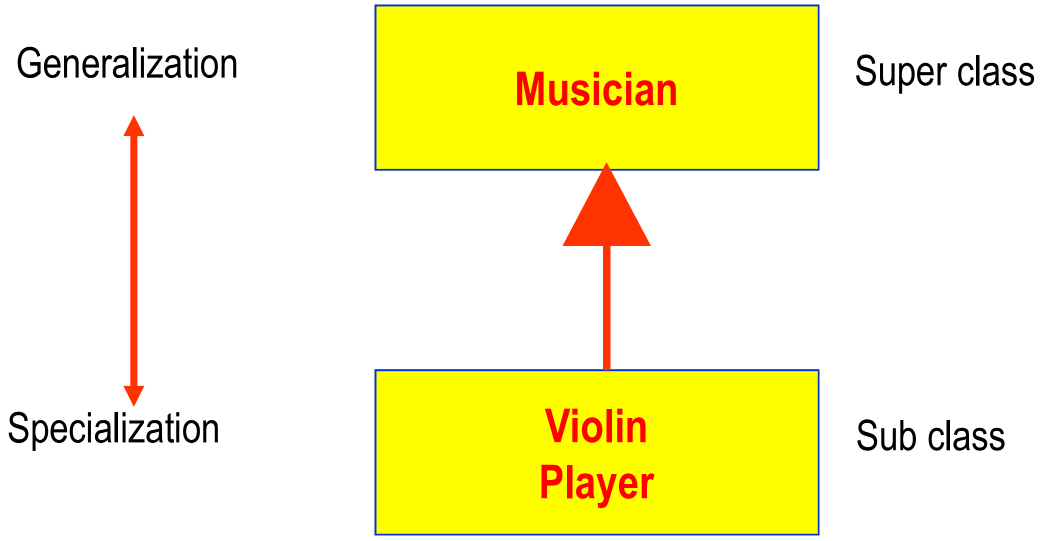
Object Orientation Second Story

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/Core-Java

What is a Inheritance?

- Inheritance specifies an “is a kind of" relationship
  - Inheritance is a class relationship
  - New classes specialize existing classes

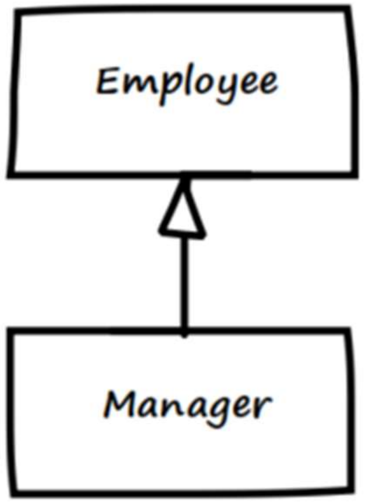


Is this a good example of inheritance ?

What is a Inheritance? (Cont.)

- ✓ 상속이란 이미 존재하는 클래스의 기능을 상속받아 새로운 클래스를 만드는 기법입니다.
- ✓ 상속은 객체지향에서 "is-a"관계에 해당합니다.
- ✓ extends 키워드를 사용하여 기존 클래스를 상속합니다.
- ✓ Java는 다중 상속을 지원하지 않습니다.

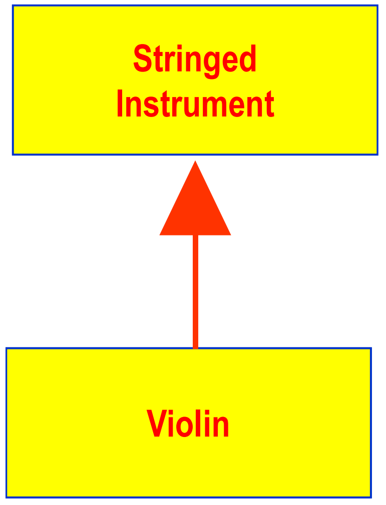
```java
public class Manager extends Employee {

    // 필드 및 메소드
}
```



