어서와 Java는 처음이지!

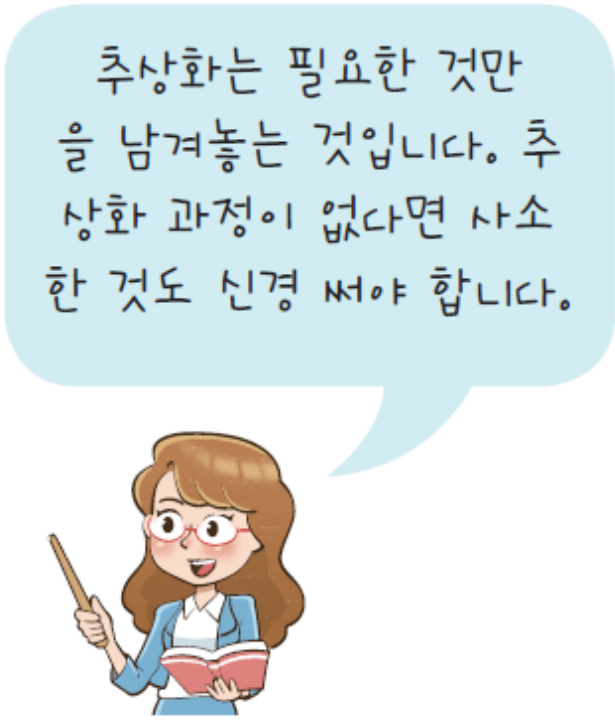
제7장 상속

- 상속
- Protected 접근자
- 메소드 오버라이딩

추상화

실제 객체

추상화된 객체

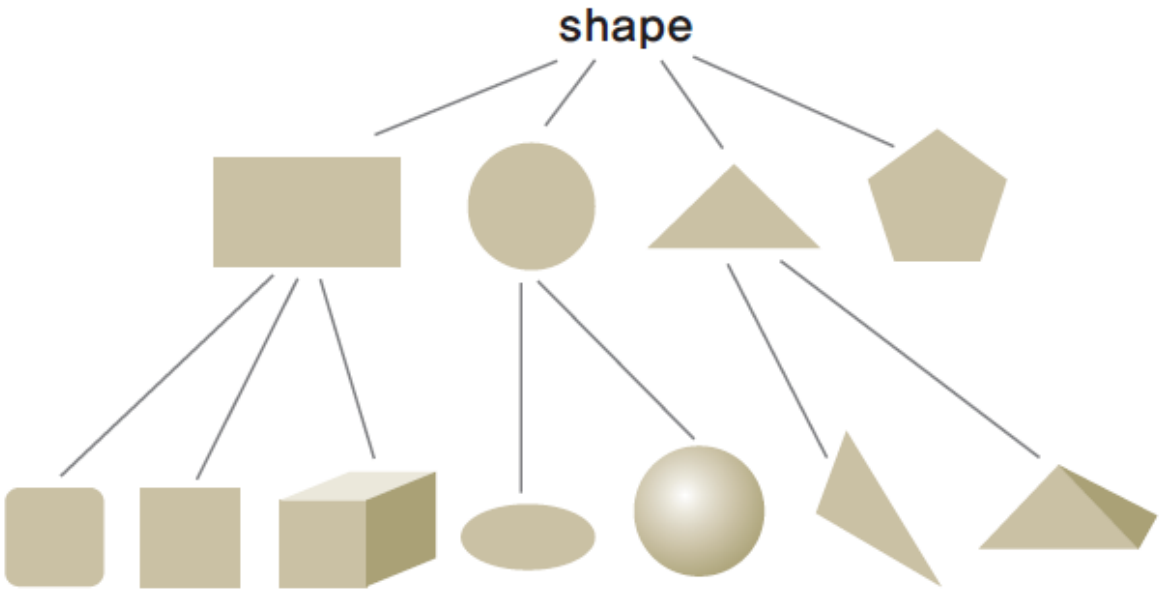


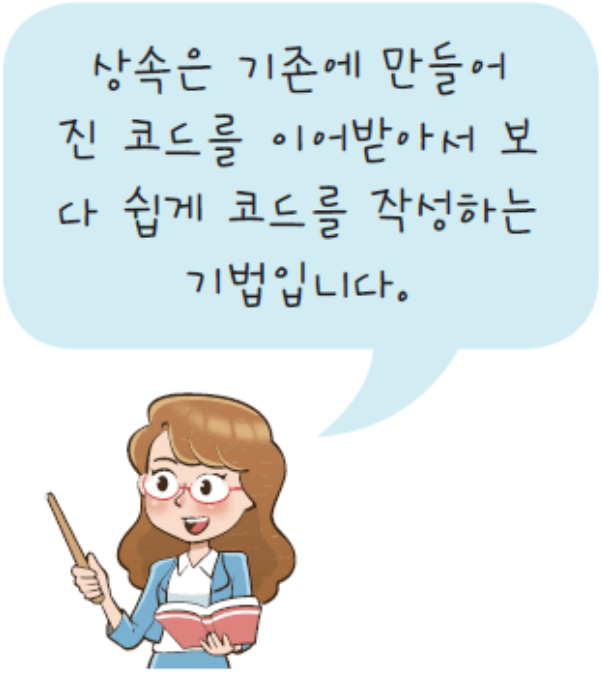
상속

- 상속(inheritance)
