객체 배열

컴퓨터공학전공
박요한

목차

- 객체 배열
- 메소드 오버로딩
- 접근 지정자
- static 멤버

객체 배열 만들기

- 배열 선언과 배열 생성의 두 단계
  - ✓배열 선언

```
int     intArray[];
char    charArray[];
```

또는

```
int[]   intArray;
char[]  charArray;
```

  - ✓배열 생성

```
intArray = new int[10];
charArray = new char[20];
```

또는

```
int intArray[] = new int[10];
char charArray[] = new char[20];
```

- 객체 배열 선언과 배열 생성의 두 단계
  - ✓객체 배열 선언

```
클래스이름[ ]  레퍼런스변수(c);
```

  - ✓객체 배열 생성

```
레퍼런스변수(c) = new 클래스이름[크기];
```

객체 배열

- 객체 배열 생성 및 사용

```
Circle [] c;
c = new Circle[5];

for(int i=0; i<c.length; i++) // c.length는 배열 c의 크기로서 5
    c[i] = new Circle(i);
```

Circle 배열에 대한 레퍼런스 변수 c 선언

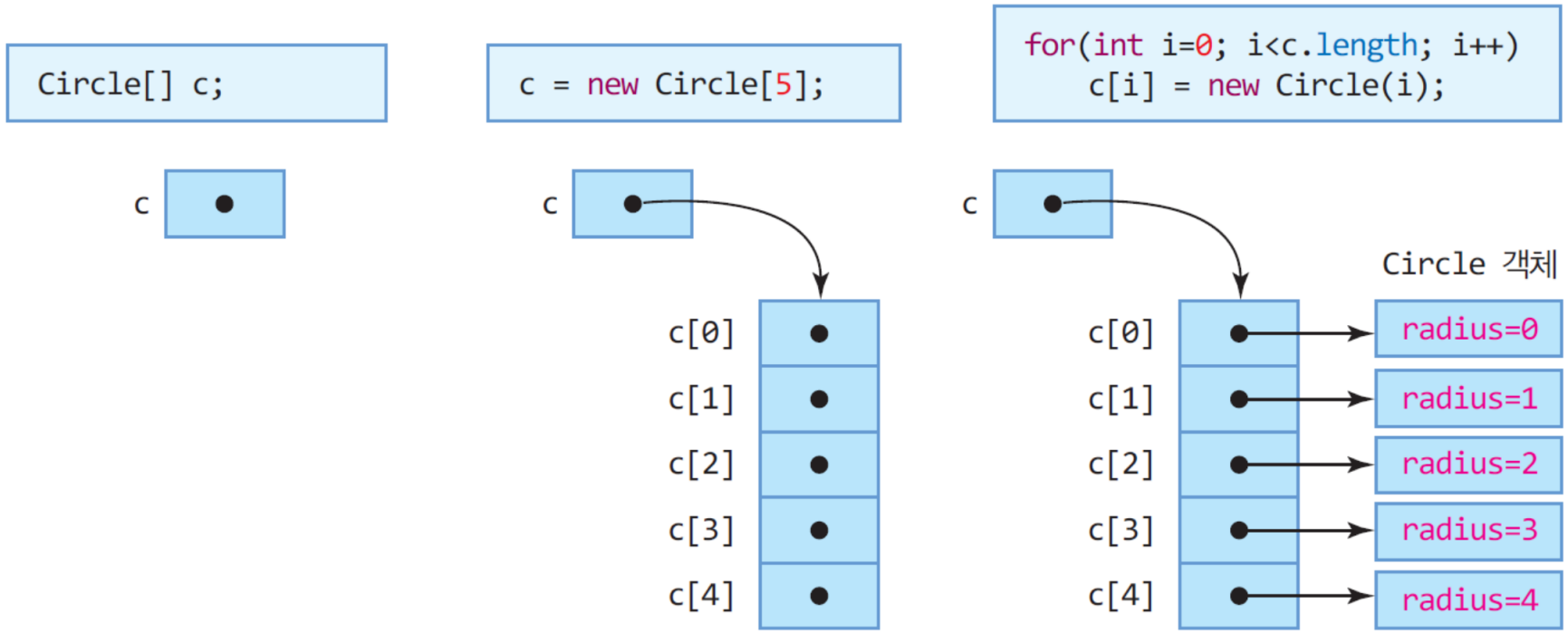
레퍼런스 배열 생성

배열의 각 원소 객체 생성

```
for(int i=0; i<c.length; i++) // 배열에 있는 모든 Circle 객체의 면적 출력
    System.out.print((int)(c[i].getArea()) + " ");
```

