dongguk UNIVERSITY

객체지향프로그래밍

실습강의 7주차

실습강의 소개

- **실습 진행 방법**
  - 간단한 이론 복습 및 해당주차 실습과제 설명
  - 실습 후 보고서와 소스코드를 압축하여 수요일 자정(23:59)까지
    꼭!! 이클래스 제출(**이메일 제출 불가, 반드시 이클래스를 통해 제출**)
  - 실습 과제 제출 기한 엄수(제출 기한 이후로는 0점 처리)
  - **보고서는 출력하여 수업 시작 전에 제출(대면으로 실습 수업 진행 시)**
- **Q & A**
  - **이클래스 및 실습조교 이메일을 통해 질의 응답**
  - **이메일 제목 : [객체지향프로그래밍_홍길동] *본인 과목명과 성명 꼭 작성!**
  - **실습조교 메일 주소 : pmj6823@dgu.ac.kr**
  - **수업용 깃허브 조직 : https://github.com/23-2-Object-Oriented-Programming**

실습강의 소개

- **실습 보고서**
  - 문제 분석, 프로그램 설계 및 알고리즘, 소스코드 및 주석, 결과 및 결과 분석, 소감
- **제출 방법**
  - 보고서, 소스코드, 실행파일을 1개의 파일로 압축하여 e-class "과제" 메뉴를 통해 제출
    - **"이름학번실습주차.zip" 형태로 제출(e.g. :김동국19919876실습7.zip)**
    - 파일명에 공백, 특수 문자 등 사용 금지
  - 대면 수업일 경우 출력한 보고서를 실습 시간에 제출
- **유의 사항**
  - 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
  - 정해진 기한내 제출
    - 기한 넘기면 0점 처리
    - e-class가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일 전에 제출
    - **당일 e-class 오류로 인한 미제출은 불인정**
  - 소스코드, 보고서를 자신이 작성하지 않은 경우 **실습 전체 점수 0점 처리**
  - **Eclipse, 동국대학교 Shastra**를 사용하여 실습 진행