  - 이미 작성된 클래스(부모 클래스)를 이어받아서 새로운 클래스(자식 클래스)를 생성하는 기법

상속

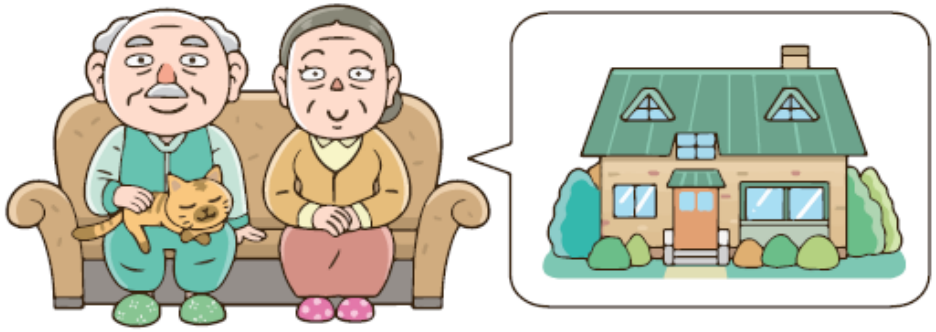
- 기존의 코드를 재활용하기 위한 기법



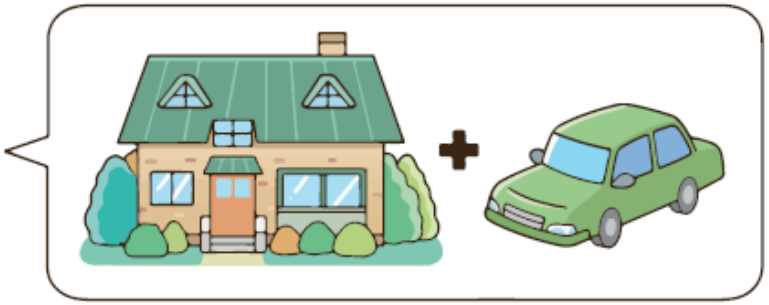


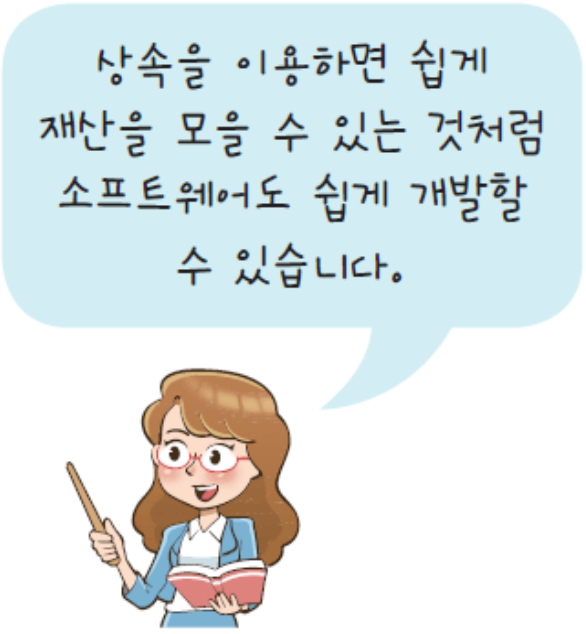
상속이란?

- 상속의 개념은 현실 세계에도 존재한다.