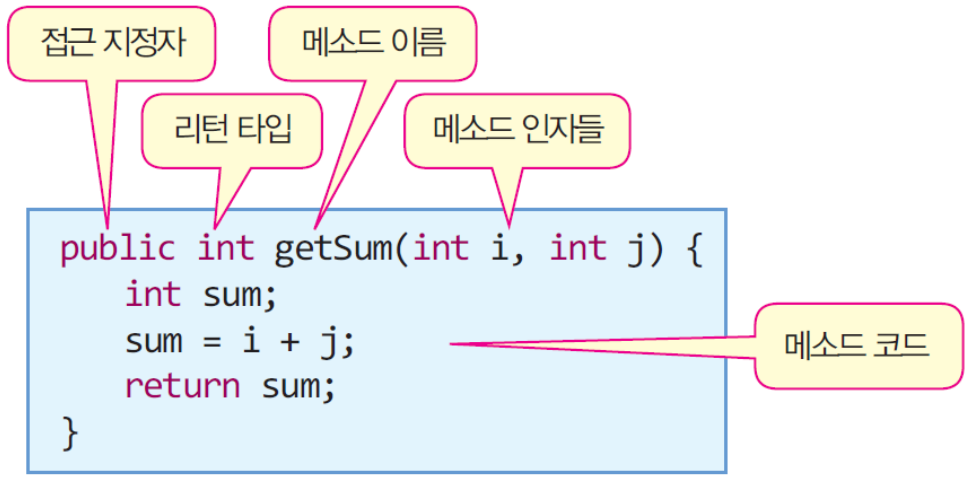
배열의 원소 객체 사용

객체 배열 선언과 생성 과정



메소드 형식

- 메소드
  - ✓클래스의 멤버 함수, C/C++의 함수와 동일
  - ✓자바의 모든 메소드는 반드시 클래스 안에 있어야 함(캡슐화 원칙)
- 메소드 구성 형식
  - ✓접근 지정자
    - public. private, protected, 디폴트(접근 지정자 생략된 경우)
  - ✓리턴 타입
    - 메소드가 반환하는 값의 데이터 타입



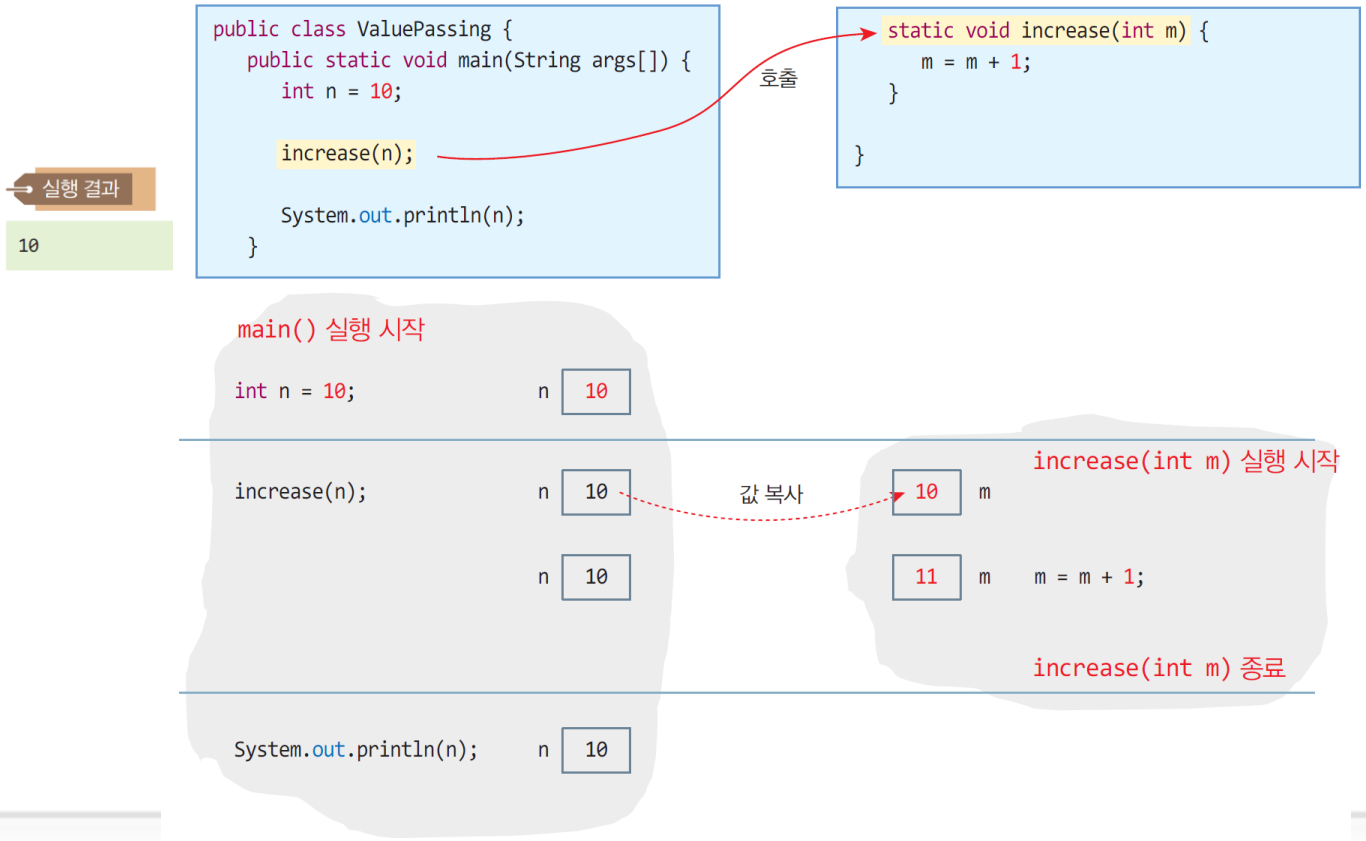
인자 전달

- 자바의 인자 전달 방식
  - ✓경우 1. 기본 타입의 값 전달
    - 값이 복사되어 전달
    - 메소드의 매개변수가 변경되어도 호출한 실인자 값은 변경되지 않음
  - ✓경우 2. 객체 혹은 (경우 3.) 배열 전달
    - 객체나 배열의 레퍼런스만 전달
      - 객체 혹은 배열이 통째로 복사되어 전달되는 것이 아님
    - 메소드의 매개변수와 호출한 실인자 객체나 배열 공유