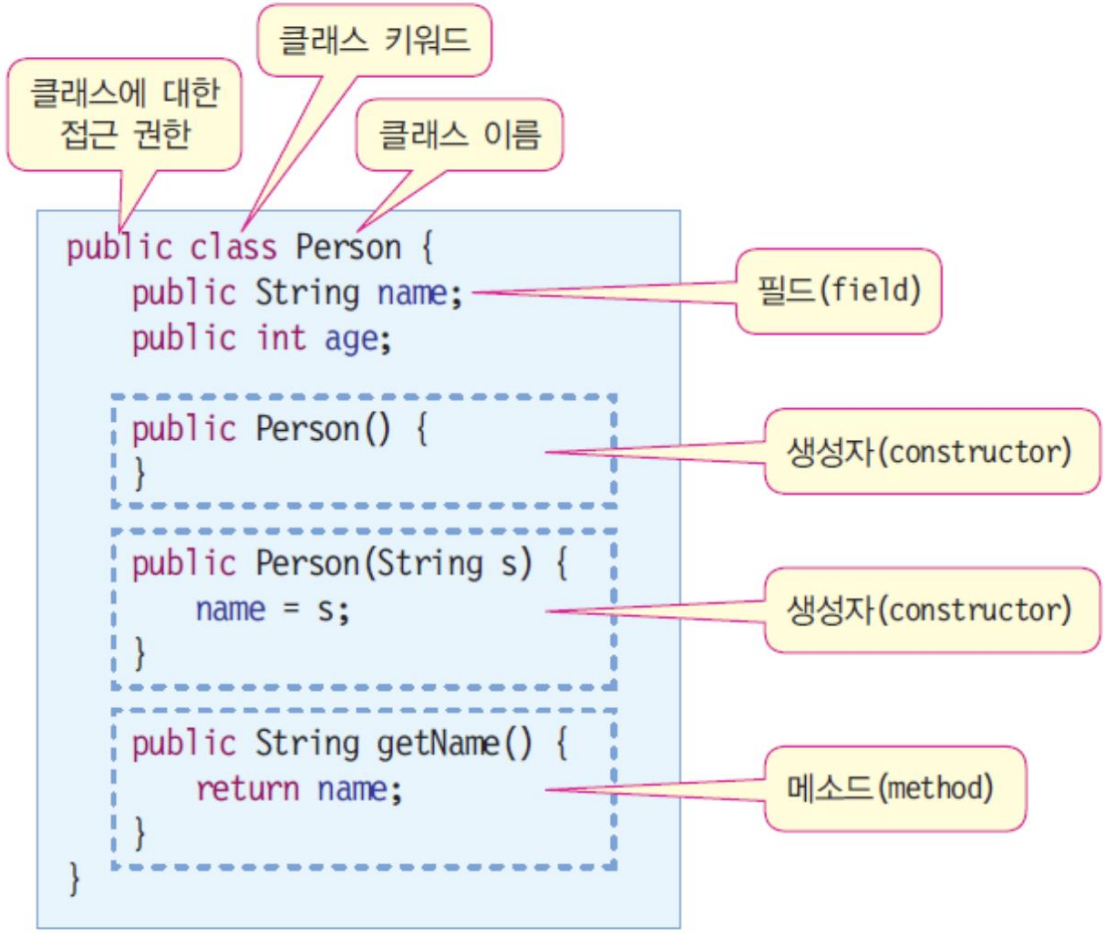
보고서 작성 방법

- **보고서 양식**
  - 문제 분석: 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술
  - 프로그램 설계 및 알고리즘
    - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
    - e.g.) 문제 해결 과정 및 핵심 알고리즘 기술
  - 소스코드 및 주석
    - 소스코드와 그에 해당하는 주석 첨부
    - 각각의 함수가 수행하는 작업, 매개변수, 반환 값 등을 명시
    - **소스코드 전체 첨부(소스코드 화면 캡처X, 소스코드는 복사/붙여넣기로 첨부)**
  - 결과 및 결과 분석
    - 결과 화면을 캡처하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
  - 소감
    - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

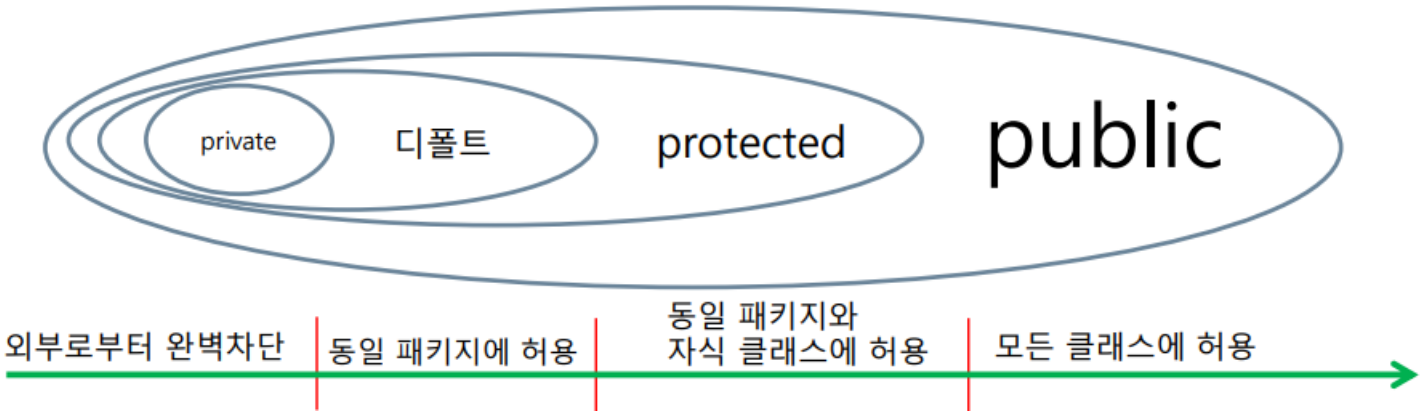
클래스와 객체

- **클래스(Class)**
  - 객체의 속성과 행위 선언
  - 객체의 설계도 혹은 틀
- **객체(Object, Instance)**
  - 클래스의 틀로 찍어낸 실체
    - 메모리 공간을 갖는 구체적인 실체
    - 클래스를 구체화한 객체를 인스턴스(instance)라고 부름
    - 객체와 인스턴스는 같은 뜻으로 사용
- **사례**
  - 클래스: 소나타 자동차, 객체: 출고된 실제 소나타 100대
  - 클래스: 벽시계, 객체: 우리집 벽에 걸린 벽시계들
  - 클래스: 책상, 객체: 우리가 사용중인 실제 책상들