상속

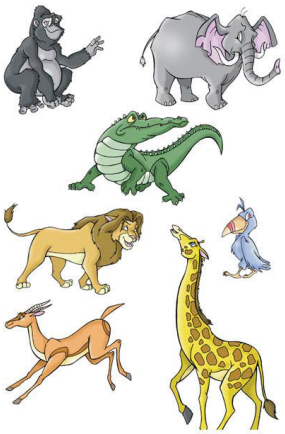


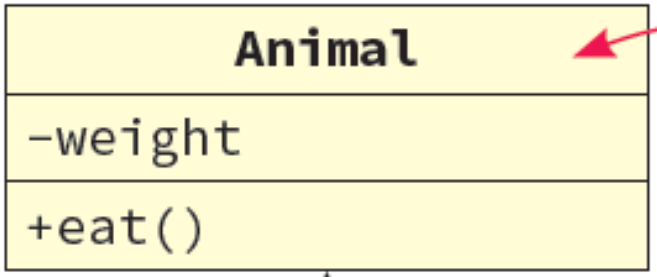


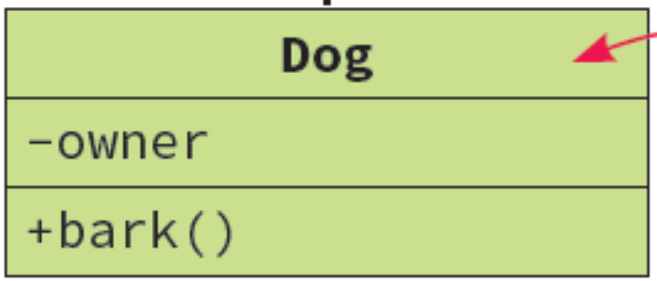
상속의 장점

- 상속의 장점
  - 상속을 통하여 기존 클래스의 필드와 메소드를 재사용
  - 기존 클래스의 일부 변경도 가능
  - 상속을 이용하게 되면 복잡한 GUI 프로그램을 순식간에 작성
  - 상속은 이미 작성된 검증된 소프트웨어를 재사용
  - 신뢰성 있는 소프트웨어를 손쉽게 개발, 유지 보수
  - 코드의 중복을 줄일 수 있다.

상속







부모 클래스 또는
수퍼 클래스라고 한다.

자식 클래스 또는
서브 클래스라고 한다.

부모 클래스와 자식 클래스

- 부모 클래스는 추상적이고 자식 클래스는 구체적이다.

부모 클래스	자식 클래스
Animal(동물)	Lion(사자), Dog(개), Cat(고양이)
Bike(자전거)	MountainBike(산악자전거), RoadBike, TandemBike
Vehicle(탈것)	Car(자동차), Bus(버스), Truck(트럭), Boat(보트), Motorcycle(오토바이)
Student(학생)	GraduateStudent(대학원생), UnderGraduate(학부생)
Employee(직원)	Manager(관리자)
Shape(도형)	Rectangle(사각형), Triangle(삼각형), Circle(원)

상속의 형식

자식 클래스 또는 서브 클래스라고 한다.

형식

```
class Childclass extends Parentclass
{
    // 여기에 필드를 추가한다.
    // 여기에 메소드를 추가한다.
}
```

부모 클래스 또는 수퍼 클래스라고 한다.

상속이 필요한 이유

- 코드를 재사용할 수 있다.
- 중복을 줄일 수 있다.

상속이 중복을 줄이는 이유

Car	Truck	Bus
setSpeed() turn()	setSpeed() turn()	setSpeed() turn()

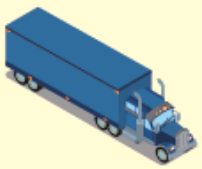
각 클래스에 코드가 중복된다.

