

상속

# 상속

## • 상속

다른 클래스가 가지고 있는 멤버(필드, 메소드)들을 상속받아 재활용하는 것  
부모와 자식간의 관계로 나타냄

## • 장점

1. 코드를 재활용함으로써 코드의 중복을 제거하고, 코드의 양이 줄어듦
2. 공통된 코드를 관리할 수 있기 때문에 코드의 유지보수(추가/변경)에 좋음

# 상속의 종류

- 다중 상속

여러 개의 부모 클래스를 가질 수 있는 상속

- 단일 상속

한가지의 부모 클래스만 가질 수 있는 상속

- 자바에서는 단일 상속만을 지원함

# 다중 상속 - 1

## • 다중 상속의 문제점

### 1. 모호성

→ 상속받은 부모 클래스들끼리 같은 이름을 가진 메소드나 변수가 있을 경우  
어떤 부모 클래스에서 호출 해야하는지 알 수 없는 모호성이 발생

### 2. 복잡성

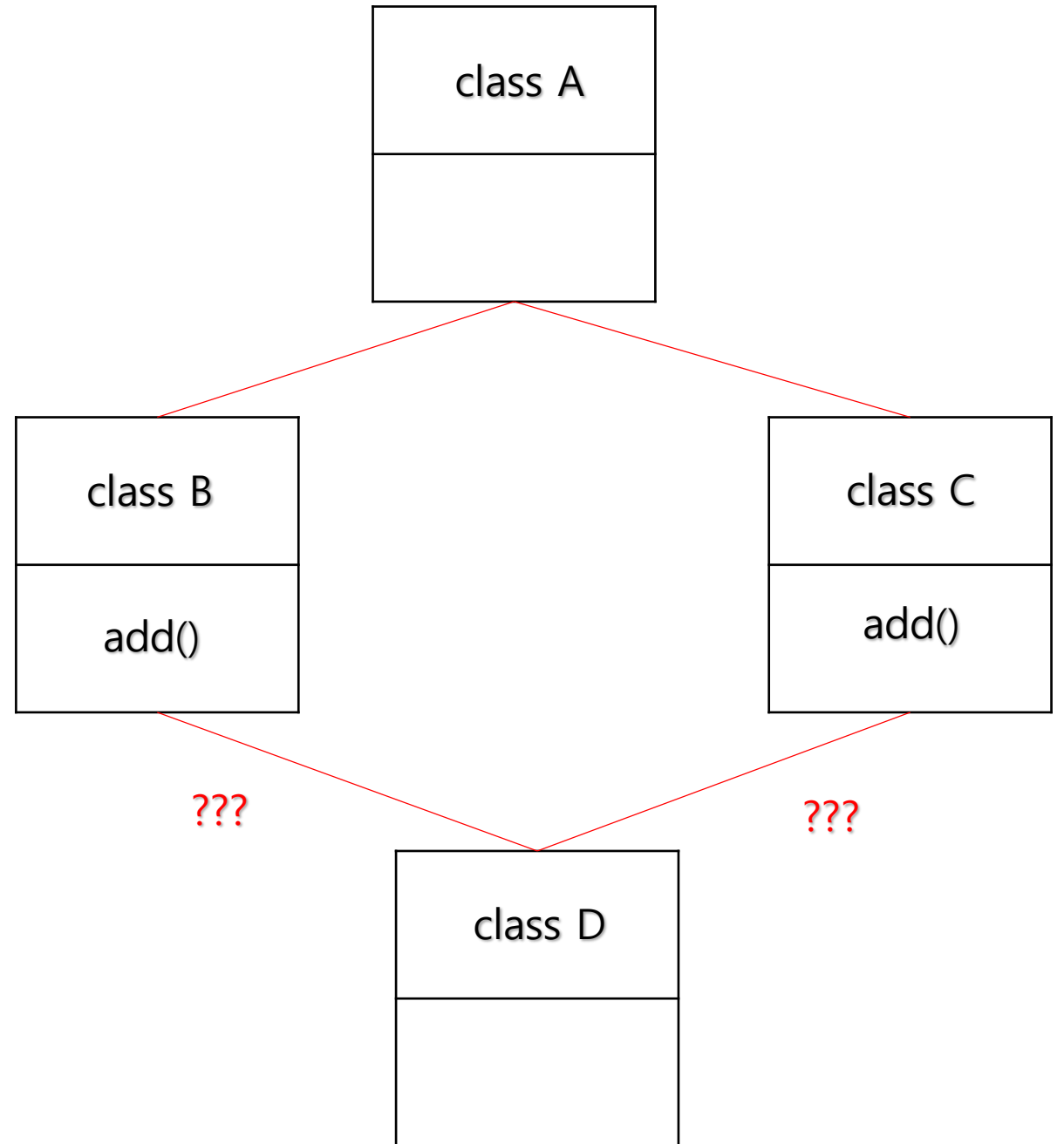
→ 다중 상속을 사용하면 클래스 간의 관계가 복잡해지고 코드의 가독성과 유지보수가  
어려워짐