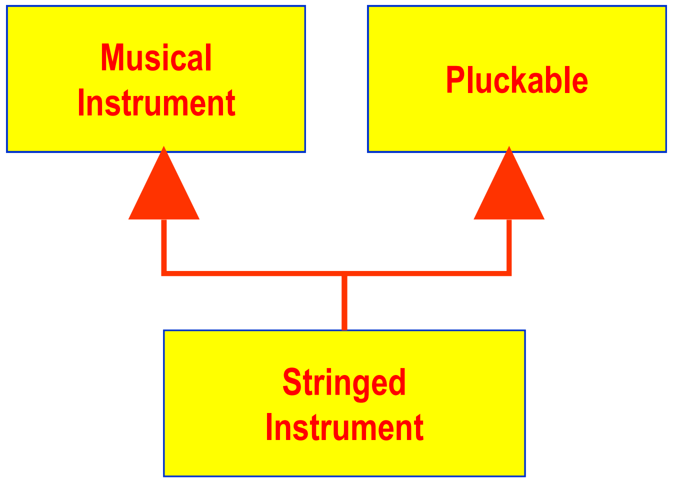
관리자는 직원이다 (O)
직원은 관리자이다 (X)

Single and Multiple Inheritance

- Single inheritance: extending from one super class
- Multiple inheritance: extending from two or more super classes



Violin has a single direct super class

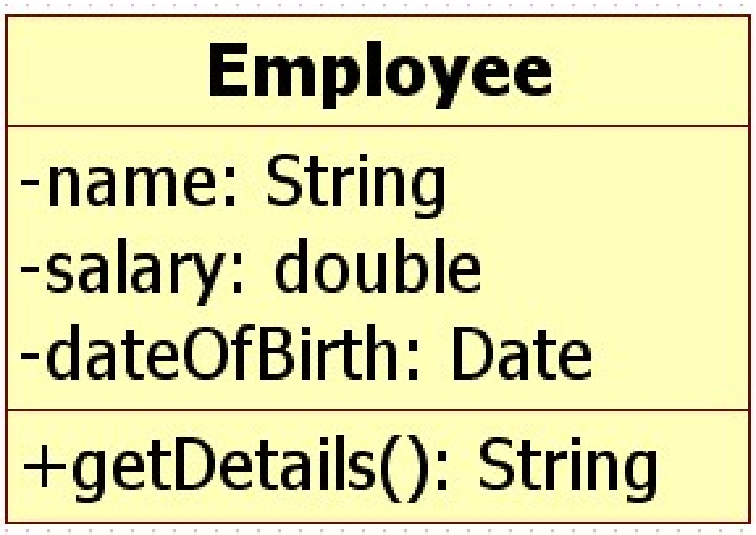


Stringed Instrument has two direct super classes

Subclassing

The **Employee** class:

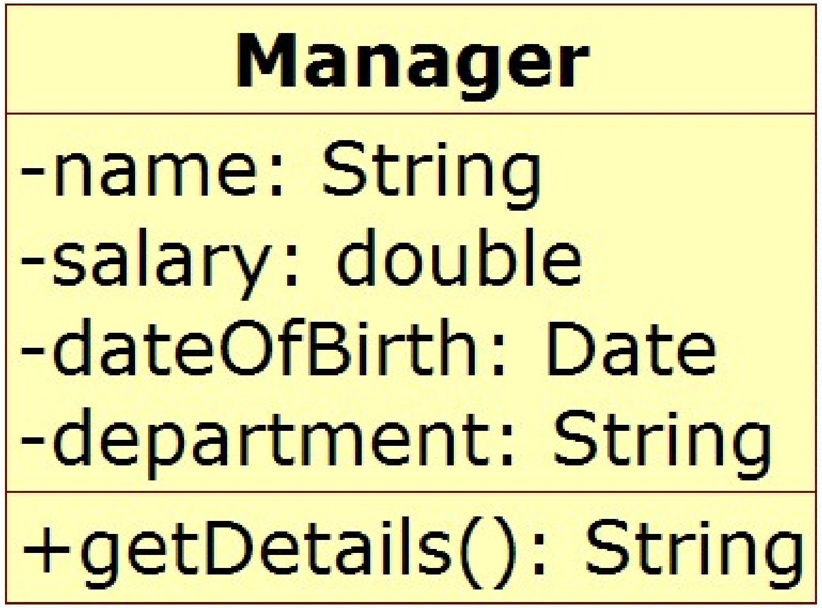
```
public class Employee {
   private String name;
   private double salary;
   private Date dateOfBirth;

   public String getDetails(){ … }
}
```