인자 전달 – 기본 타입의 값이 전달되는 경우

✓ 매개변수가 byte, int, double 등 기본 타입의 값일 때

- 호출자가 건네는 값이 매개변수에 복사되어 전달. 실인자 값은 변경되지 않음



인자 전달 – 객체가 전달되는 경우

✓ 객체의 레퍼런스만 전달

- 매개 변수가 실인자 객체 공유

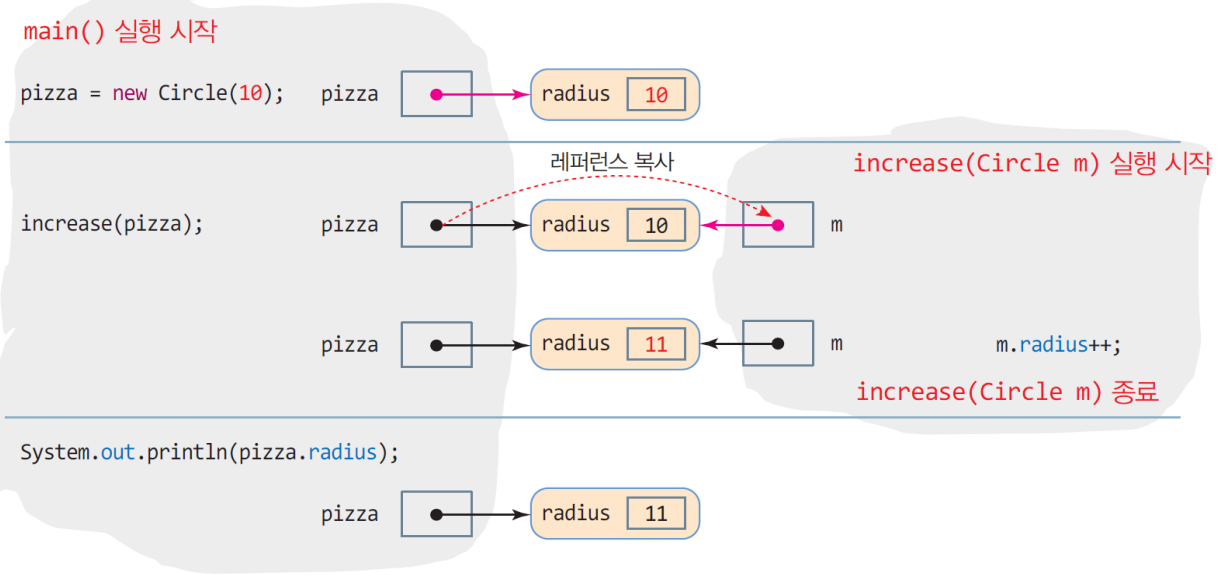
```
public class ReferencePassing {
   public static void main (String args[]) {
      Circle pizza = new Circle(10);

      increase(pizza);

      System.out.println(pizza.radius);
   }
```

호출

```
static void increase(Circle m) {
      m.radius++;
   }

}
```

실행 결과

```
11
```



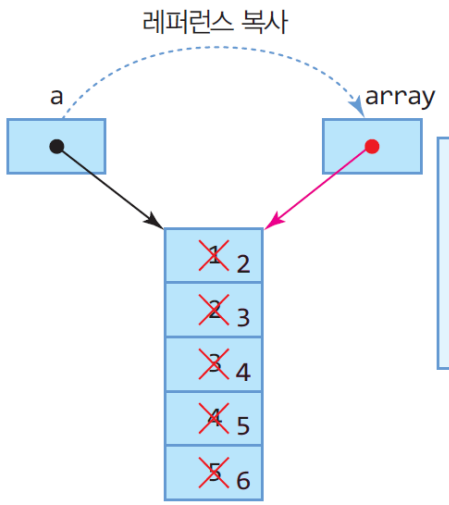
인자 전달 – 배열이 전달되는 경우

✓ 배열 레퍼런스만 매개 변수에 전달
- 배열 통째로 전달되지 않음
- 객체가 전달되는 경우와 동일
- 매개변수가 실인자의 배열을 공유

```
public class ArrayPassing {

    public static void main(String args[]) {
        int a[] = {1, 2, 3, 4, 5};


        increase(a);


        for(int i=0; i<a.length; i++)
            System.out.print(a[i]+" ");
    }
```

실행 결과

```
2 3 4 5 6
```




```
    static void increase(int[] array) {
        for(int i=0; i<array.length; i++) {
            array[i]++;
        }
    }
}
```

메소드 오버로딩

- 메소드 오버로딩(Overloading)
  - ✓이름이 같은 메소드 작성
    - 매개변수의 개수나 타입이 서로 다르고
    - 이름이 동일한 메소드들
  - ✓리턴 타입은 오버로딩과 관련 없음

```
// 메소드 오버로딩이 성공한 사례

class MethodOverloading {
   public int getSum(int i, int j) {
      return i + j;
   }
   public int getSum(int i, int j, int k) {
      return i + j + k;
   }
}
```

```
// 메소드 오버로딩이 실패한 사례

class MethodOverloadingFail {
   public int getSum(int i, int j) {
      return i + j;
   }
   public double getSum(int i, int j) {
      return (double)(i + j);
   }
}
```