## 다중 상속 - 2

### • 다이아몬드 상속

다중 상속의 대표적인 예시중 하나로서 클래스간의 관계가 다이아몬드처럼 된다 하여 다이아몬드 상속이라고 함

이러한 다중 상속에는 여러가지 문제(모호성, 복잡성, ...)이 발생하기 때문에 죽음의 다이아몬드(Diamond Of Death)라고도 함

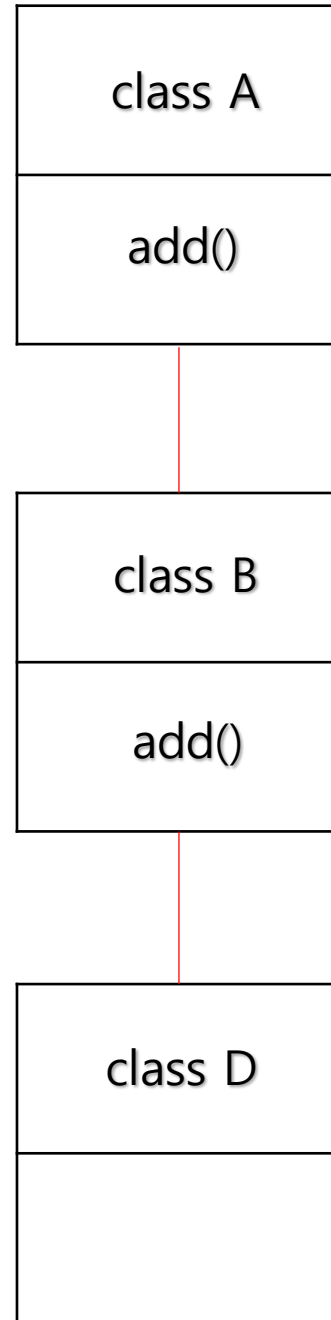


# 단일 상속

- 단일 상속

클래스당 하나의 부모 클래스만 가질 수 있는 상속 형태

자바에서는 단일 상속만을 지원함



# 상속 표현식

- 상속 표현식

class 클래스명 extends 부모클래스명

```
class Parent {  
    public void parentMethod() {  
        System.out.println("Parent's method");  
    }  
}  
  
class Child extends Parent {  
    public void childMethod() {  
        System.out.println("Child's method");  
    }  
}
```

# 상속 관계 - 1

- 상속 관계(is-a)

상속을 통한 관계를 의미 (child는 parent 이다.)

