상 속

컴퓨터공학전공
박요한

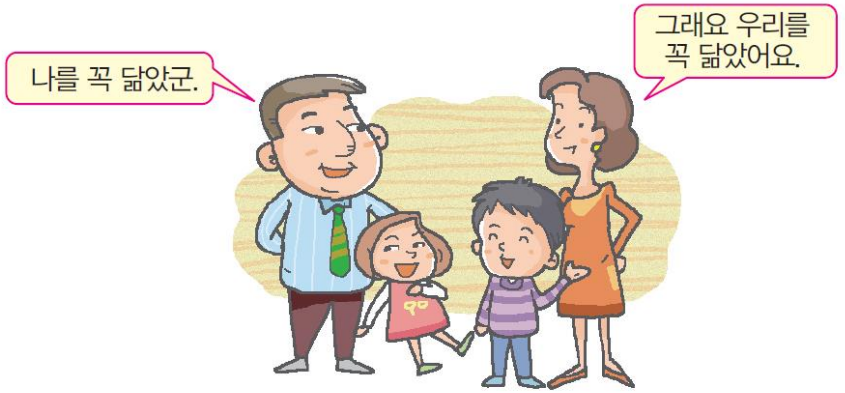
수업내용

- 상속의 개념과 사용법
- 상속과 접근지정자
- 상속과 생성자

상속 (inheritance)