두 개의 getSum() 메소드는 매개변수의 개수, 타입이 모두 같기 때문에 메소드 오버로딩 실패

오버로딩된 메소드 호출

```
public static void main(String args[]) {
    MethodSample a = new MethodSample();

    int i = a.getSum(1, 2);

    int j = a.getSum(1, 2, 3);

    double k = a.getSum(1.1, 2.2);

}
```

매개 변수의 개수와 타입이 서로 다른 3 함수 호출

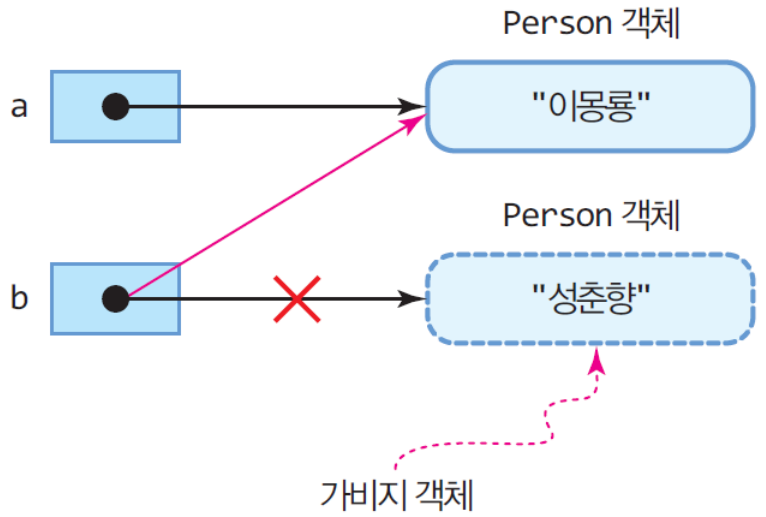
```
public class MethodSample {
    public int getSum(int i, int j) {
        return i + j;
    }

    public int getSum(int i, int j, int k) {
        return i + j + k;
    }

    public double getSum(double i, double j) {
        return i + j;
    }
}
```

객체의 소멸과 가비지 컬렉션

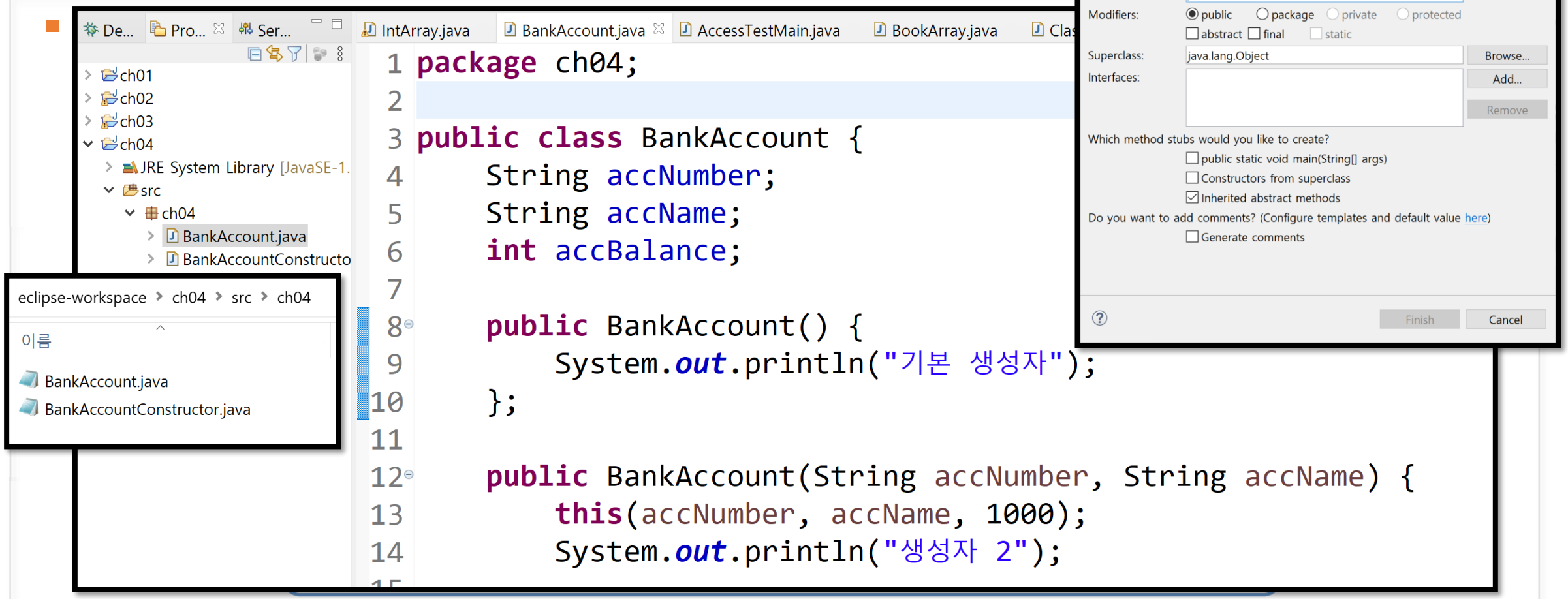
- 객체 소멸
  - ✓new에 의해 할당된 객체 메모리를 자바 가상 기계의 가용 메모리로 되돌려 주는 행위
- 자바 응용프로그램에서 임의로 객체 소멸할 수 없음
  - ✓객체 소멸은 자바 가상 기계의 고유한 역할
  - ✓자바 개발자에게는 매우 다행스러운 기능
    - C/C++에서는 할당받은 객체를 개발자가 되돌려 주어야 함
      - C/C++ 프로그램 작성을 어렵게 만드는 요인
- 가비지
  - ✓가리키는 레퍼런스가 하나도 없는 객체
    - 누구도 사용할 수 없게 된 메모리
- 가비지 켈렉션
  - ✓자바 가상 기계의 가비지 컬렉터가 자동으로 가비지 수집 반환