Vehicle
setSpeed() turn()

부모 클래스

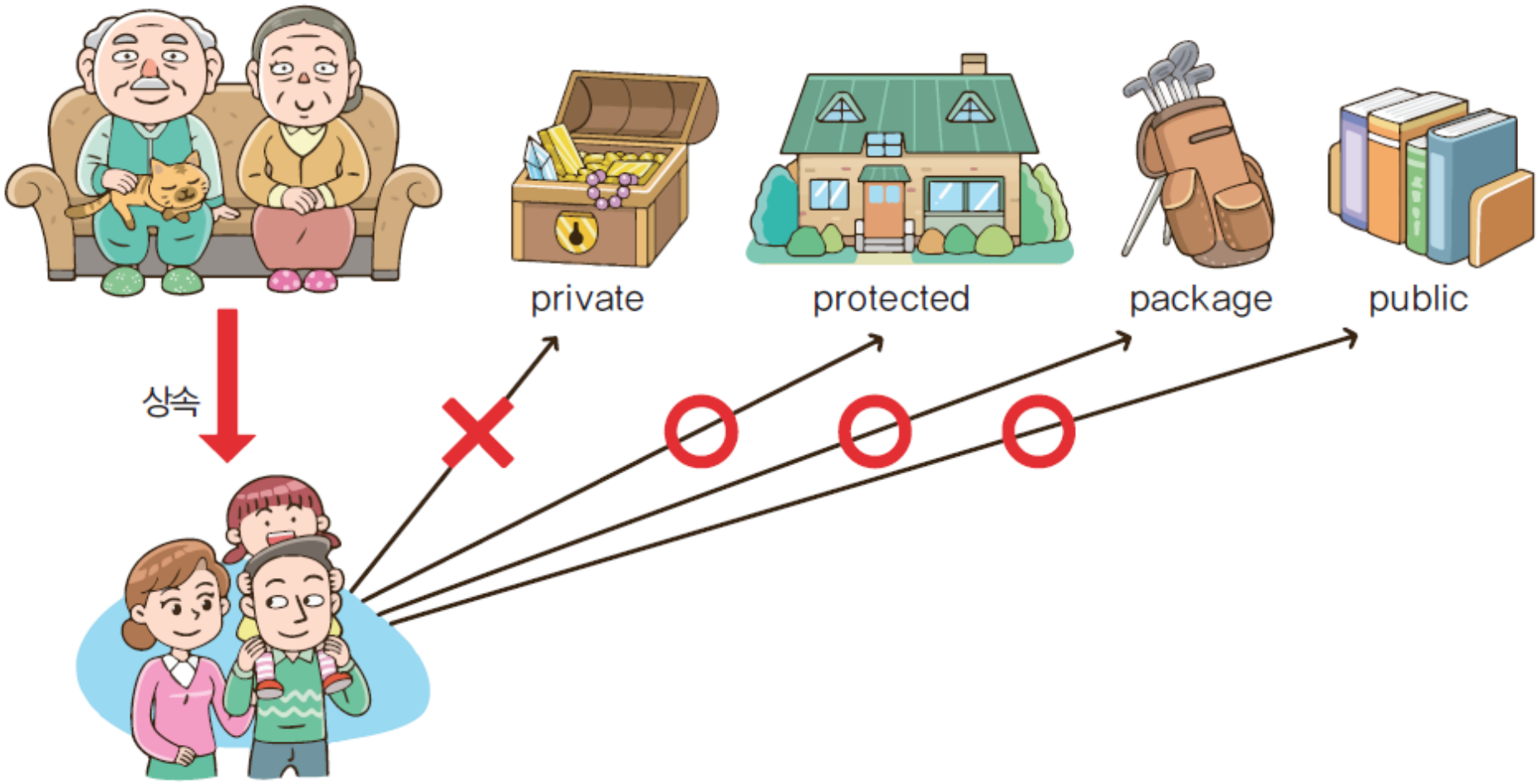
중복되는 코드는 부모 클래스에 모은다.

Car	Truck	Bus



자식 클래스

상속과 접근제어



접근 제어 지정자

접근 지정자	클래스	패키지	자식 클래스	전체 세계
public	O	O	O	O
protected	O	O	O	X
없음	O	O	X	X
private	O	X	X	X