클래스와 객체



접근 지정자



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

[연습 1] 클래스 접근 지정

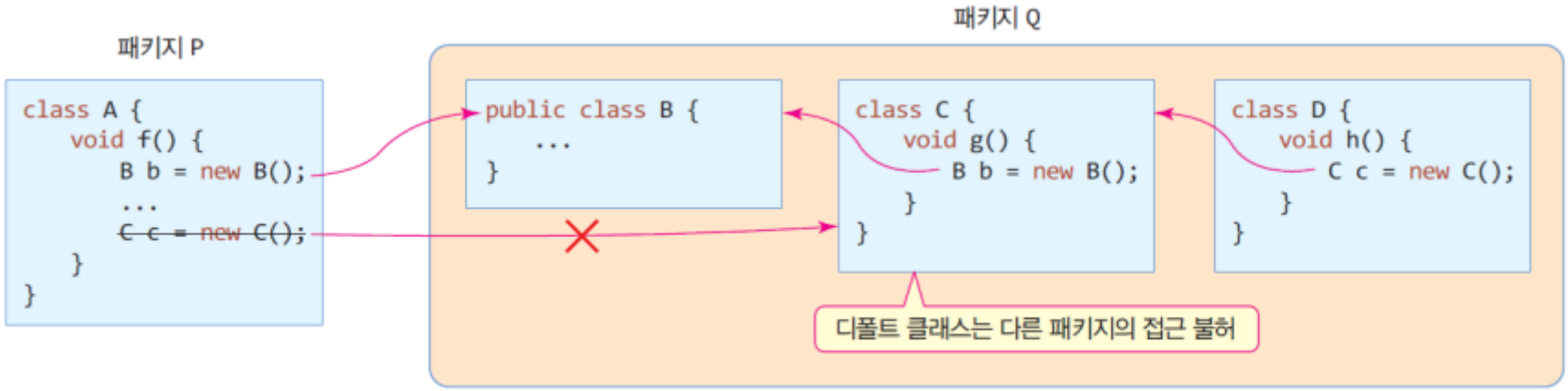
- class A의 main 함수를 실행시킬 경우 발생하는 컴파일 오류를 찾아 내고 이유를 설명하라.

```
package P;

import Q.*;

public class A {
    void f() {
        System.out.print("Initialize
default class B");
        B b = new B();

        System.out.print("Initialize
default class C");
        C c = new C();
    }
    public static void main(String[] args)
{
        A a = new A();
        a.f();
    }
}
```

```
package Q;

public class B {
}
```

```
package Q;
class C {
    void g() {
        B b = new B();
    }
}
```

[연습 1] 클래스 접근 지정



public 클래스와 디폴트 클래스의 접근 사례

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
        The type C is not visible
        The type C is not visible

        at P.A.f(A.java:12)
        at P.A.main(A.java:17)
```

[연습 2] 멤버 접근 지정

- 아래 코드의 컴파일 오류를 찾고 이유를 설명하라.

```
class Sample{
    public int a;
    private int b;
    int c;
}

public class AccessEx {
    public static void main(String[] args) {
        Sample aClass = new Sample();
        aClass.a = 10;
        aClass.b = 10;
        aClass.c = 10;
    }
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field Sample.b is not visible

        at AccessEx.main(AccessEx.java:13)
```

static 멤버

	non-static 멤버	static 멤버

선언	class Sample {
	  int n;
	  void g() {…}
	}
		class Sample {
		  static int m;
		  static void g() {…}
		}

공간적 특성	멤버는 객체마다 별도 존재 • 인스턴스 멤버라고 부름	멤버는 클래스당 하나 생성 • 멤버는 객체 내부가 아닌 별도의 공간에 생성 • 클래스 멤버라고 부름
시간적 특성	객체 생성 시에 멤버 생성됨 • 객체가 생길 때 멤버도 생성 • 객체 생성 후 멤버 사용 가능 • 객체가 사라지면 멤버도 사라짐	클래스 로딩 시에 멤버 생성 • 객체가 생기기 전에 이미 생성 • 객체가 생기기 전에도 사용 가능 • 객체가 사라져도 멤버는 사라지지 않음 • 멤버는 프로그램이 종료될 때 사라짐
공유의 특성	공유되지 않음 • 멤버는 객체 내에 각각 공간 유지	동일한 클래스의 모든 객체들에 의해 공유됨

static 메소드의 제약 조건

- **static 메소드는 오직 static 멤버만 접근 가능**
  - 객체가 생성되지 않은 상황에서도 static 메소드는 실행될 수 있기 때문에, non-static 멤버 활용 불가
  - non-static 메소드는 static 멤버 사용 가능
- **static 메소드는 this 사용 불가**
  - static 메소드는 객체 없이도 사용 가능하므로, this 래퍼런스 사용할 수 없음.