가비지 사례

```
Person a, b;
a = new Person("이몽룡");
b = new Person("성춘향");
b = a; // b가 가리키던 객체는 가비지가 됨
```

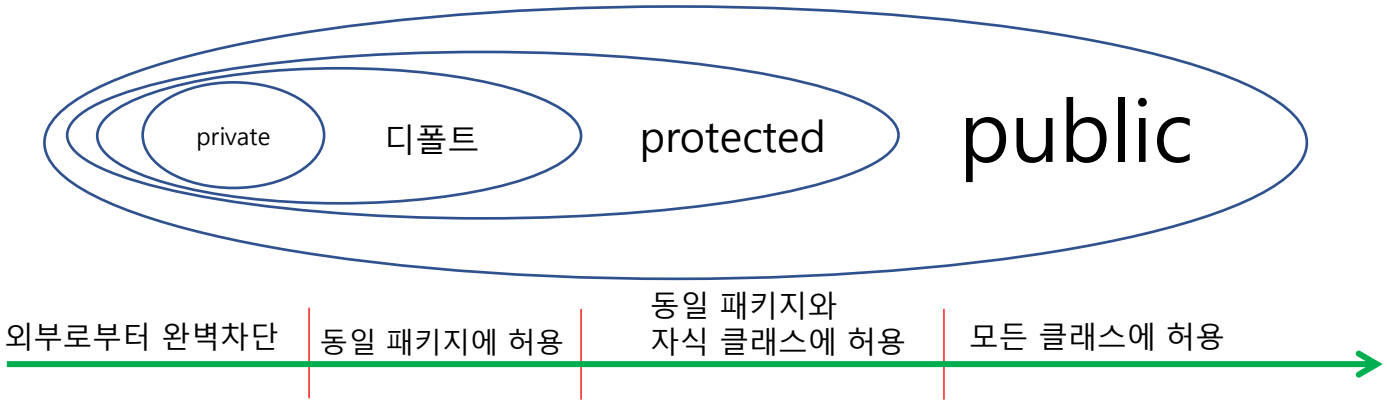


자바의 패키지 개념


De...
Pro...
Ser...
ch01
ch02
ch03
ch04
JRE System Library [JavaSE-1.
src
ch04
BankAccount.java
BankAccountConstructo
IntArray.java
BankAccount.java
AccessTestMain.java
BookArray.java
1 package ch04;
2
3 public class BankAccount {
4 String accNumber;
5 String accName;
6 int accBalance;
7
8 public BankAccount() {
9 System.out.println("기본 생성자");
10 };
11
12 public BankAccount(String accNumber, String accName) {
13 this(accNumber, accName, 1000);
14 System.out.println("생성자 2");
eclipse-workspace › ch04 › src › ch04
이름
BankAccount.java
BankAccountConstructor.java
New Java Class
Java Class
Create a new Java class.
Source folder: ch04/src
Package: ch04
Enclosing type:
Name:
Modifiers: public package private protected
abstract final static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Browse...
Add...
Remove
Finish
Cancel

접근 지정자

- 자바의 접근 지정자
  - ✓ 4가지
    - private, protected, public, 디폴트(접근지정자 생략)
- 접근 지정자의 목적
  - ✓ 클래스나 일부 멤버를 공개하여 다른 클래스에서 접근하도록 허용
  - ✓ 객체 지향 언어의 캡슐화 정책은 멤버를 보호하는 것
    - 접근 지정은 캡슐화에 묶인 보호를 일부 해제할 목적
- 접근 지정자에 따른 클래스나 멤버의 공개 범위