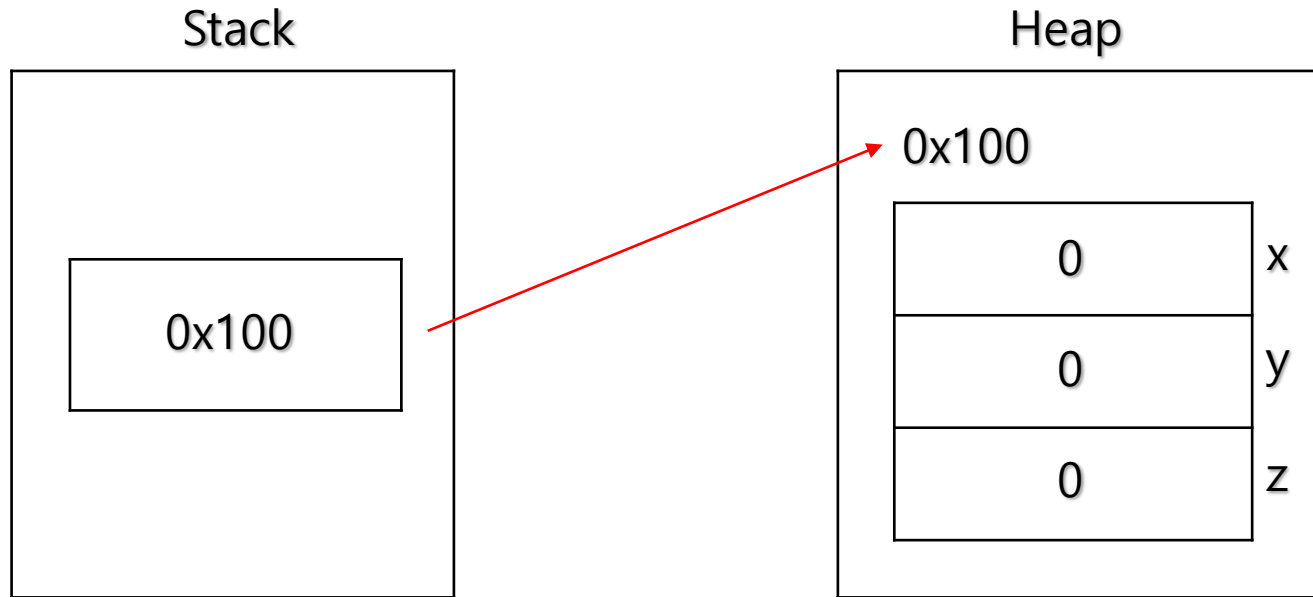
```
class Parent {  
    int x;  
    int y;  
    public void parentMethod() {  
        System.out.println("Parent's method");  
    }  
}  
  
class Child extends Parent {  
    int z;  
    public void childMethod() {  
        System.out.println("Child's method");  
    }  
}
```



## 상속 관계 - 2

- 상속 관계(is-a)

부모의 멤버 변수도 모두 가지고 있으므로 "child는 parent 이다. (is-a)" 라고 표현해도 맞는 표현이 됨



# 포함 관계 - 1

- 포함 관계(has-a)

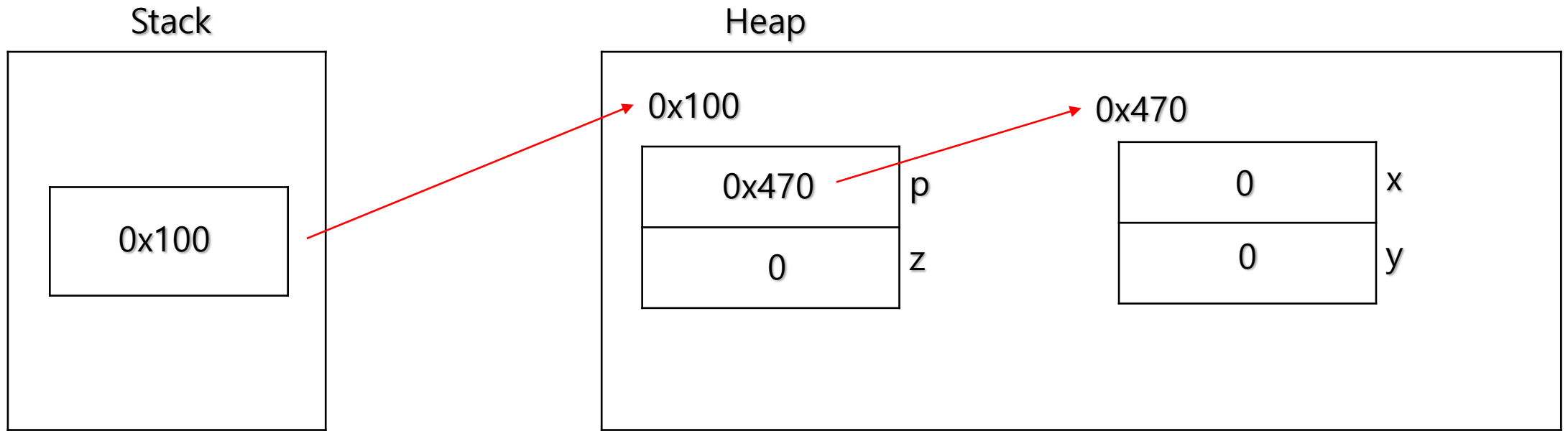
포함을 통한 관계를 의미 (child는 parent를 가지고 있다.)

```
class Parent {  
    int x;  
    int y;  
    public void parentMethod() {  
        System.out.println("Parent's method");  
    }  
}  
  
class Child {  
    Parent p = new Parent();  
    int z;  
    public void childMethod() {  
        System.out.println("Child's method");  
    }  
}
```