- 객체 지향의 상속
  - ✓부모클래스에 만들어진 필드, 메소드를 자식클래스가 물려받음
  - ✓부모의 생물학적 특성을 물려받는 유전과 유사

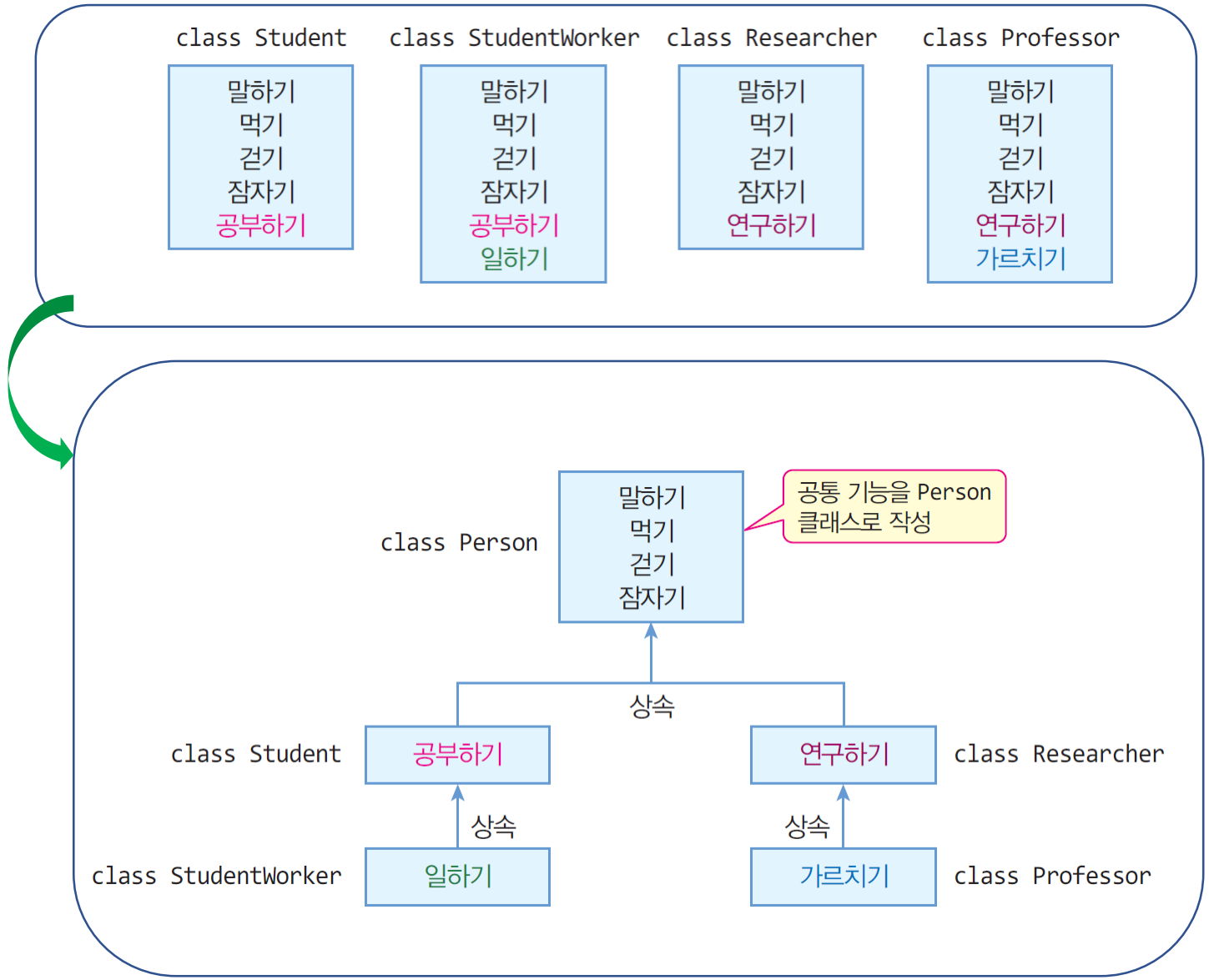
유산 상속



유전적 상속 : 객체 지향 상속

- 상속을 통해 간결한 자식 클래스 작성
  - ✓동일한 특성을 재정의할 필요가 없어 자식 클래스가 간결해짐

상속의 편리한 사례



상속이 없는 경우
중복된 멤버를 가진
4 개의 클래스

상속을 이용한
경우 중복이 제거되고
간결해진 클래스 구조

객체 지향에서 상속의 장점

- 클래스의 간결화
  - ✓멤버의 중복 작성 불필요
- 소프트웨어의 생산성 향상
  - ✓클래스 재사용과 확장 용이
  - ✓새로운 클래스의 작성 속도 빠름
- 클래스 관리 용이
  - ✓클래스들의 계층적 분류

클래스 상속과 객체

- 자바의 상속 선언

```
public class Person {
        …
}
public class Student extends Person { // Person을 상속받는 클래스 Student 선언
        …
}
public class StudentWorker extends Student { // Student를 상속받는 StudentWorker 선언
        …
}
```

✓부모 클래스 -> 슈퍼 클래스(super class)로 부름