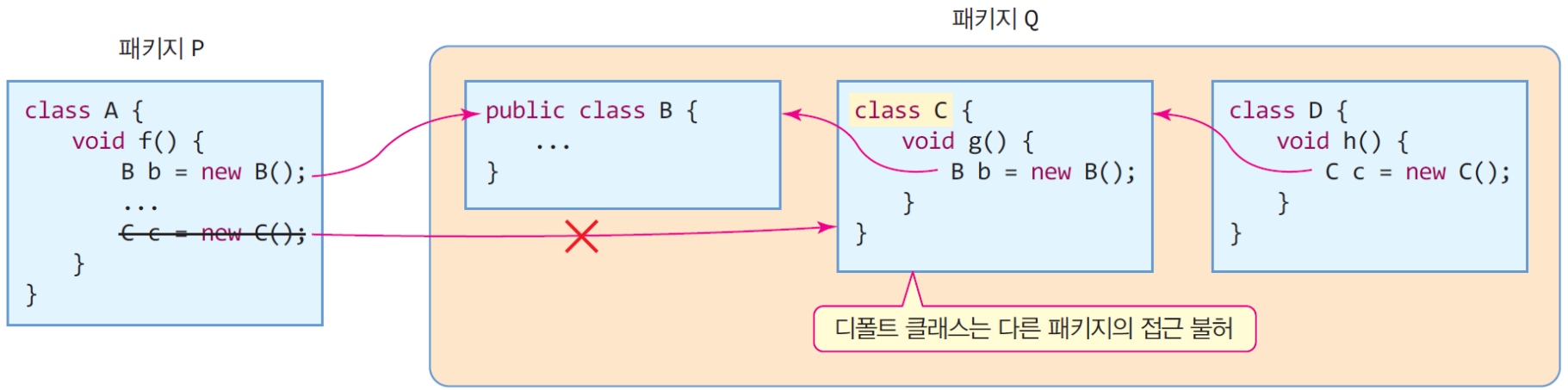


클래스 접근 지정

- 클래스 접근지정
  - ✓ 다른 클래스에서 사용하도록 허용할 지 지정
  - ✓ public 클래스
    - 다른 모든 클래스에게 접근 허용
  - ✓ 디폴트 클래스(접근지정자 생략)
    - package-private라고도 함
    - 같은 패키지의 클래스에만 접근 허용

```
public class World { // public 클래스
.....
}
```

```
class Local { // 디폴트 클래스
.....
}
```



public 클래스와 디폴트 클래스의 접근 사례

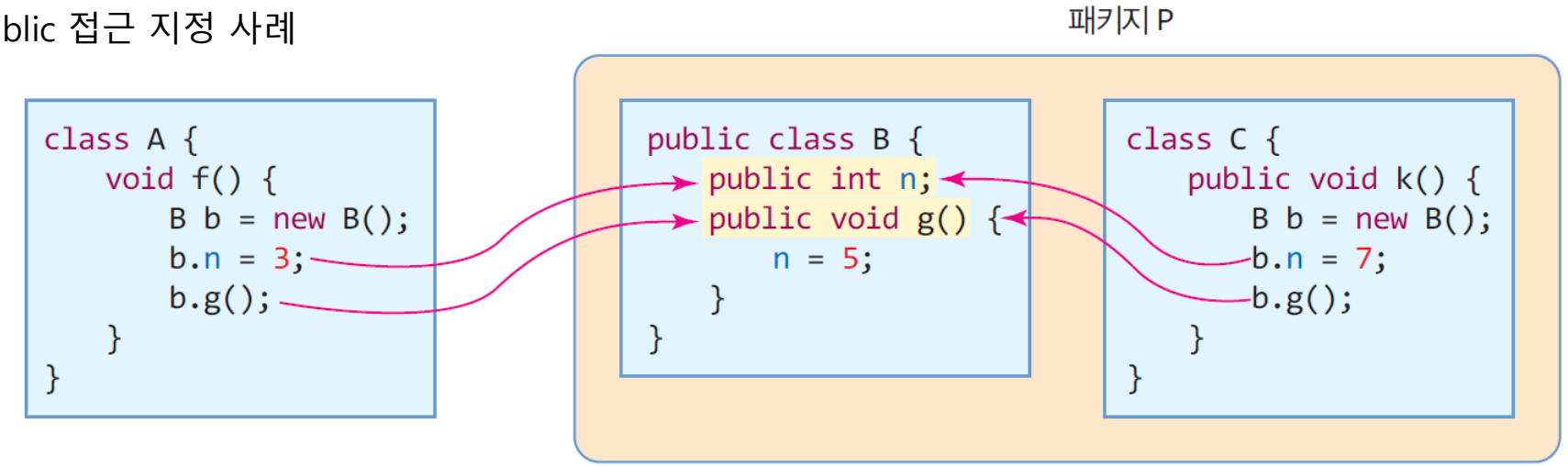
멤버 접근 지정

- ✓ public 멤버
  - 패키지에 관계 없이 모든 클래스에게 접근 허용
- ✓ private 멤버
  - 동일 클래스 내에만 접근 허용
  - 상속 받은 서브 클래스에서 접근 불가
- ✓ protected 멤버
  - 같은 패키지 내의 다른 모든 클래스에게 접근 허용
  - 상속 받은 서브 클래스는 다른 패키지에 있어도 접근 가능
- ✓ 디폴트(default) 멤버
  - 같은 패키지 내의 다른 클래스에게 접근 허용