## 포함 관계 - 2

- 포함 관계(has-a)

parent에 대한 메모리 주소를 가지고 있으므로 "child는 parent를 가지고 있다.(has-a)" 라고 표현해도 맞는 표현이 됨



참조변수 super

# 참조변수 super

- 참조변수 super

부모 클래스의 멤버를 가르킴

# 참조변수 super

- 설명

부모 클래스의 멤버 변수에 값을 넣고 싶을 때, 부모 클래스의 멤버 변수 이름 앞에 super를 붙여서 사용 가능

```
class Parent {  
    int x;  
    int y;  
    public void parentMethod() {  
        System.out.println("Parent's method");  
    }  
}  
  
class Child extends Parent{  
    int z;  
  
    Child() {  
        super.x = 11;  
    };  
    public void childMethod() {  
        System.out.println("Child's method");  
    }  
}
```

생성자 super()

# 생성자 super() - 1

- 생성자 this()

부모 클래스의 생성자를 호출할 때 사용  
\* 반드시 첫 줄에 선언되어야 함

- 참고

아래와 같이 첫 줄에 생성자가 없을 경우, 컴파일 시 자동으로 super(); 가 삽입됨

```
Parent(int num1, int num2) {  
    x = num1;  
    y = num2;  
}
```

```
Parent(int num1, int num2) {  
    super();  
    x = num1;  
    y = num2;  
}
```

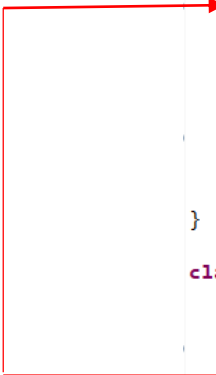


# 생성자 super() - 2

- 설명

부모 클래스의 멤버 변수에 값을 넣고 싶을 때, 부모 클래스의 멤버 변수 이름 앞에 super를 붙여서 사용 가능

```
class Parent {  
    int x;  
    int y;  
  
    Parent(){};  
  
    Parent(int num1, int num2) {  
        x = num1;  
        y = num2;  
    }  
  
    public void parentMethod() {  
        System.out.println("Parent's method");  
    }  
}  
  
class Child extends Parent{  
    int z;  
  
    Child() {  
        super(11, 12);  
    }  
  
    public void childMethod() {  
        System.out.println("Child's method");  
    }  
}
```



# 실습 - 1

## 상속 활용하기 - 1

• Q1. “사람”을 나타내는 클래스와 “학생”을 나타내는 클래스를 작성해보시오.

1. 사람 클래스(Person)을 만들고, 상속받을 학생(Student) 클래스를 만드세요.
2. Person 클래스에는 인스턴스 변수 name과 age가 있습니다.
3. Person 클래스에는 displayInfo() 메소드가 있습니다.  
→ displayInfo() 메소드는 이름과 나이를 출력합니다.
4. Student 클래스에는 인스턴스 변수 StudentId가 있습니다.
5. Student 클래스에는 study()와 studentInfo 메소드가 있습니다.  
→ study() 메소드는 “공부를 시작합니다.” 를 출력합니다.  
→ studentInfo() 메소드는 학번을 출력합니다.

## 상속 활용하기 - 2

### • Q2. 동물의 정보를 다루는 프로그램을 상속 관계를 활용해 작성하시오.

1. 동물 클래스(Animal)을 만들고, 상속받을 강아지(Dog)와 고양이(Cat) 클래스를 만드세요.
2. Animal 클래스에는 인스턴스 변수 name과 age가 있습니다.
3. Animal 클래스에는 매개변수 있는 생성자(String name, int age)가 있습니다.
4. Animal 클래스에는 eat 메소드와 sleep 메소드가 있습니다.
  - eat 메소드는 "강아지(3) 이(가) 먹는 중입니다."와 같이 이름과 나이를 출력합니다.
  - sleep 메소드는 "고양이(1) 이(가) 잠을 자고 있습니다."와 같이 이름과 나이를 출력합니다.
5. Dog 클래스에는 매개변수 있는 생성자(String name, int age)가 있습니다.
  - 이 생성자에서 부모 클래스 Animal의 멤버 변수에 값을 넣으세요.
6. Dog 클래스에는 bark() 메소드가 있으며 "멍멍!"을 출력합니다.
7. Cat 클래스에는 매개변수 있는 생성자(String name, int age)가 있습니다.
8. Cat 클래스에는 meow() 메소드가 있으며 "야옹~"을 출력합니다.