✓자식 클래스 -> 서브 클래스(sub class)로 부름
✓extends 키워드 사용
  - 슈퍼 클래스를 확장한다는 개념

예제 5-1 : 클래스 상속 만들기 - Point와 ColorPoint 클래스

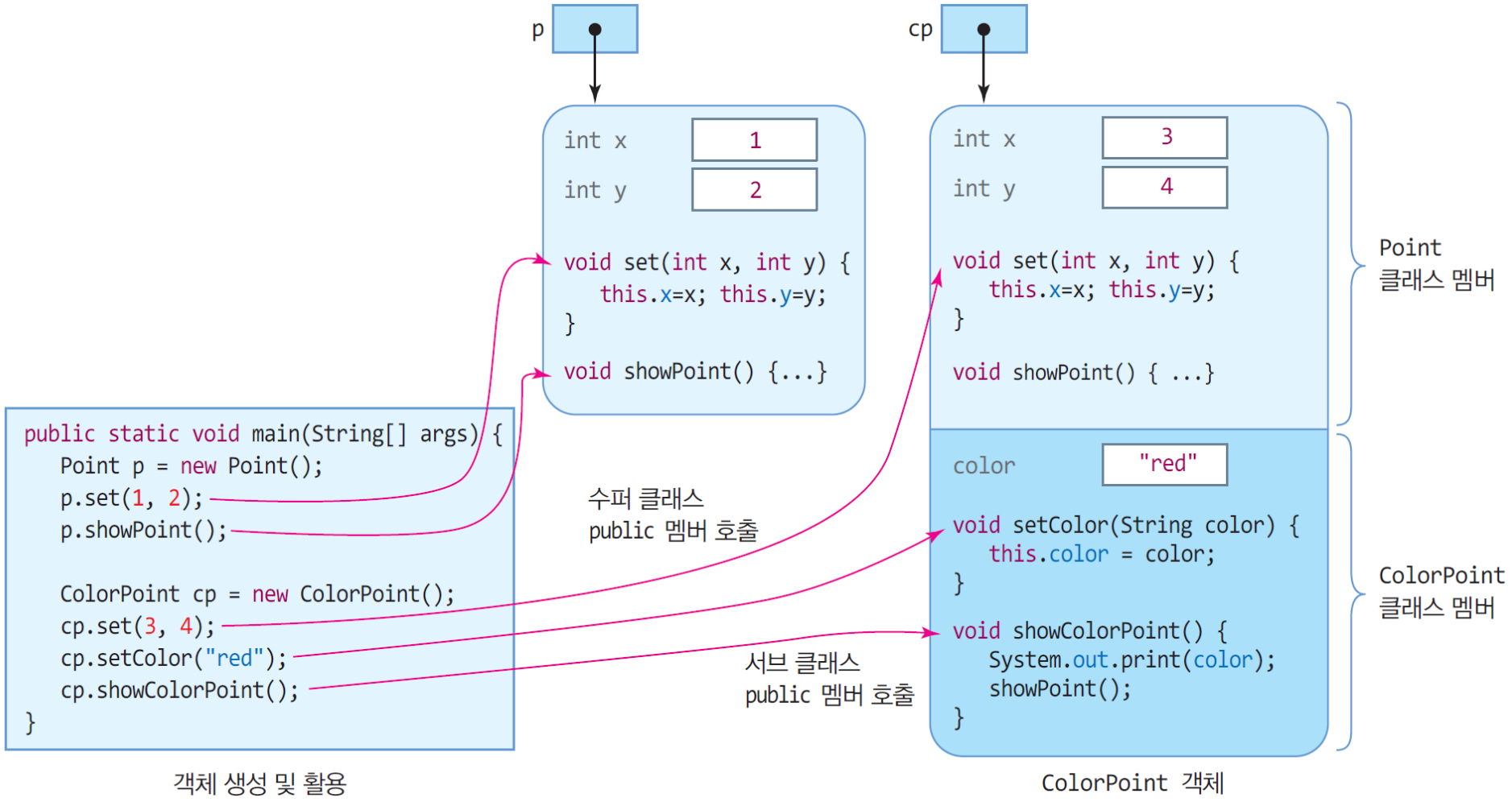
(x, y)의 한 점을 표현하는 Point 클래스와 이를 상속받아 색을 가진 점을 표현하는 ColorPoint 클래스를 만들고 활용해보자.

```
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() { // 점의 좌표 출력
        System.out.println("(" + x + "," + y + ")");
    }
}
```

```
// Point를 상속받은 ColorPoint 선언
class ColorPoint extends Point {
    private String color; // 점의 색
    public void setColor(String color) {
        this.color = color;
    }
    public void showColorPoint() { // 컬러 점의 좌표 출력
        System.out.print(color);
        showPoint(); // Point 클래스의 showPoint() 호출
    }
}
```