```
class StaticMethod {
    int n;
    void f1(int x) {n = x;} // 정상
    void f2(int x) {m = x;} // 정상
    static int m;
오류  static void s1(int x) {n = x;} // 컴파일 오류. static 메소드는 non-static 필드 n 사용 불가
오류  static void s2(int x) {f1(3);} // 컴파일 오류. static 메소드는 non-static 메소드 f1() 사용 불가
    static void s3(int x) {m = x;} // 정상. static 메소드는 static 필드 m 사용 가능
    static void s4(int x) {s3(3);} // 정상. static 메소드는 static 메소드 s3() 호출 가능
}
```

```
오류 static void f() { this.n = x;} // 오류. static 메소드에서는 this 사용 불가능
오류 static void g() { this.m = x;} // 오류. static 메소드에서는 this 사용 불가능
```

[연습 3] static 멤버를 가진 클래스 사용

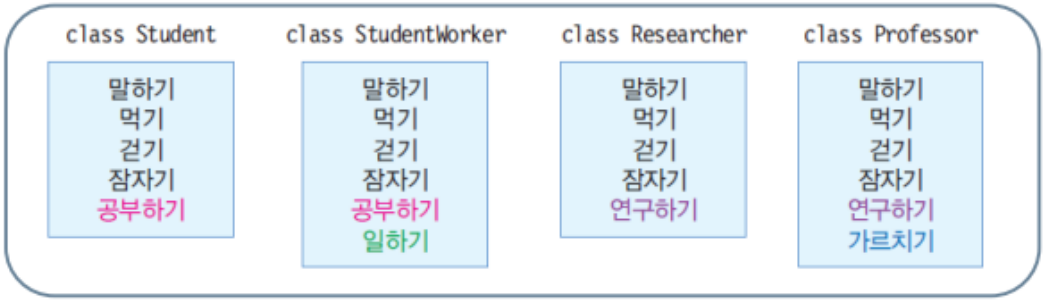
- 전역 함수로 작성하고자 하는 abs, max, min의 3개 함수를 static 메소드를 작성하고 호출하는 사례를 보여라

```
class Calc{
    public static int abs(int a) {return a>0?a:-a;}
    public static int max(int a, int b) {return (a>b)?a:b;}
    public static int min(int a, int b) {return (a>b)?b:a;}
}