예제

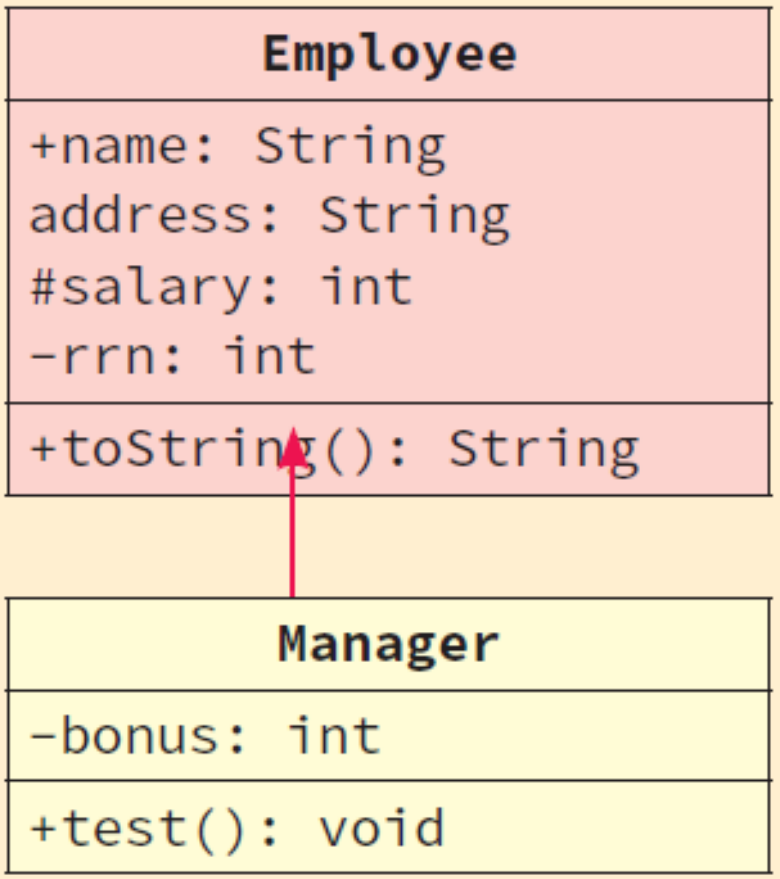
```
public class Shape {
    private int x;
    private int y;
    void print() {
        System.out .println("x좌표: " + x + " y좌표: " + y);
    }
}
```

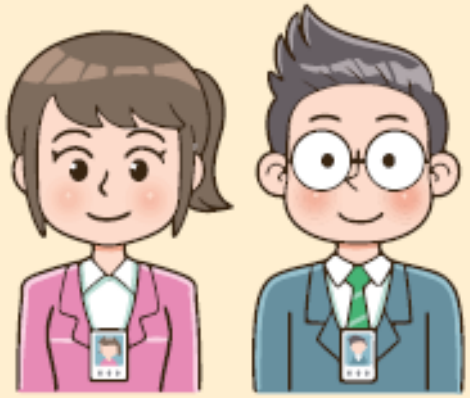
```
public class Rectangle extends Shape {
        private int width;
        private int height;
        double calcArea() {
                return width * height;
        }
        void draw() {
                System.out.println("(" + x + "," + y + ") 위치에 " + "가로: "
                + width + " 세로: " + height);
        }
}
```

부모 클래스의 private 멤버 x와 y는 사용할 수 없다.