Subclassing (Cont.)

The **Manager** class:

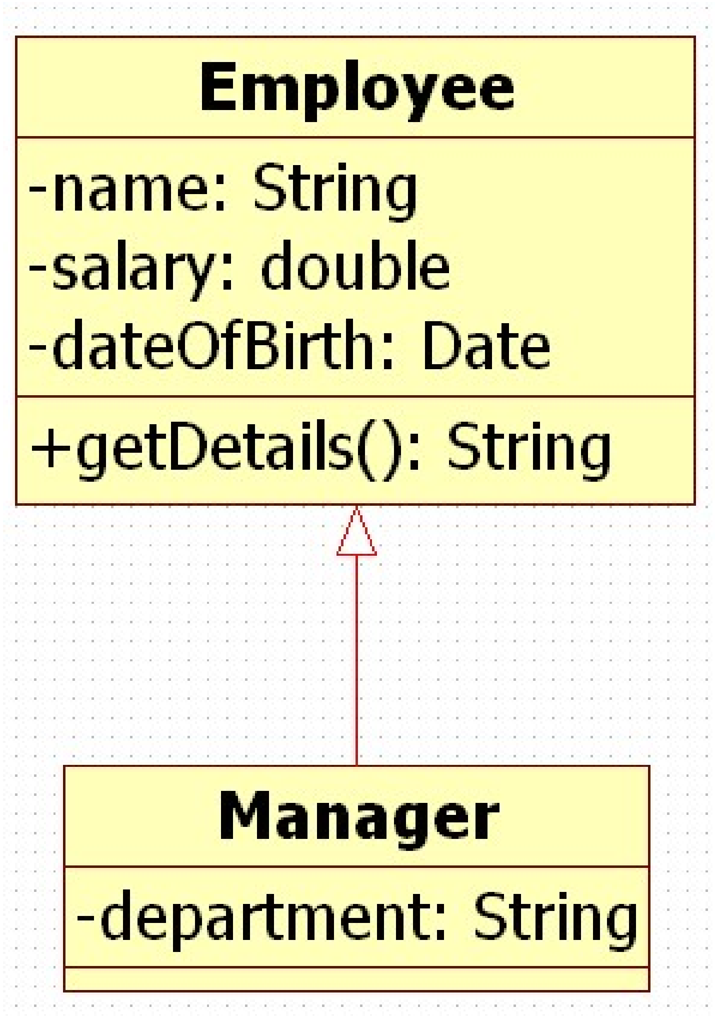
```
public class Manager {
    private String name;
    private double salary;
    private Date dateOfBirth;
    private String department;

    public String getDetails() {…}
}
```



Subclassing (Cont.)

```
public class Employee  {
    public String name;
    public double salary;
    public Date dateOfBirth;
    public String getDetails() { … }
}

public class Manager extends Employee {
    public String department;
}
```

Subclassing (Cont.)



Inheritance

- Inheritance is the OO term referring to grouping classes together based on common theme or common attributes.
- Lets common members be defined in one class and shared by other classes
- Class inherited from superclass or parent class
- Class that inherits subclass or child class
- Use the keyword `extends`.