public class CalcEx {
    public static void main(String[] args) {
        System.out.println(Calc.abs(-5));
        System.out.println(Calc.max(10, 8));
        System.out.println(Calc.min(-3, -8));
    }
}
```

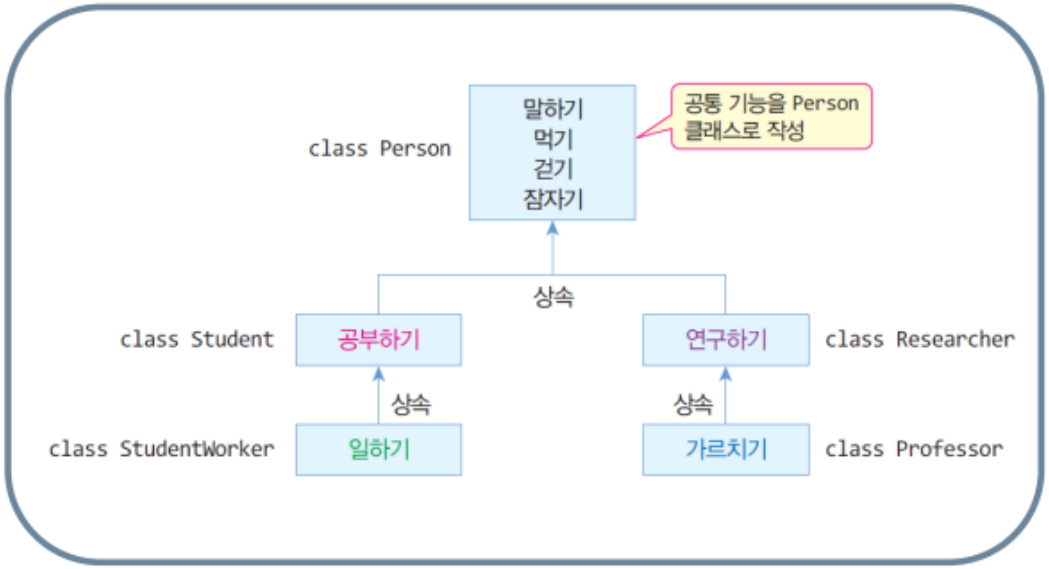
```
5
10
-8
```

상속

- **상속의 필요성**



상속이 없는 경우
중복된 멤버를 가진
4 개의 클래스



상속을 이용한
경우 중복이 제거되고
간결해진 클래스 구조

[연습 4] 클래스 상속

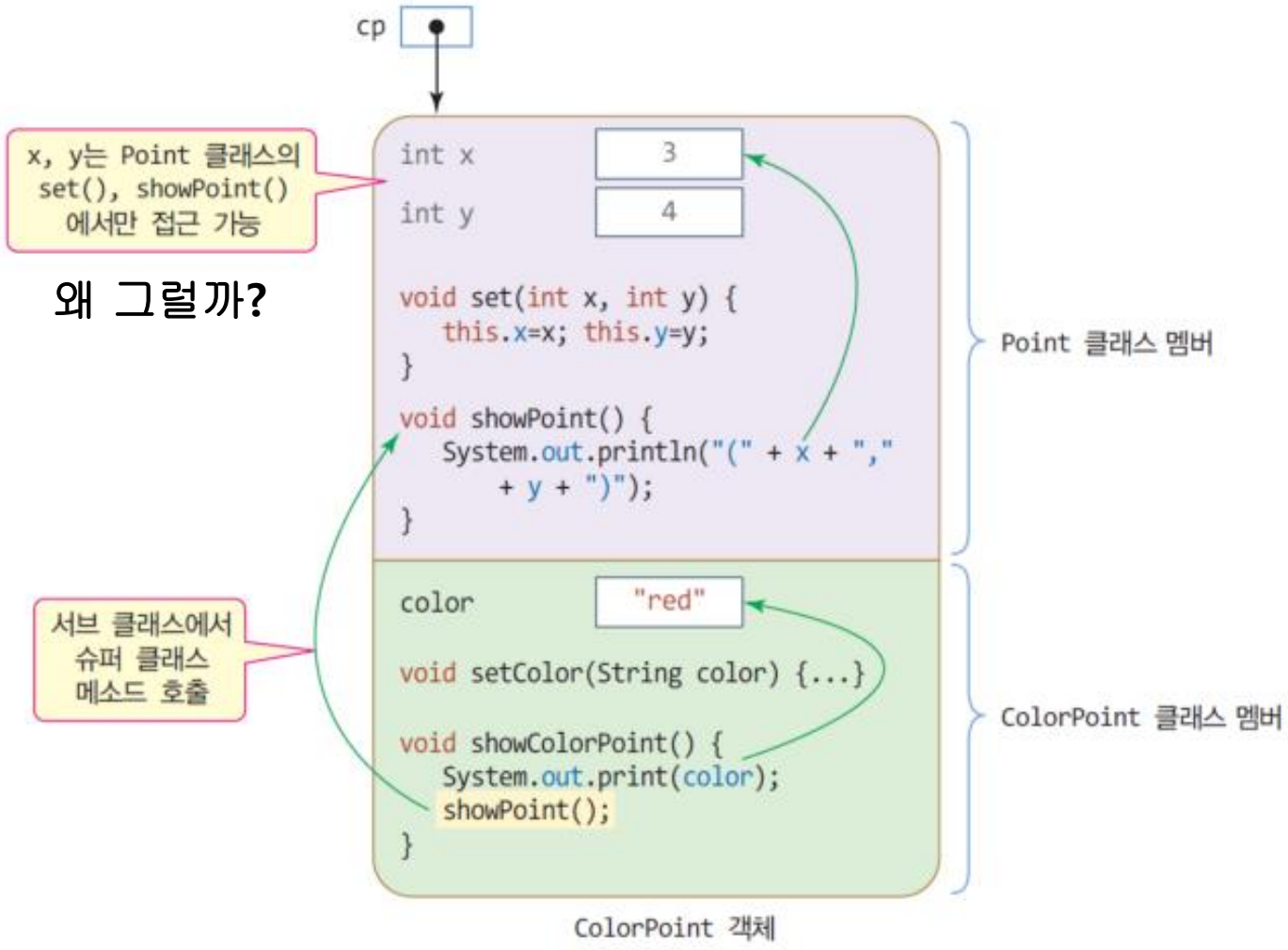
- 전역 함수로 작성하고자 하는 abs, max, min의 3개 함수를 static 메소드를 작성하고 호출하는 사례를 보여라

```
class Point {
    private int x, y; // 한 점을 구성하는 x, y 좌표
    public void set(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public void showPoint() {
        System.out.println("(" + x + "," + y + ")");
    }
}
class ColorPoint extends Point{
    private String color;
    public void setColor(String color) {
        this.color = color;
    }
    public void showColorPoint() {
        System.out.print(color);
        showPoint();
    }
}
```

```
public class ColorPointEx {
    public static void main(String[] args)
{
        Point p = new Point();
        p.set(1, 2);
        p.showPoint();
        ColorPoint cp = new ColorPoint();
        cp.set(3, 4);
        cp.setColor("red");
        cp.showColorPoint();
    }
}
```