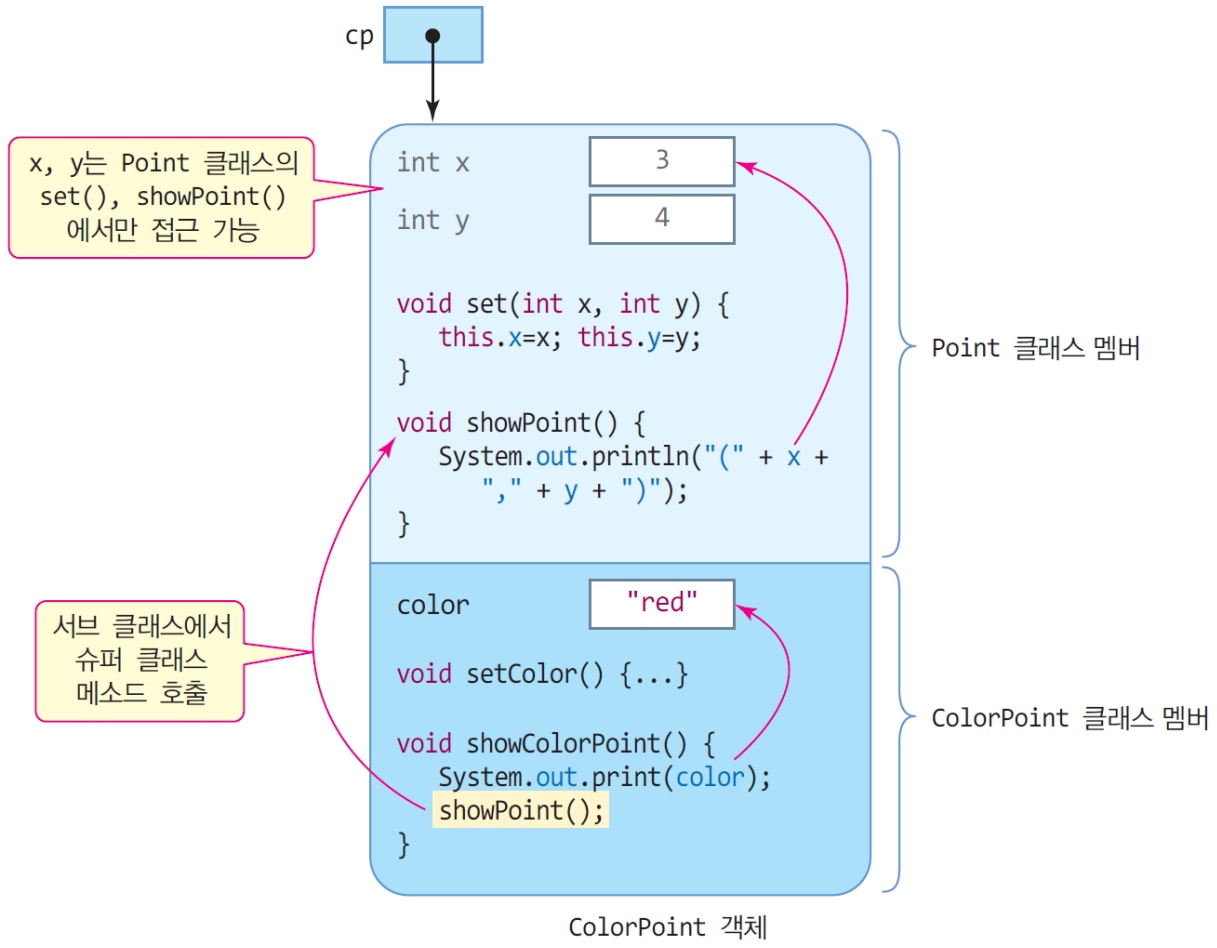
```
public class ColorPointEx {
    public static void main(String [] args) {
        Point p = new Point(); // Point 객체 생성
        p.set(1, 2); // Point 클래스의 set() 호출
        p.showPoint();

        ColorPoint cp = new ColorPoint(); // ColorPoint 객체
        cp.set(3, 4); // Point의 set() 호출
        cp.setColor("red"); // ColorPoint의 setColor() 호출
        cp.showColorPoint(); // 컬러와 좌표 출력
    }
}
```