Inheritance (Cont.)

✓ 자식클래스는 부모클래스를 상속받아서 부모클래스의 모든 자원(속성, 메소드)을 사용할 수 있습니다.

✓ 자식클래스는 부모클래스에 없는 필드와 메소드를 정의하여 기능을 추가할 수 있습니다.

✓ 또한, 상위클래스에 정의된 메소드를 재정의하여 다르게 동작시킬 수 있습니다. (오버라이딩)

```java
public class Employee {

    private String name;
    private double salary;

    public Employee(String name) {
        this.name = name;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public String getName() {
        return name;
    }
}
```

```java
public class Manager extends Employee {

    private double bonus;

    public Manager(String name) {
        super(name);
    }

    public void setBonus(double bonus) {
        this.bonus = bonus;
    }

    public double getSalary() {
        return super.getSalary() + bonus;
    }
}
```

추가적인 필드 정의

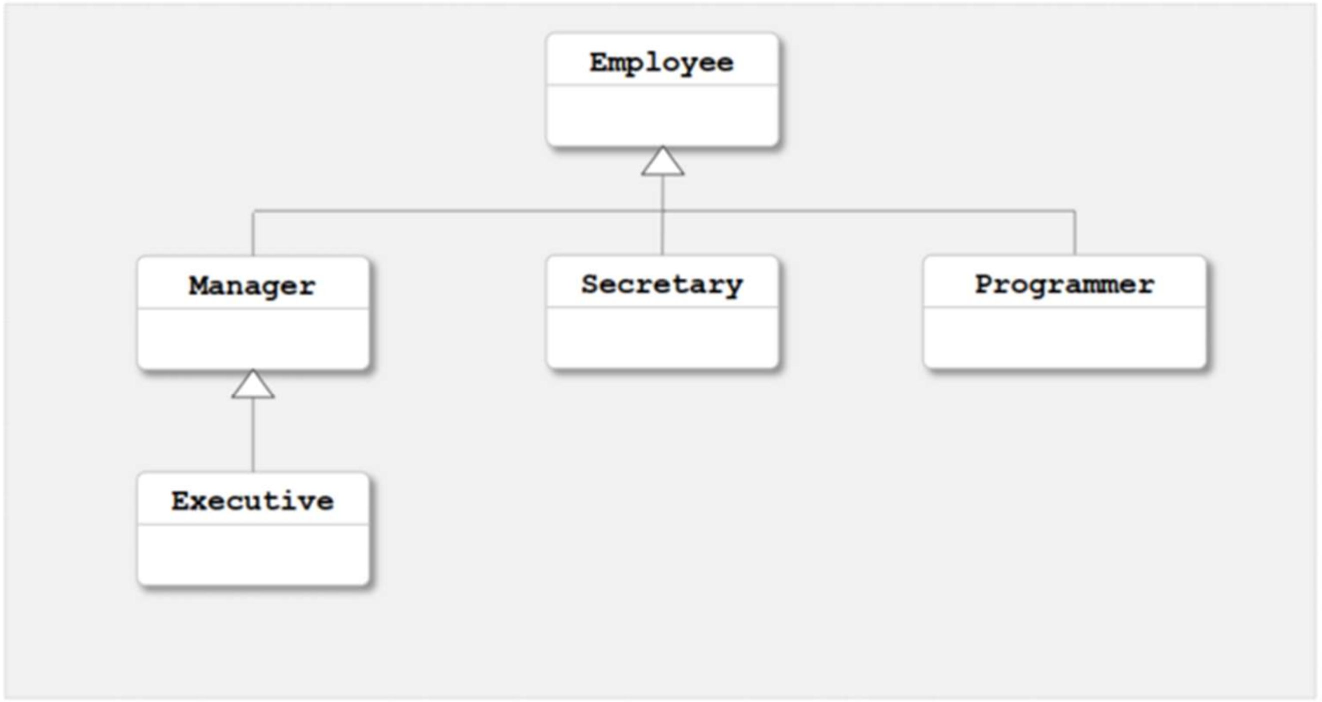
추가적인 메소드 정의

메소드 재정의 (Override)

➔ Manager 객체는 추가적으로 상여금을 가질 수 있으며,
월급 계산 시 상여금이 포함됩니다.