```
(1,2)
red(3,4)
```

서브 클래스에서 슈퍼 클래스 멤버 접근



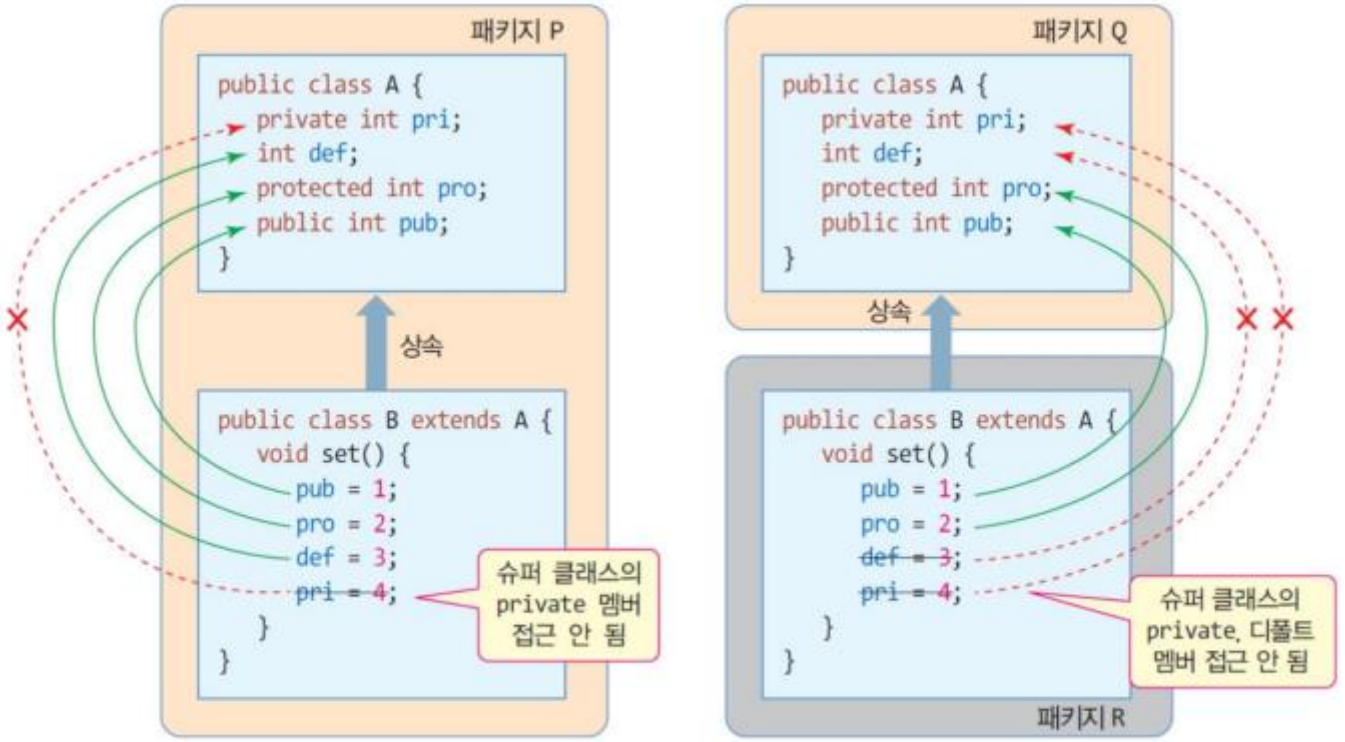
슈퍼 클래스 멤버의 접근 지정자

슈퍼 클래스 멤버에 접근하는 클래스 종류	슈퍼 클래스 멤버의 접근 지정자			
	private	디폴트	protected	public
같은 패키지의 클래스	×	○	○	○
다른 패키지의 클래스	×	×	×	○
같은 패키지의 서브 클래스	×	○	○	○
다른 패키지의 서브 클래스	×	×	○	○

(○는 접근 가능함을, ×는 접근 불가능함을 뜻함)

protected 멤버

- **protected 멤버에 대한 접근**