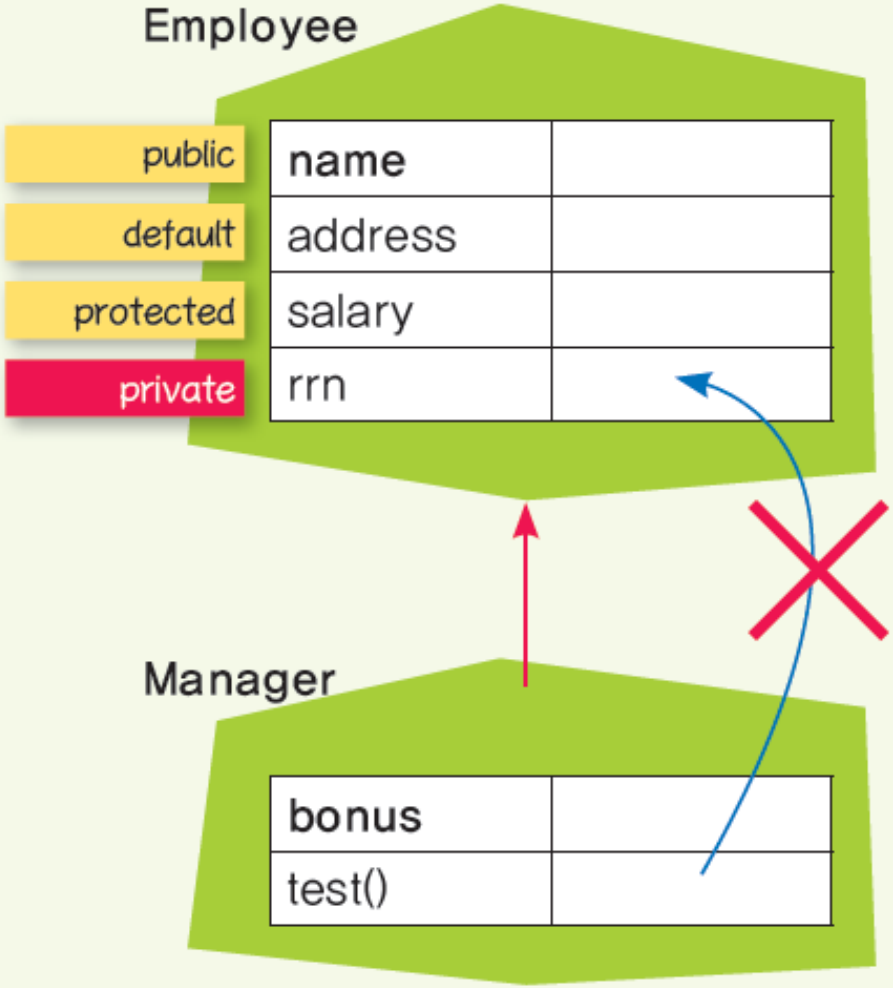
예제: 직원과 매니저 클래스

- 직원(Employee)과 매니저(Manager)의 예



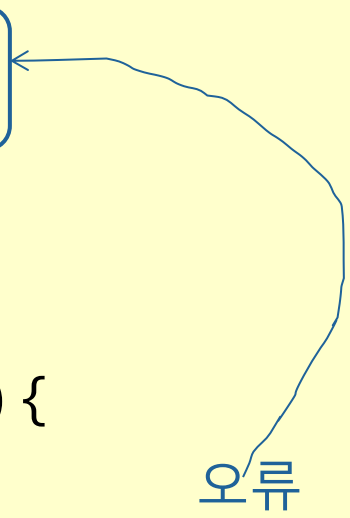


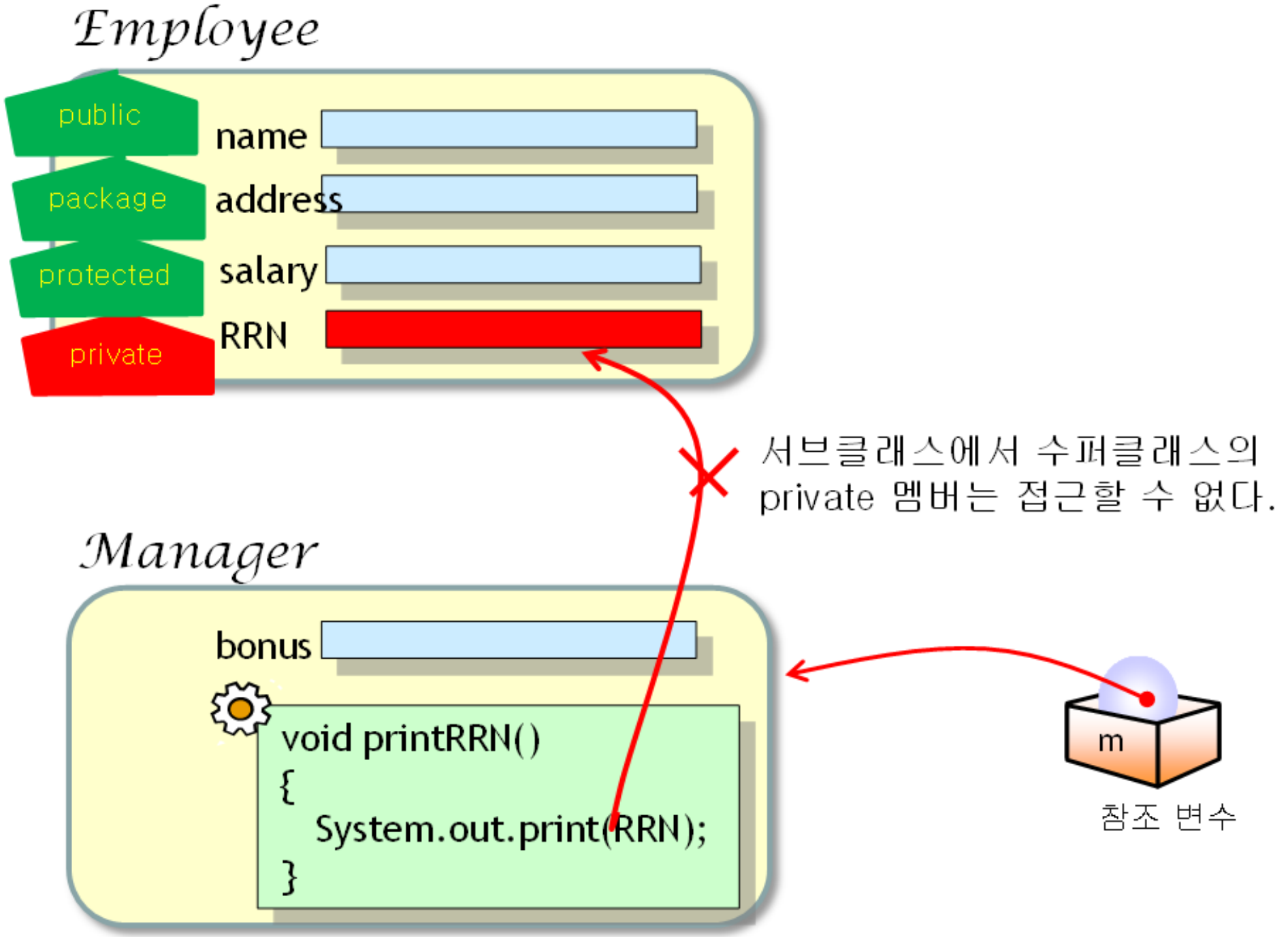
예제: 직원과 매니저 클래스



자식 클래스에서는 부모 클래스의 필드 중에서 private로 선언된 것만 제외하고 모두 접근할 수 있습니다.

```
class Manager extends Employee {
        private int bonus;
        public void printSalary() {
            System.out.println(name + "(" + address + "):" + (salary +
bonus));
        }