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



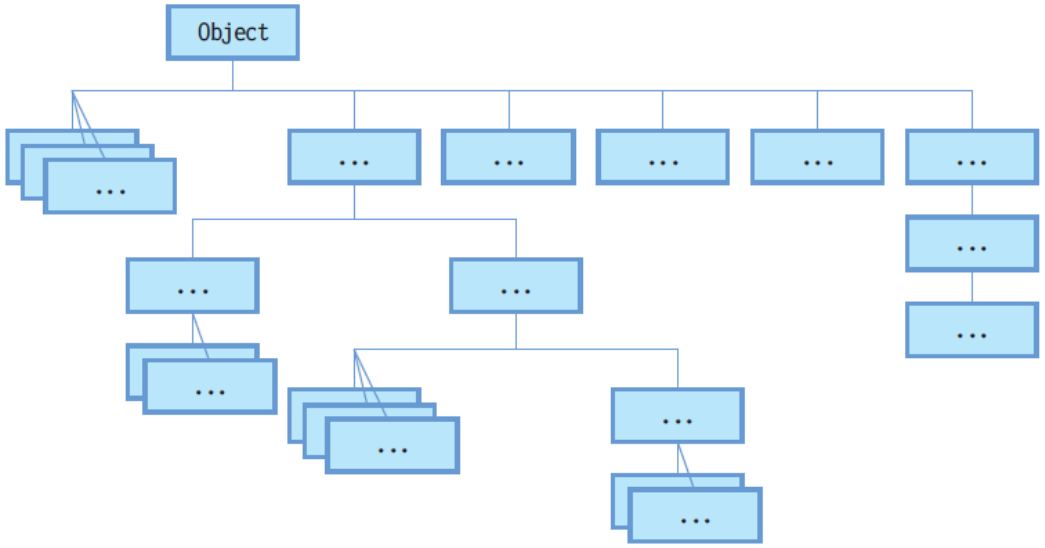
* new ColorPoint()에 의해 생긴
서브 클래스 객체에 주목

서브클래스에서 슈퍼 클래스의 멤버 접근



자바 상속의 특징

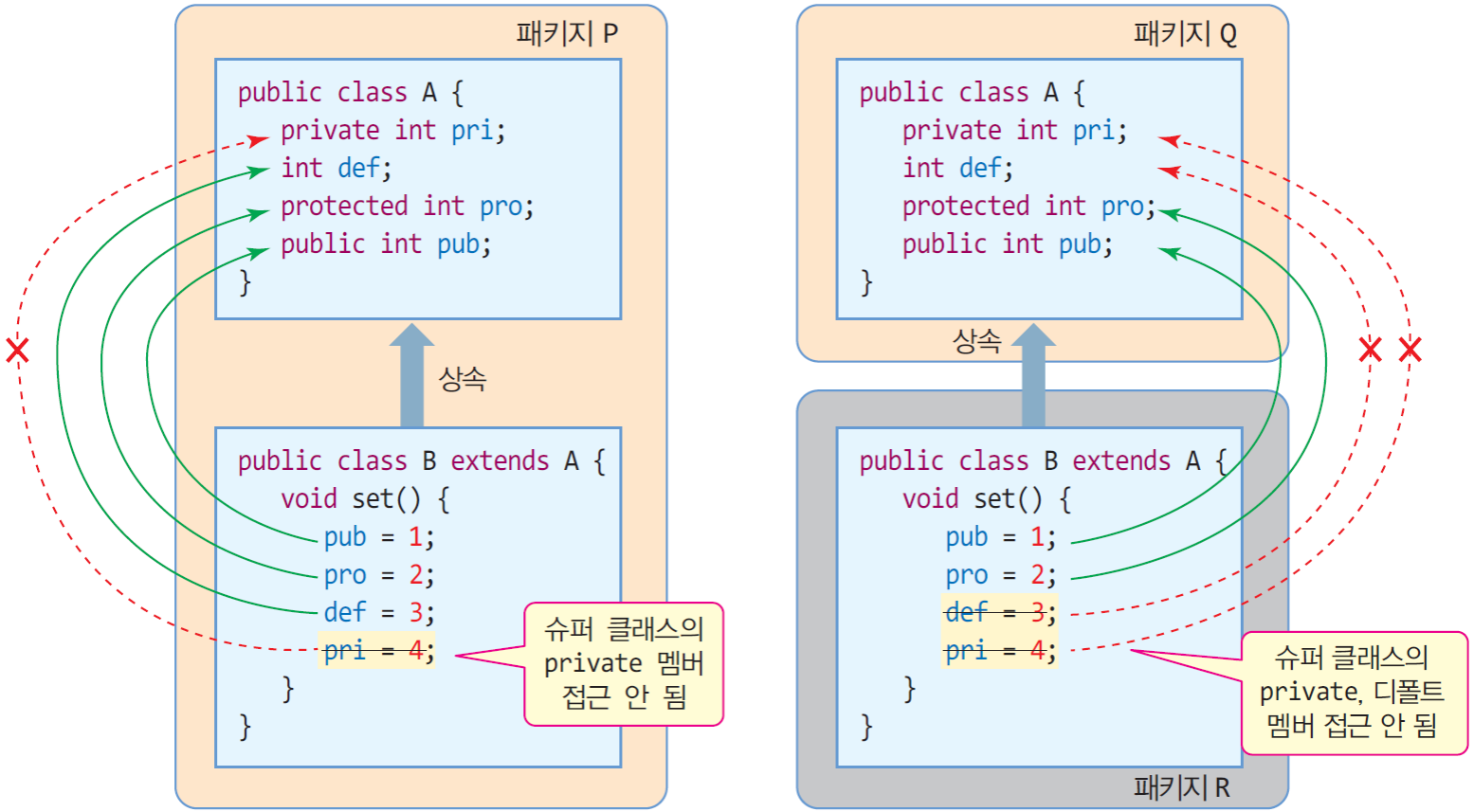
- 클래스의 다중 상속 지원하지 않음
- 상속 횟수 무제한
- 상속의 최상위 조상 클래스는 java.lang.Object 클래스
  - 모든 클래스는 자동으로 java.lang.Object를 상속받음
  - 자바 컴파일러에 의해 자동으로 이루어짐



상속과 접근 지정자

- 자바의 접근 지정자 4 가지
  - ✓public, protected, 디폴트, private
    - 상속 관계에서 주의할 접근 지정자는 private와 protected
- 슈퍼 클래스의 private 멤버
  - ✓슈퍼 클래스의 private 멤버는 다른 모든 클래스에 접근 불허
  - ✓클래스내의 멤버들에게만 접근 허용
- 슈퍼 클래스의 디폴트 멤버
  - ✓슈퍼 클래스의 디폴트 멤버는 패키지내 모든 클래스에 접근 허용
- 슈퍼 클래스의 public 멤버
  - ✓슈퍼 클래스의 public 멤버는 다른 모든 클래스에 접근 허용
- 슈퍼 클래스의 protected 멤버
  - ✓같은 패키지 내의 모든 클래스 접근 허용
  - ✓다른 패키지에 있어도 서브 클래스는 슈퍼 클래스의 protected 멤버 접근 가능

슈퍼 클래스의 멤버에 대한 서브 클래스의 접근



(a) 슈퍼 클래스와 서브 클래스가 동일한 패키지에 있는 경우

(b) 슈퍼 클래스와 서브 클래스가 서로 다른 패키지에 있는 경우

상속과 생성자

- new에 의해 서브 클래스의 객체가 생성될 때
  - ✓슈퍼클래스 생성자와 서브 클래스 생성자 모두 실행됨
  - ✓호출 순서
    - 서브 클래스의 생성자가 먼저 호출, 서브 클래스의 생성자는 실행 전 슈퍼 클래스 생성자 호출
  - ✓실행 순서
    - 슈퍼 클래스의 생성자가 먼저 실행된 후 서브 클래스의 생성자 실행