Inheritance (Cont.)

✓ 상속 계층(Inheritance hierarchy) : 부모클래스를 확장하는 모든 집합을 의미합니다.
✓ 상속 체인(Inheritance chain) : 특정 클래스와 상위 클래스간 계층 상의 경로를 말합니다.



Single Inheritance

- When a class inherits from only one class, it is called ***single inheritance***.
- Single inheritance makes code more reliable.
- `interface`s provide the benefits of multiple inheritance without drawbacks.
- Syntax of a Java class:

```
[modifier] class class_name [extends
                                    <superclass> ] {
              …
 }
```

The *is a* Relationship

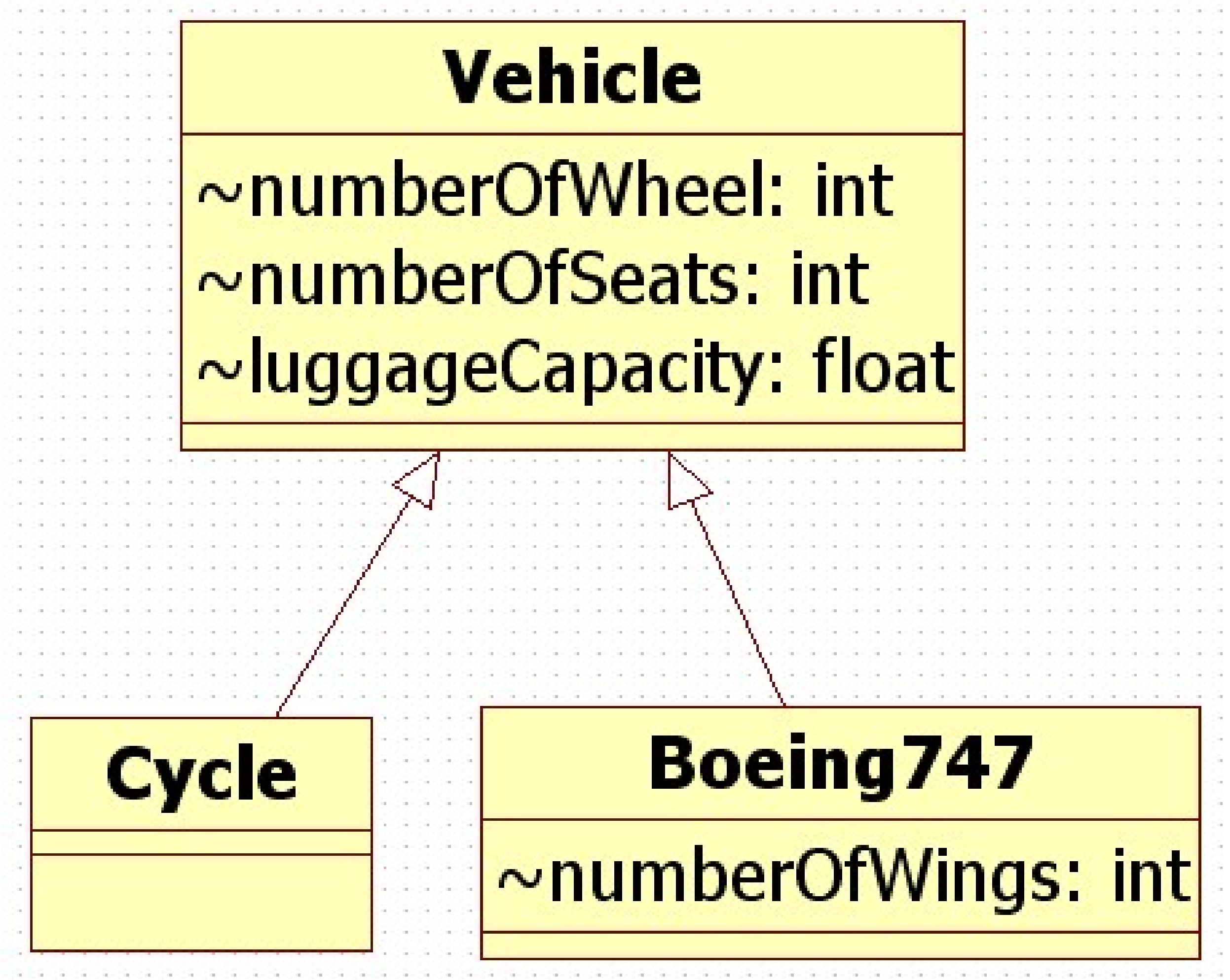
- A class can inherit from only one superclass at a time.
- Use the ***is a*** phrase to determine if a proposed inheritance link is valid.
  - “A Manager object ***is an*** Employee.”