        public void printRRN() {
                System.out.println(RRN);
        }
}
public class ManagerTest {
        public static void main(String[] args) {
                Manager m = new Manager();
                m.printRRN();
        }
}
```



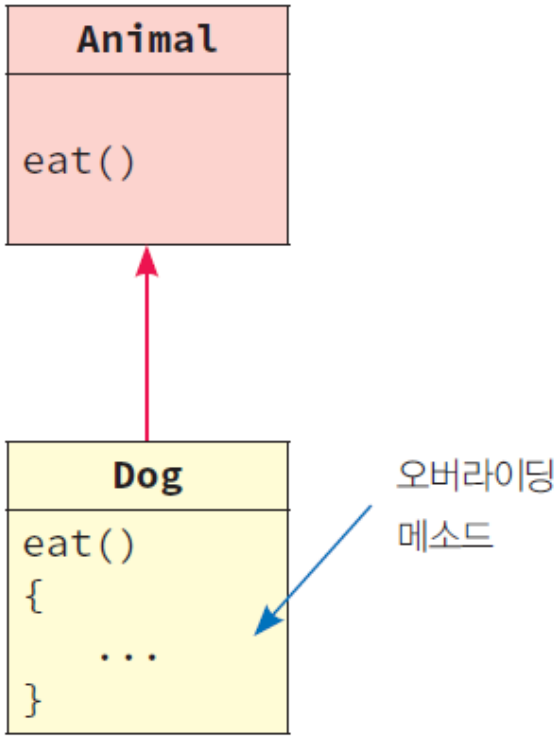

Employee
public
package
protected
private
name
address
salary
RRN
서브클래스에서 수퍼클래스의
private 멤버는 접근할 수 없다.
Manager
bonus
void printRRN()
{
System.out.print(RRN);
}
m
참조 변수