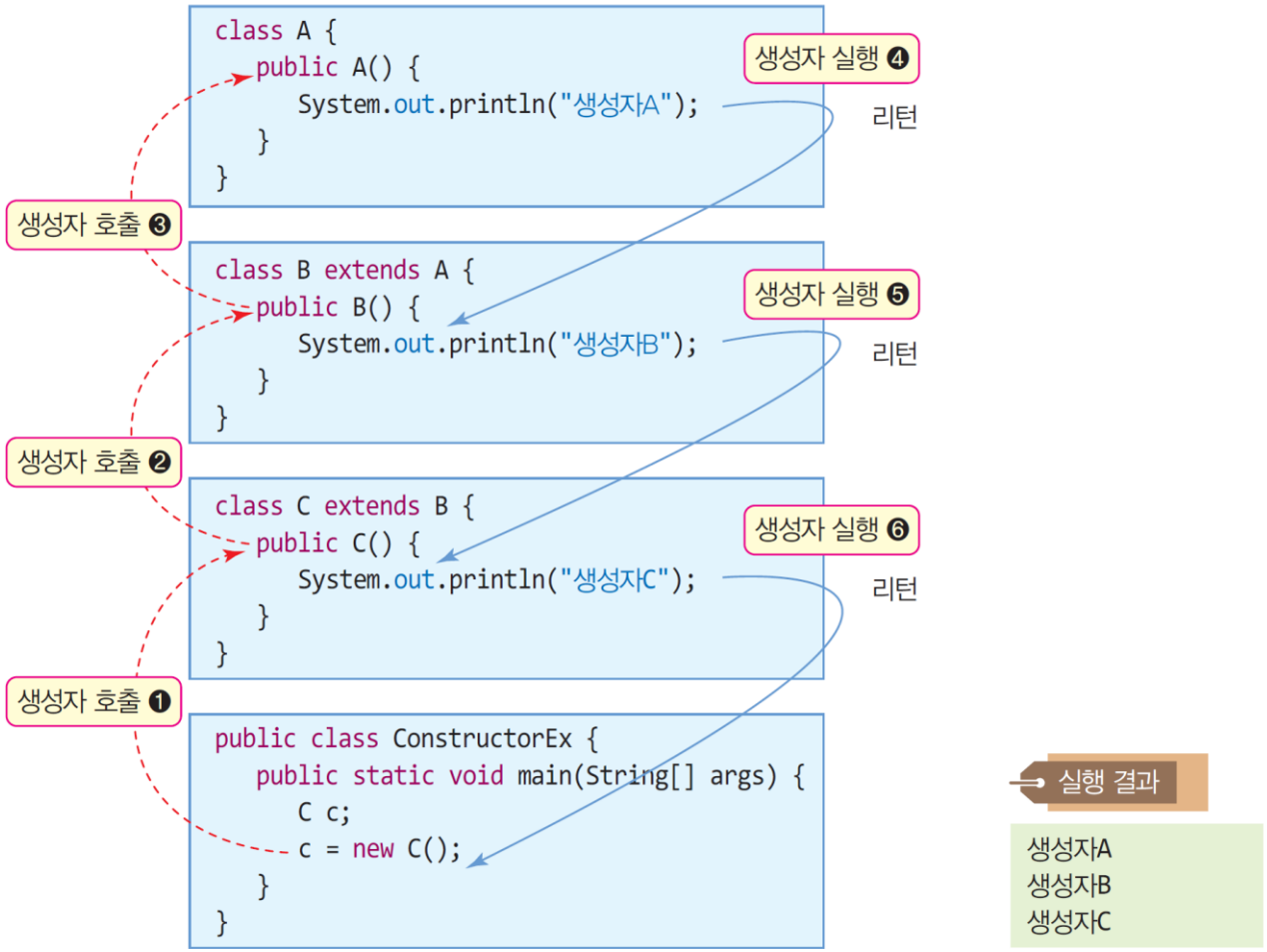
질문 1 서브 클래스 객체가 생성될 때 서브 클래스의 생성자와 슈퍼 클래스의 생성자가 모두 실행되는가? 아니면 서브 클래스의 생성자만 실행되는가?

답 둘 다 실행된다. 서브 클래스의 객체가 생성되면 이 객체 속에 서브 클래스와 멤버와 슈퍼 클래스의 멤버가 모두 들어 있다. 생성자의 목적은 객체 초기화에 있으므로, 서브 클래스의 생성자는 생성된 객체 속에 들어 있는 서브 클래스의 멤버 초기화나 필요한 초기화를 수행하고, 슈퍼 클래스의 생성자는 생성된 객체 속에 있는 슈퍼 클래스의 멤버 초기화나 필요한 초기화를 각각 수행한다.

질문 1 서브 클래스의 생성자와 슈퍼 클래스의 생성자 중 누가 먼저 실행되는가?