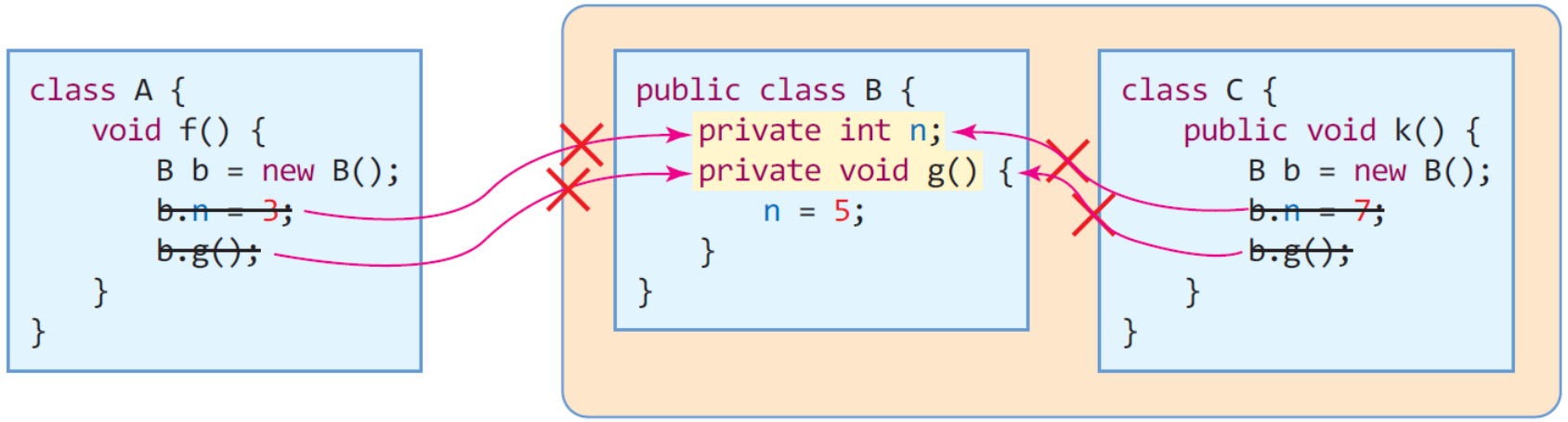
멤버에 접근하는 클래스	멤버의 접근 지정자			
	private	디폴트 접근 지정	protected	public
같은 패키지의 클래스	×	○	○	○
다른 패키지의 클래스	×	×	×	○
접근 가능 영역	클래스 내	동일 패키지 내	동일 패키지와 자식 클래스	모든 클래스