오버라이딩

# 오버라이딩 - 1

## • 오버라이딩(Overriding)

상속 받은 부모 클래스의 메소드를 재작성 하는 행위

이미 작성되어 있는 부모 클래스의 메소드를 일부 수정하여 사용할 수 있으며  
자식 클래스의 객체를 통해 실행하면 후손일수록 우선권을 가지게 됨

## • 조건

1. 선언부 (반환타입, 메서드이름, 매개변수 타입 및 위치)가 부모 클래스의 메소드와 일치해야함
2. 접근 제어자를 부모 클래스의 메소드보다 좁은 범위로 변경 불가
3. 부모 클래스의 메소드보다 예외 처리를 많이 선언할 수 없음
4. private 접근제어자는 오버라이딩 불가

## 오버라이딩 - 2

### • 오버라이딩 표현 시 주의 사항

오버라이딩을 하게 될 때 일반적으로 @Override 어노테이션을 해주어야 함  
해주지 않아도 에러는 발생하지 않지만, 아래의 이유로 어노테이션을 작성해줌

1. 컴파일 오류 방지  
→ 부모 클래스에 해당 메소드가 존재하는지 컴파일 시 확인을 해주고 오타가  
발생하거나 메소드가 없을 경우 오류를 감지해줌
2. 가독성 향상  
→ @Override를 작성함으로써 개발자가 작업을 할 때 이 메소드는 오버라이딩을 한  
메소드라는것을 쉽게 알 수 있게 해줌
3. 유지보수 용이성  
→ 부모 클래스에 정의된 메소드 시그니처(이름, 매개변수, 반환타입 등)가 변경되는  
경우 오류를 감지하여 제대로 오버라이딩이 된 상태인지 미리 파악할 수 있음

# 오버라이딩 - 3

- 오버라이딩 표현 시 주의 사항

아래와 같이 @override라는 어노테이션을 사용하면 에러가 발생할 경우 알려줌

```
@Override  
public void makeSounda() {  
    System.out.println("멍멍멍멍멍");  
}
```




# 오버라이딩 - 4

## • 오버라이딩 예시

부모 클래스에 makeSound() 메소드가 있고, 자식 클래스에도 동일하게 있으나 makeSound() 메소드를 재정의 함으로써 다른 기능으로 동작하게 됨

```
//동물 클래스
class Animal {
    public void makeSound() {
        System.out.println("동물이 소리를 내고 있습니다.");
    }
}

//개 클래스 (동물 클래스를 상속)
class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("멍멍멍멍");
    }
}
```



# 오버라이딩 - 5

## • 오버라이딩 vs 오버로딩

오버라이딩

→ 부모 클래스에서 상속받은 메소드를 재정의 하는 행위

오버로딩

→ 같은 클래스 내에서 메소드의 매개변수 타입/위치를 종류별로 만드는 행위

# 실습 - 1

## 오버라이딩 활용하기 - 1

- Q1. "사람"을 나타내는 클래스와 "학생"을 나타내는 클래스를 작성해보시오.

1. 실습-1(슬라이드19p)에서 진행했던 Q1 문제에서 아래의 오버라이딩을 구현하세요.
2. `displayInfo()` 메소드를 추가하고 이름, 나이, 학번을 모두 출력하세요.

## 오버라이딩 활용하기 - 2

- Q2. 동물의 정보를 다루는 프로그램을 상속 관계를 활용해 작성하시오.

1. 실습-2(슬라이드20p)에서 진행했던 Q2 문제에서 아래의 오버라이딩을 구현하세요.
2. 자식 클래스의 bark(), meow() 메소드를 주석처리 하세요.
3. makeSound() 메소드를 생성하고 아래와 같이 출력하세요.
  - 부모 클래스 : "동물들이 소리를 냅니다."
  - "고양이(1)이가 야옹~ 하고 소리를 냅니다." 이름(나이)
  - "강아지(3)이가 멍멍! 하고 소리를 냅니다." 이름(나이)