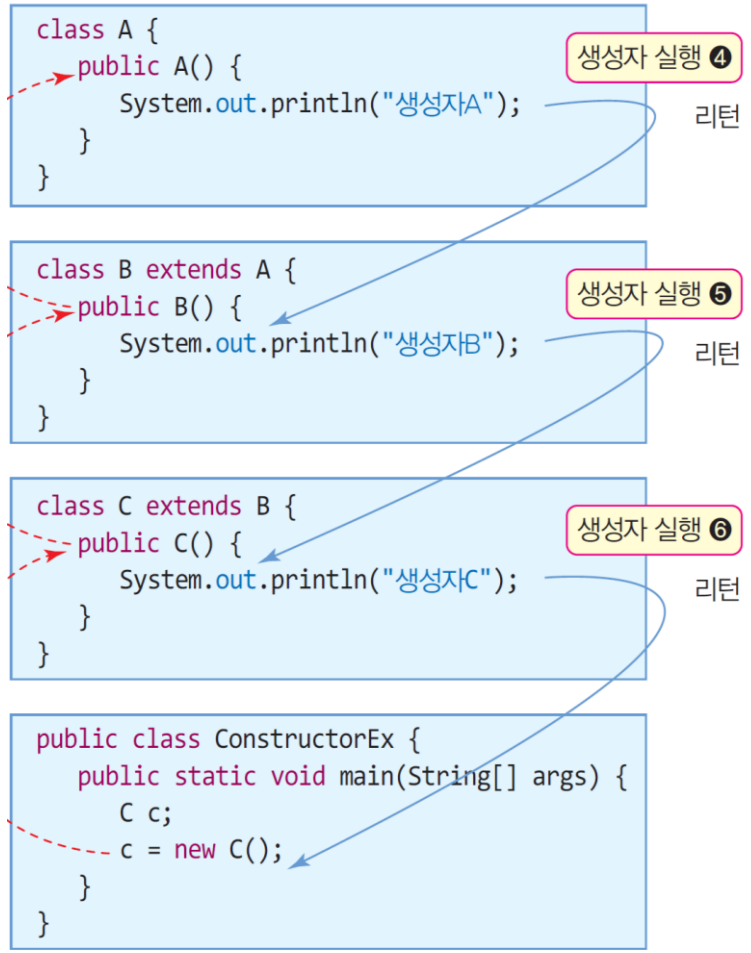
답 슈퍼 클래스의 생성자가 먼저 실행된 후 서브 클래스의 생성자가 실행된다.

슈퍼클래스와 서브 클래스의 생성자간의 호출 및 실행 관계



서브 클래스에서 슈퍼 클래스의 생성자 선택

- 상속 관계에서의 생성자
  - ✓ 슈퍼 클래스와 서브 클래스 각각 각각 여러 생성자 작성 가능
- 서브 클래스 생성자 작성 원칙
  - ✓ 서버 클래스 생성자에서 슈퍼 클래스 생성자 하나 선택
- 서브 클래스에서 슈퍼 클래스의 생성자를 선택하지 않는 경우
  - ✓ 컴파일러가 자동으로 슈퍼 클래스의 기본 생성자 선택
- 서브 클래스에서 슈퍼 클래스의 생성자를 선택하는 방법
  - ✓ super() 이용