The *is a* Relationship (Cont.)

- Check the *is a* relationship of the following code:

```
class Cycle {
       int numberOfWheels;
       int numberOfSeats;
       float luggageCapacity;
        //and so on
}

class Boeing747 extends Cycle {
       int numberOfWings;
         //and so on
}
```

The *is a* Relationship (Cont.)



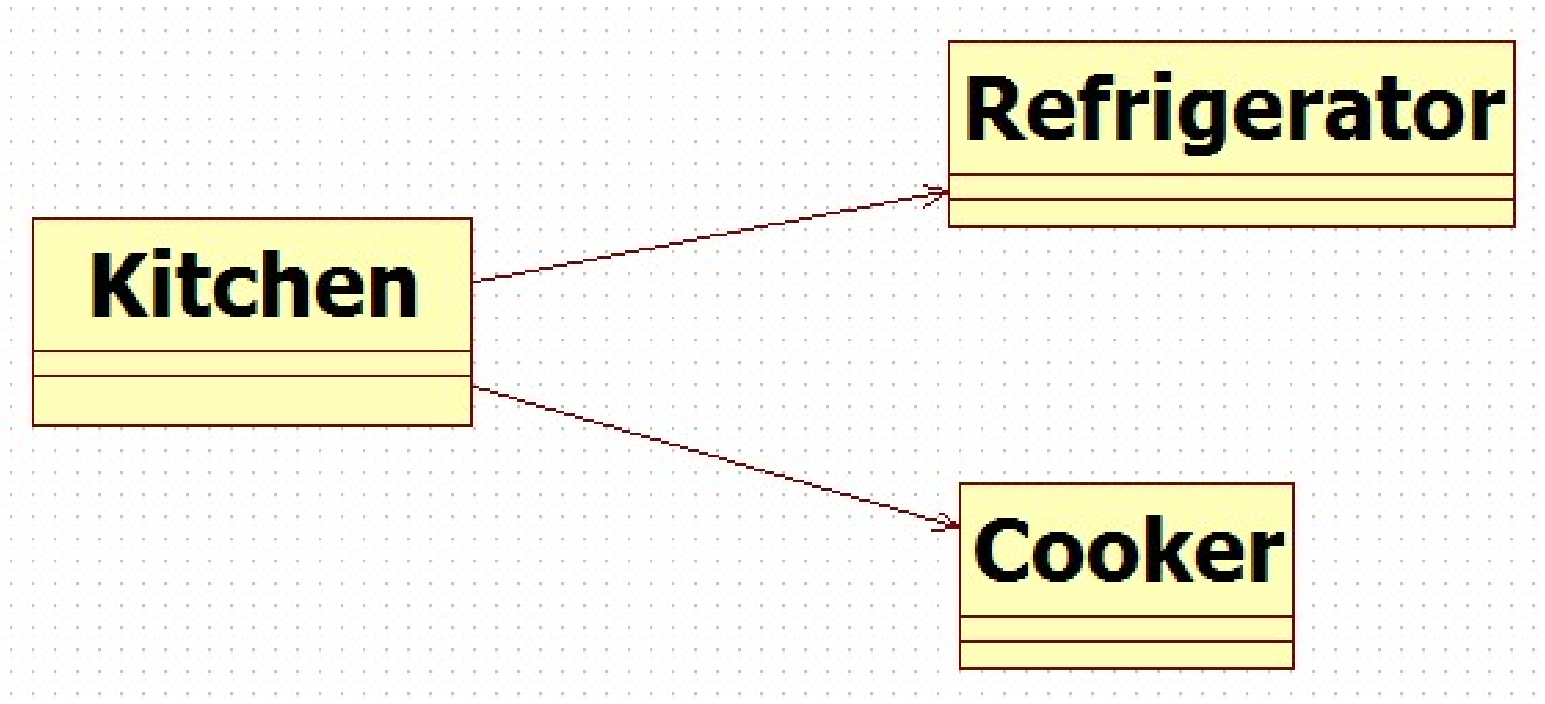
Containment

- Write a class that contains a reference to other classes.
- Objects have to be instantiated separately, but the overall effect is syntactically and realistically improved.

```
class Cooker {
          //whatever the class does
}

class Refrigerator {