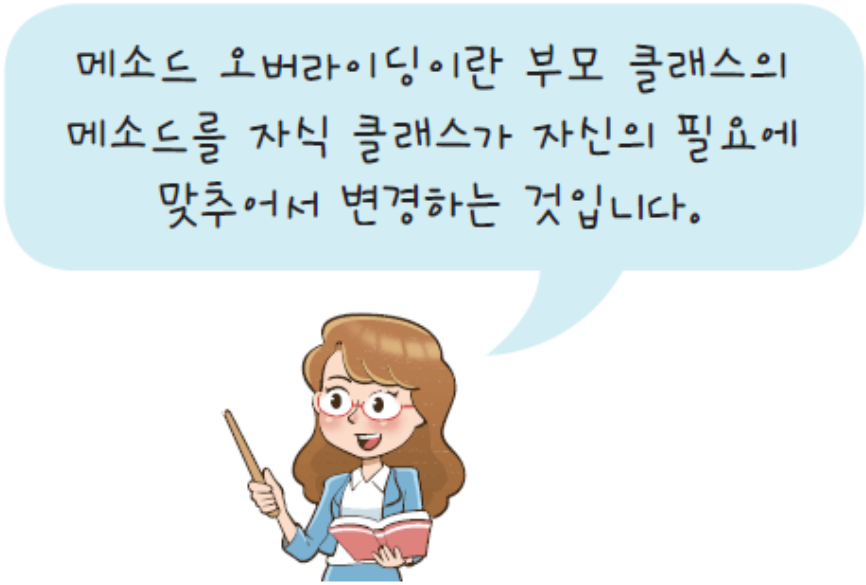
LAB#5-1 패밀리 수입과 지출

- 이번 달 가족의 수입과 지출을 시뮬레이션하자
  - 각자 가족의 수입과 지출 구조를 설명하자
    - 누가 돈을 버는가?
    - 얼마나 많은 가족이 번 돈을 쓰는가?
    - 돈의 수입과 지출에 대한 어떤 규정이나 특성이 있는가?
  - 논의한 학생 중 한 사람의 가족을 선택하자
  - 가족을 위한 클래스와 상속 관계를 만들자
  - 메인 메소드를 활용하여 가족의 수입과 사용에 대한 시뮬레이션 코드를 작성하자

메소드 오버라이딩

- 메소드 오버라이딩(method overriding):
  - 자식 클래스가 필요에 따라 상속된 메소드를 다시 정의하는 것





동물 예제

Animal
weight
picture
eat()
sleep()
부모 클래스
Lion
legs
roar()
Eagle
wings
fly()
자식 클래스

Animal.java

```
01 public class Animal {
02     private double weight;
03     String picture;
04
05     void eat() {       System.out.println("eat()가 호출되었음");    }
06     void sleep() {       System.out.println("sleep()가 호출되었음");    }
07 }
```

Lion.java

```
01 public class Lion extends Animal {
02     private int legs=4;
03     void roar() {       System.out.println("roar()가 호출되었음");    }
04 }
```

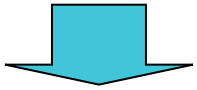
Animal을 상속하여 Lion 클래스를 정의한다.

Eagle.java

```
01 public class Eagle extends Animal {
02     private int wings=2;
03     void fly() {       System.out.println("fly()가 호출되었음");    }
04 }
```

메소드 재정의의 예

```
class Animal {
        public void sound() {
        }
};
```



```
class Dog extends Animal {
        public void sound()     {
                System.out.println("멍멍!");
        }
};
public class DogTest {
        public static void main(String[] args) {
                Dog d = new Dog();
                d.sound();
        }
};
```

Animal

메소드를 오버라이딩하려면

- 메소드의 이름, 반환형, 매개 변수의 개수와 데이터 타입이 일치하여야 한다.

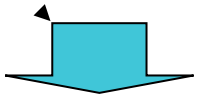
```
public class Animal {
        public void sound()
        {
        }
};
```

오버라이딩이 아님!

```
public class Dog extends Animal {
        public int sound()
        {
        }
};
```

어노테이션

```
public class Animal {
        public void sound()
        {
        }
};
```



오버라이딩이 아님

```
class Dog extends Animal {
        @Override
        void saund() {   // 오류 발생!
                System.out.println("멍멍!");
        }
}
```

LAB#5-2. 동물의 소리

- 동물의 소리에 대해서 메소드 오버라이딩 기법을 적용해 보자
  - 부모 클래스: Animal
  - 자식 클래스: Dog, Cat, Cow, Lion
  - 그들의 메소드: bark()
- 각 동물에 대한 클래스를 작성하자
- Animal 클래스에 대해 bark() 메소드를 만들자
- 각 동물 클래스가 Animal을 상속받도록 하자
- 각 동물 클래스에서 bark() 메소드를 구현하자