슈퍼 클래스의 기본 생성자가 자동 선택

서브 클래스의 생성자가
슈퍼 클래스의 생성자를
선택하지 않은 경우

컴파일러는
서브 클래스의 기본
생성자에 대해
자동으로 슈퍼 클래스의
기본 생성자와 짝을 맺음

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

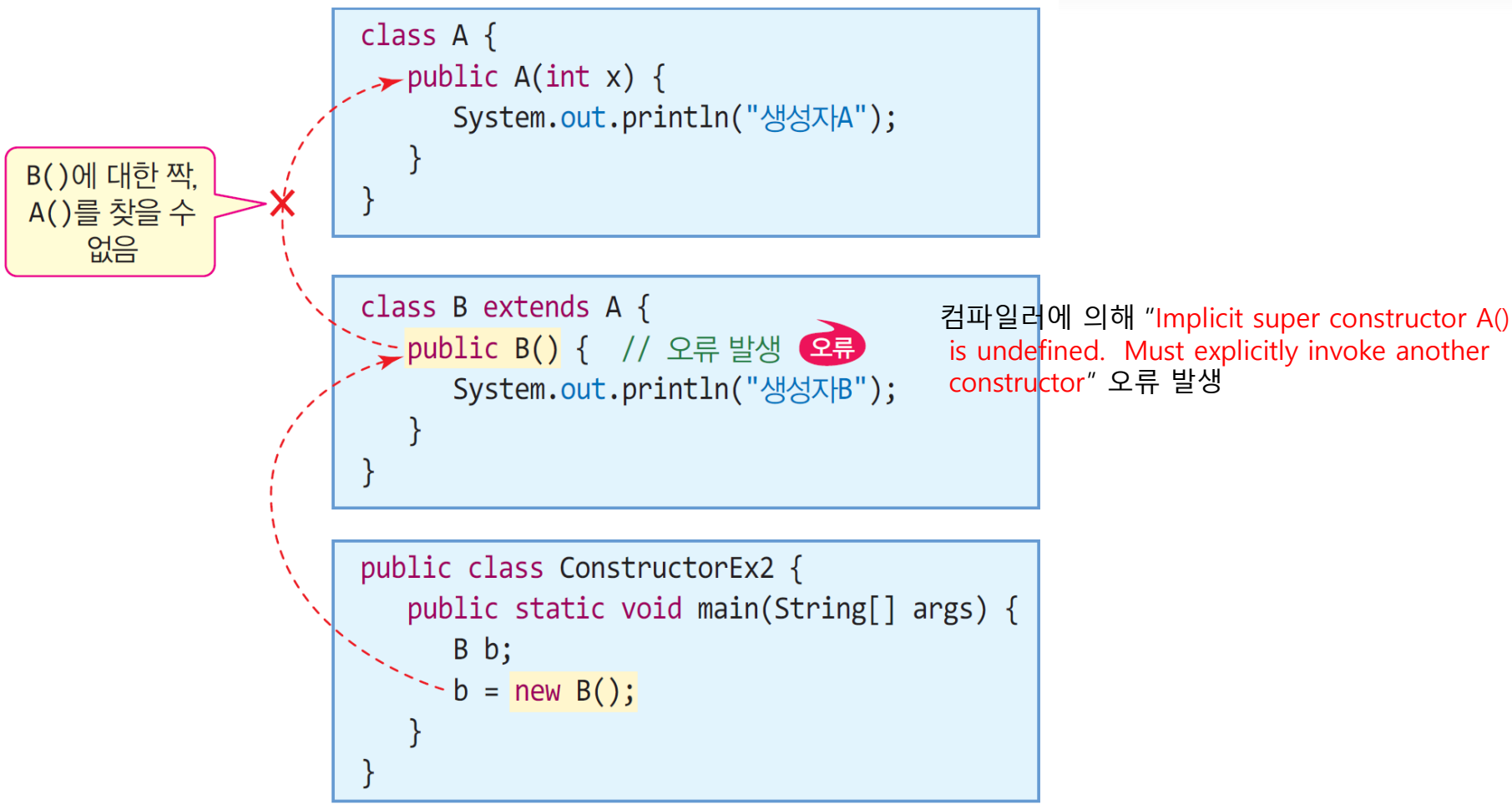
```
class B extends A {
    public B() {
        System.out.println("생성자B");
    }
}
```

```
public class ConstructorEx2 {
    public static void main(String[] args) {
        B b;
        b = new B();    // 생성자 호출
    }
}
```

실행 결과

```
생성자A
생성자B
```

슈퍼 클래스에 기본 생성자가 없어 오류 난 경우



서브 클래스에 매개변수를 가진 생성자

서브 클래스의 생성자가
슈퍼 클래스의 생성자를
선택하지 않은 경우

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

```
class B extends A {
    public B() {
        System.out.println("생성자B");
    }
    public B(int x) {
        System.out.println("매개변수생성자B");
    }
}
```

```
public class ConstructorEx3 {
    public static void main(String[] args) {
        B b;
        b = new B(5);
    }
}
```

실행 결과

```
생성자A
매개변수생성자B
```

super()를 이용하여 명시적으로 슈퍼 클래스 생성자 선택

- super()
  - ✓서브 클래스에서 명시적으로 슈퍼 클래스의 생성자 선택 호출
    - super(parameter);
    - 인자를 이용하여 슈퍼 클래스의 생성자 호출
    - 반드시 서브 클래스 생성자 코드의 제일 첫 라인에 와야 함

super()를 이용한 사례

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

x에 5 전달

```
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);
    }
}
```

x는 5

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

실행 결과

```
매개변수생성자A5
매개변수생성자B5
```

예제 5-3 : super()를 활용한 ColorPoint 작성

super()를 이용하여 ColorPoint 클래스의 생성자에서 슈퍼 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() { // 점의 좌표 출력
        System.out.println("(" + x + "," + y + ")");
    }
}

class ColorPoint extends Point {
    private String color; // 점의 색
    public ColorPoint(int x, int y, String color) {
        super(x, y); // Point의 생성자 Point(x, y) 호출
        this.color = color;
    }
    public void showColorPoint() { // 컬러 점의 좌표 출력
        System.out.print(color);
        showPoint(); // Point 클래스의 showPoint() 호출
    }
}
```

x=5, y=6 전달

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

x=5, y=6,
color = "blue" 전달

```
blue(5,6)
```