          //whatever the class does
}

class Kitchen {
          Cooker myCooker;
          Refrigerator myRefrigerator;
          //and so on
}
```

The *has a* Relationship

- Validate containment relationships with the ***has a*** phrase.
  - “My Kitchen ***has a*** Cooker.”



Constructors Are Not Inherited

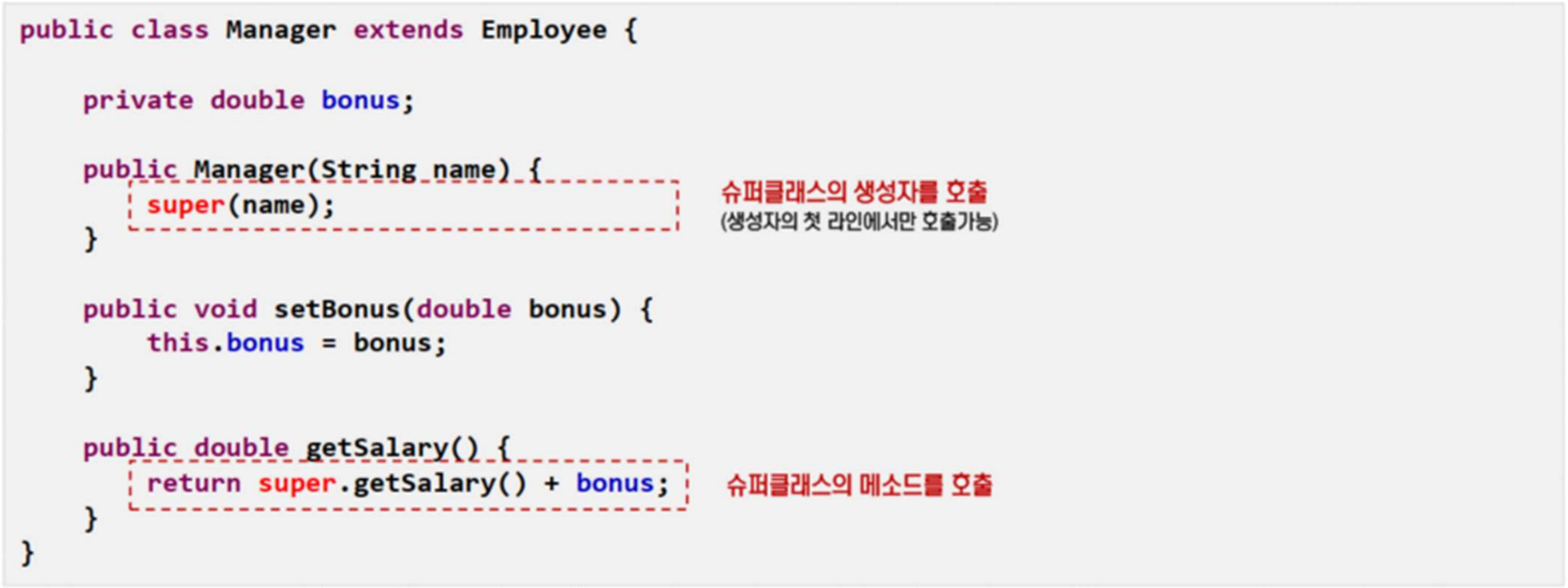
- A subclass inherits all methods and variables from the superclass(parent class).
- A subclass does not inherit the constructor from the superclass.
- Two ways to include a constructor are:
  - Use the default constructor.
  - Write one or more explicit constructors.