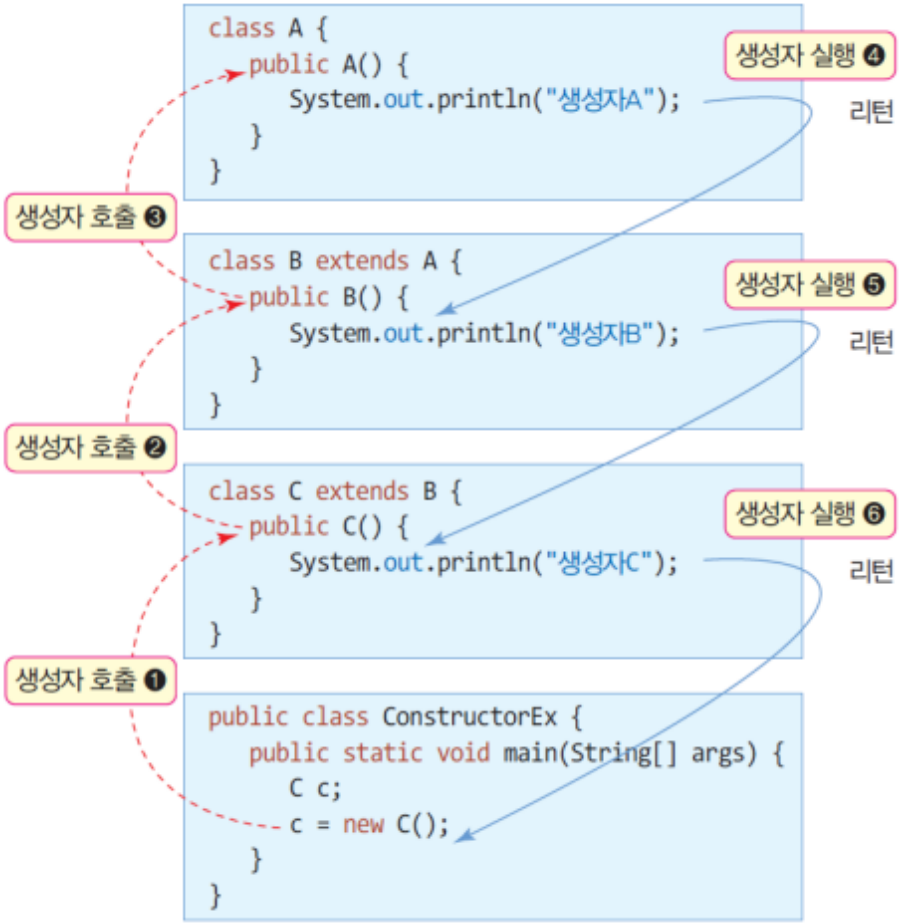
(a) 슈퍼 클래스와 서브 클래스가 동일한 패키지에 있는 경우

(b) 슈퍼 클래스와 서브 클래스가 서로 다른 패키지에 있는 경우

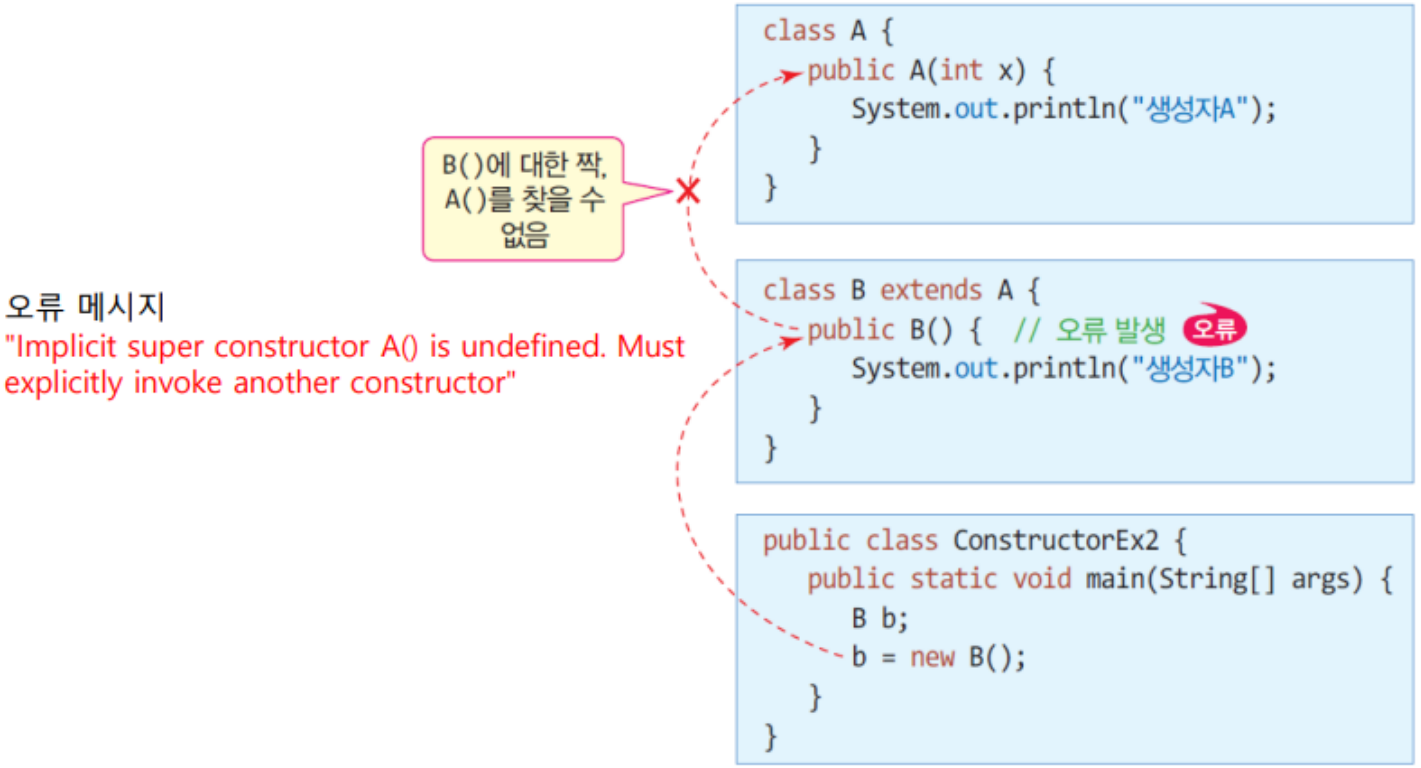
슈퍼 클래스의 생성자 결정 방식

- 개발자의 명시적 선택
  - 서브 클래스 개발자가 슈퍼 클래스의 생성자 명시적 선택
  - super() 키워드를 이용하여 선택
- 컴파일러의 기본 생성자 선택
  - 서브 클래스 개발자가 슈퍼 클래스의 생성자를 선택하지 않는 경우

컴파일러의 기본 생성자 선택



컴파일러의 기본 생성자 선택



개발자의 명시적인 선택