멤버 접근 지정자의 이해

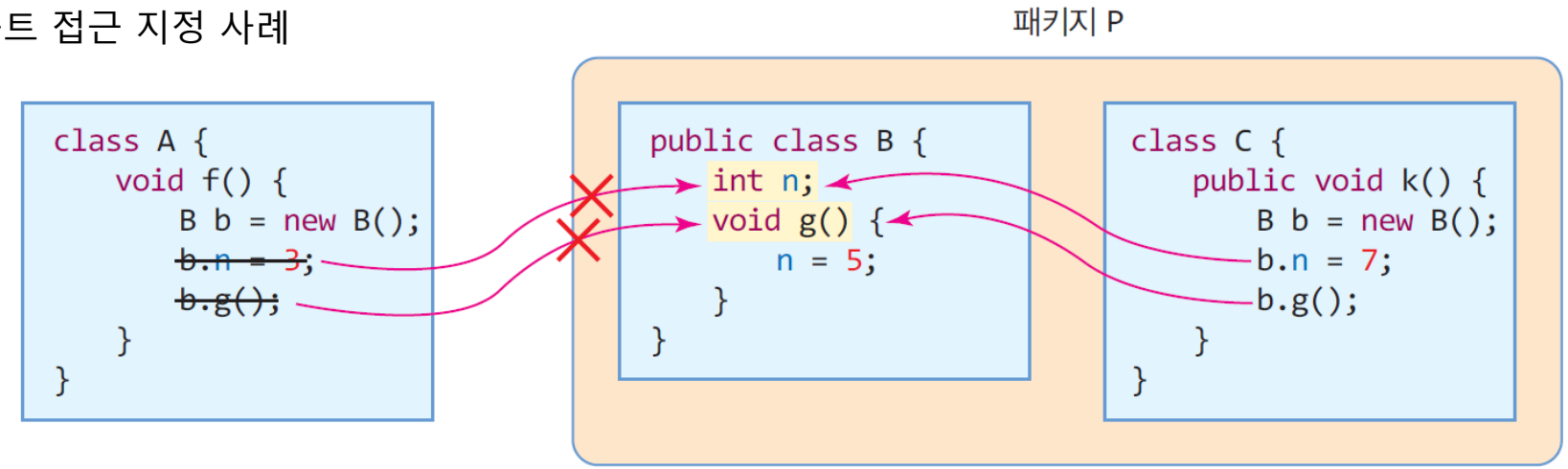
public 접근 지정 사례



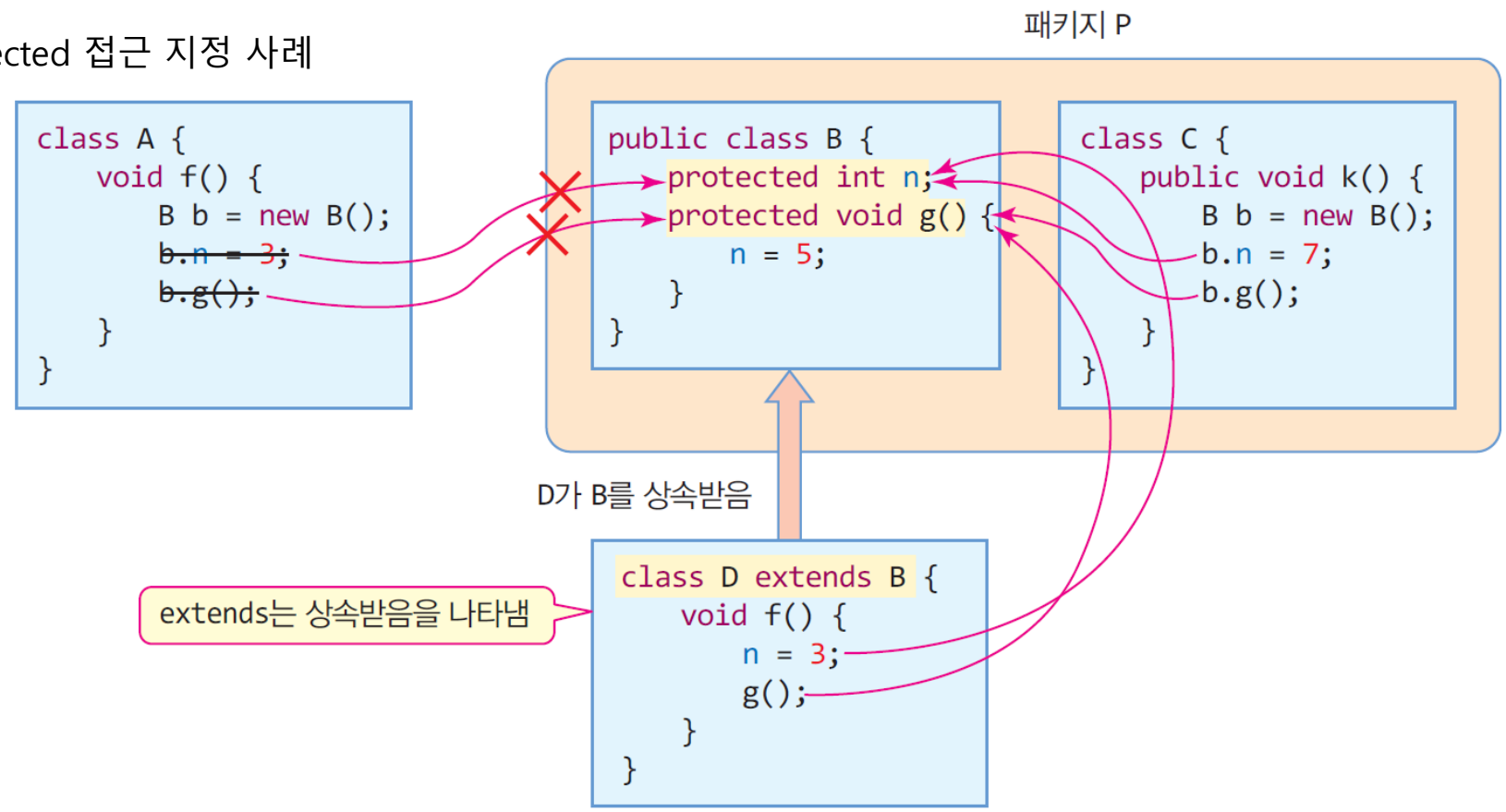
private 접근 지정 사례



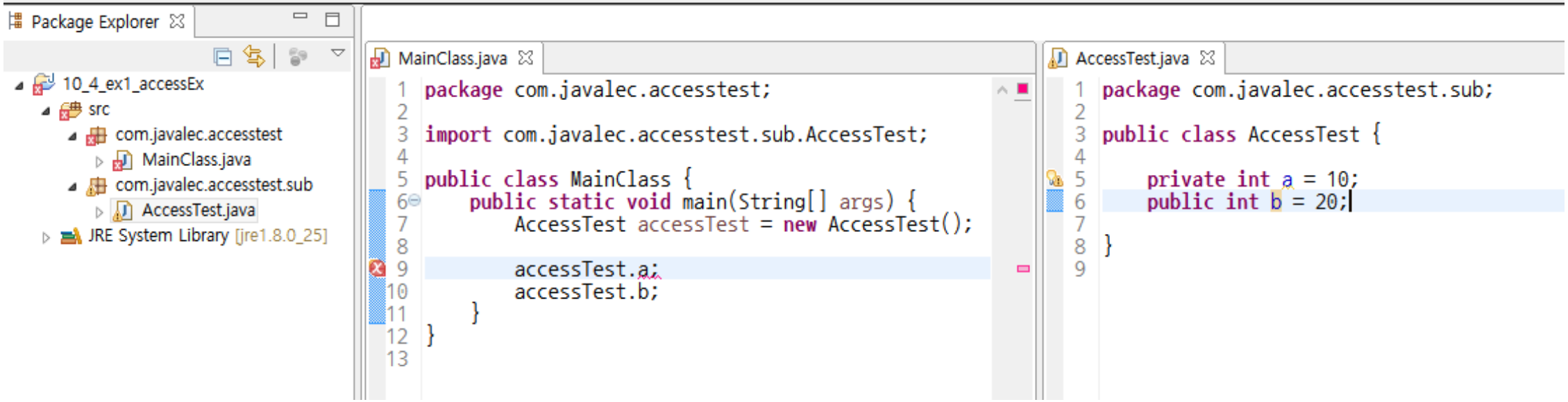
디폴트 접근 지정 사례



protected 접근 지정 사례



접근지정자 예


Package Explorer
10_4_ex1_accessEx
src
com.javalec.accesstest
MainClass.java
com.javalec.accesstest.sub
AccessTest.java
JRE System Library [jre1.8.0_25]
MainClass.java
1 package com.javalec.accesstest;
2
3 import com.javalec.accesstest.sub.AccessTest;
4
5 public class MainClass {
6 public static void main(String[] args) {
7 AccessTest accessTest = new AccessTest();
8
9 accessTest.a;
10 accessTest.b;
11 }
12 }
13
AccessTest.java
1 package com.javalec.accesstest.sub;
2
3 public class AccessTest {
4
5 private int a = 10;
6 public int b = 20;
7
8 }
9

import의 이해

- 다른 패키지의 클래스를 사용하고 싶을 때…