The `super` Keyword

- `super` is used in a class to refer to its superclass.
- `super` is used to refer to the member of superclass, both data attributes and methods.
- Behavior invoked does not have to be in the superclass ; it can be further up in the hierarchy.

The `super` Keyword (Cont.)

✓ super는 부모클래스를 의미합니다.
✓ 상속관계의 자식클래스에서 부모클래스의 속성을 참조하거나 메소드를 호출하고자 할 때 super 키워드를 사용합니다.
✓ 생성자 역시 메소드이므로 super 키워드를 사용하여 부모클래스의 생성자를 호출할 수 있습니다.



```java
public class Manager extends Employee {

    private double bonus;

    public Manager(String name) {
        super(name);
    }

    public void setBonus(double bonus) {
        this.bonus = bonus;
    }

    public double getSalary() {
        return super.getSalary() + bonus;
    }
}
```

슈퍼클래스의 생성자를 호출
(생성자의 첫 라인에서만 호출가능)

슈퍼클래스의 메소드를 호출

Invoking Parent Class Constructors

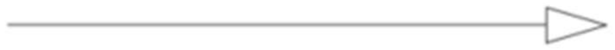
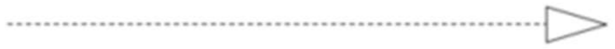
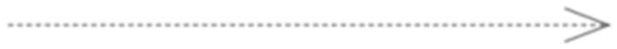
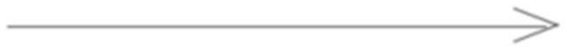
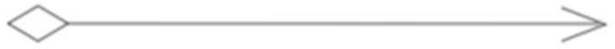
- In many circumstances, the default constructor is used to initialize the parent object.
- If used, you must place **`super`** or **`this`** in the first line of the constructor.

```
public class Employee {
        String name;
        public Employee(String name) {
                this.name = name ;
        }
}
public class Manager extends Employee{
        String department;
        public Manager(String s, String d){
                super(s);
                department = d;
        }
}
```

Class Relations

✓ **가장 일반적인 클래스 간의 관계**

- 의존관계(dependency) : uses-a 관계, 의존대상이 변경될 경우 영향을 받습니다.
- 집합관계(aggregation) : has-a 관계, 클래스가 다른 클래스를 포함하고 있는 관계입니다.
- 상속관계(inheritance) : is-a 관계, 일반적인 클래스와 상세한 클래스 간의 관계입니다.

클래스 간의 관계	UML 표기법
상속 (Inheritance)	
인터페이스 구현 (Interface implementation)	
의존관계 (Dependency)	
연관관계 (Association)	
집합 연관관계 (Aggregated Association)	
복합 연관관계 (Composite Association)