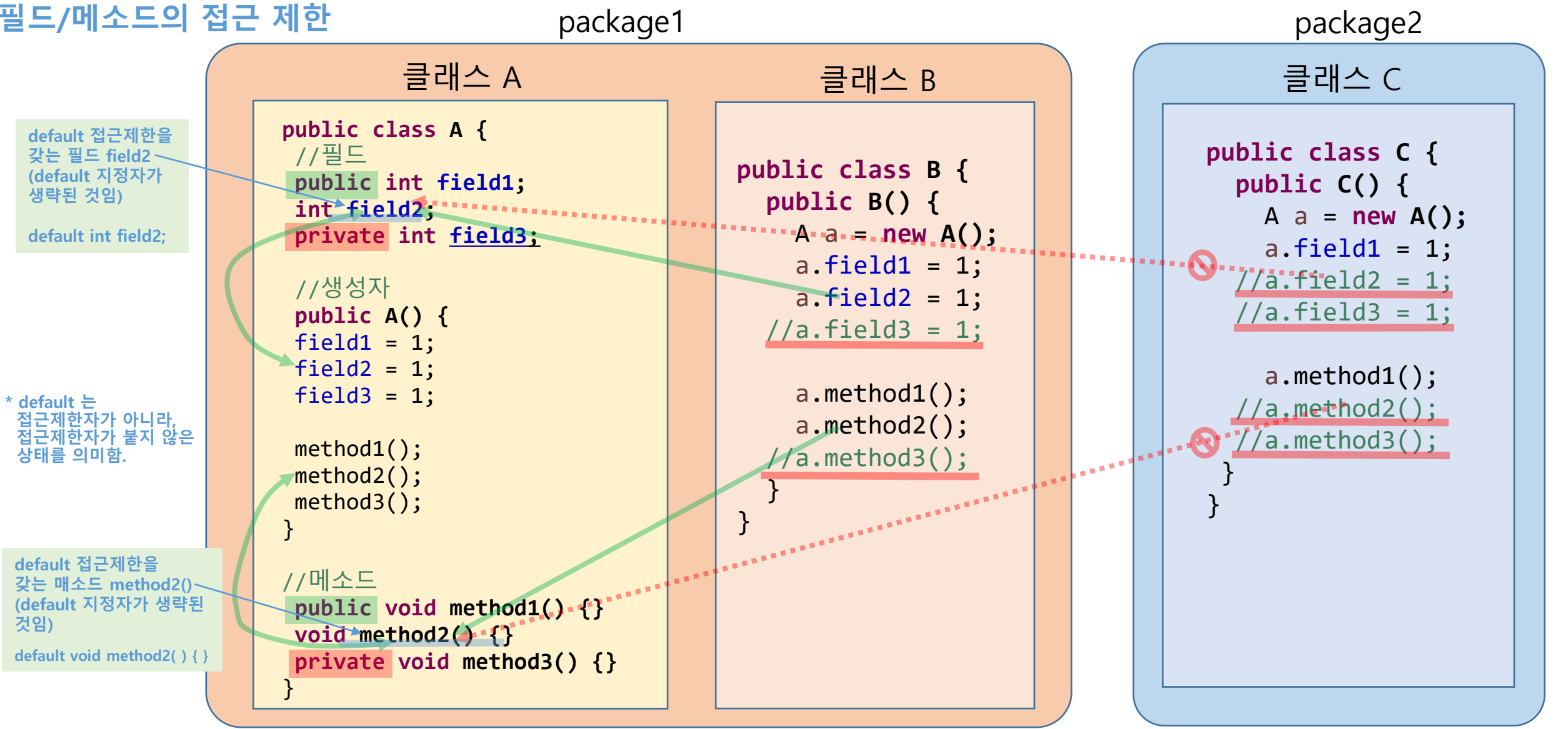




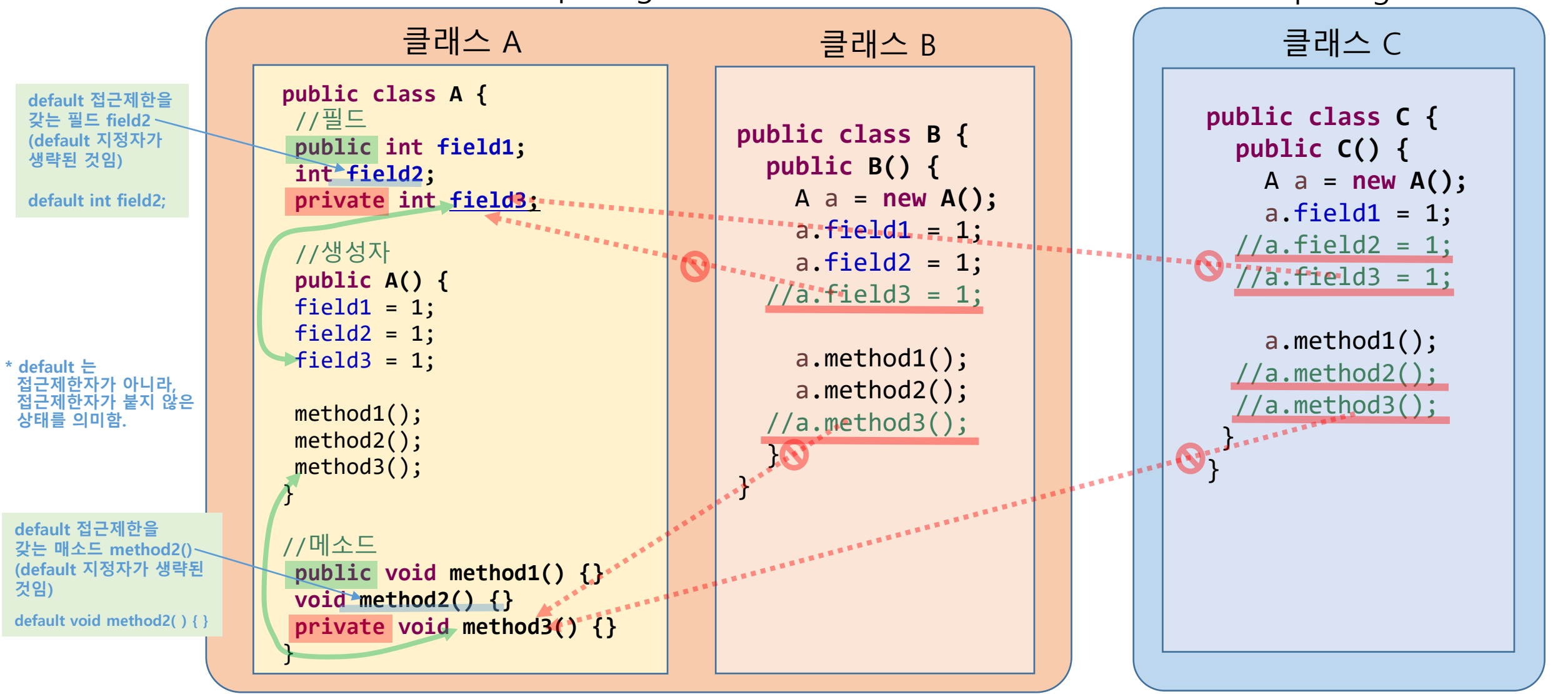
필드/메소드의 접근 제한



## 필드/메소드의 접근 제한



## 필드/메소드의 접근 제한





[교재6장 보충교재]

클래스(class)

- 끝 -