```java
class A {
    public A() {
        System.out.println("생성자A");
    }
    public A(int x) {
        System.out.println("매개변수생성자A" + x);
    }
}

class B extends A {
    public B() {
        System.out.println("생성자B");
    }
    public B(int x) {
        super(x);  // 첫 줄에 와야 함
        System.out.println("매개변수생성자B" + x);
    }
}

public class ConstructorEx4 {
    public static void main(String[] args) {
        B b;
        b = new B(5);
    }
}
```

[연습 5] super()를 활용한 ColorPrint 작성

- super()를 이용하여 ColorPoint 클래스의 생성자에서 서브 클래스 Point의 생성자를 호출하는 예를 보인다.

```
class Point {
    private int x, y; // 한 점을 구성하는 x, y 좌표
    public Point() {
        this.x = this.y = 0;
    }
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public void showPoint() {
        System.out.println("(" + x + "," + y + ")");
    }
}
class ColorPoint extends Point{
    private String color;
    public ColorPoint(int x, int y, String color) {
        super(x, y);
        this.color = color;
    }
    public void showColorPoint() {
        System.out.print(color);
        showPoint();
    }
}
```

```
public class ColorPointEx {
    public static void main(String[] args) {
    ColorPoint cp = new ColorPoint(5, 6, "blue");
        cp.showColorPoint();
    }
}
```

```
(1,2)
red(3,4)
```

실습과제 안내 [1/2]

- 실습과제 1
  - 다음 메인 메소드와 실행 결과를 참고하여 TV를 상속받은 ColorTV 클래스를 작성하라. 단, 생성자는 사용하지 않는다.

- 처리조건
  - 멤버 접근 지정자
  - extends
  - 입출력

```
public class TV {
    private int size;
    public void setSize(int size) {this.size = size;}
    protected int getSize() {return size;}
}
...
public static void main(String[] args) {
    ColorTV myTV = new ColorTV();
    myTV.setSize(32);
    myTV.setColor(1024);
    myTV.printProperty();
}
```

- 실행결과

실습과제 안내 [2/2]

- **실습과제 2**
  - 동아리에는 일반 동아리원과 집행부에 속한 학생이 있다. 일반 동아리원을 나타내는 Member 클래스와 집행부에 속한 학생을 의미하는 Crew 클래스를 구현하라. 단, Crew 클래스는 Member 클래스를 상속받는다. Member 클래스의 멤버 필드로는 학번과 이름이 있고, Member 정보를 출력하는 printMember() 멤버 메소드를 갖는다. Crew 클래스의 멤버 필드는 담당 분야(ex. president, secretary 등)이 있고 printCrew() 멤버 메소드는 Crew 정보를 Member 정보와 함께 출력한다.

- **처리조건**
  - extends
  - 접근 지정자
  - super()
  - 입출력

- **실행결과**

```
일반 동아리원 수를 입력하시오: 2
집행부 수를 입력하시오: 2

---Input Crew Information---
Student ID: 1234
Name: ST1

Student ID: 5678
Name: ST2


---Input Member Information---
Student ID: 1011
Name: CREW1
Department: PRESIDENT

Student ID: 1213
Name: CREW2
Department: SECRETARY
```

```
---Crew Information---
Student ID: 1011
Name: CREW1
Department: PRESIDENT

Student ID: 1213
Name: CREW2
Department: SECRETARY


---Member Information---
Student ID: 1234
Name: ST1

Student ID: 5678
Name: ST2
```