

# C++언어 9강

강사 | 최정호

# INDEX

1. 상속
2. 다중상속
3. 가상함수

상속

## 상속

- 다른 class의 멤버 변수, 멤버 함수를 자신의 멤버인 것 처럼 사용이 가능하다.
- Class의 **재사용**과 관련이 있다.

부모 class(Parent Class)

사람

학생

사람

학년

학번

성적

자식 class(Child Class)

직장인

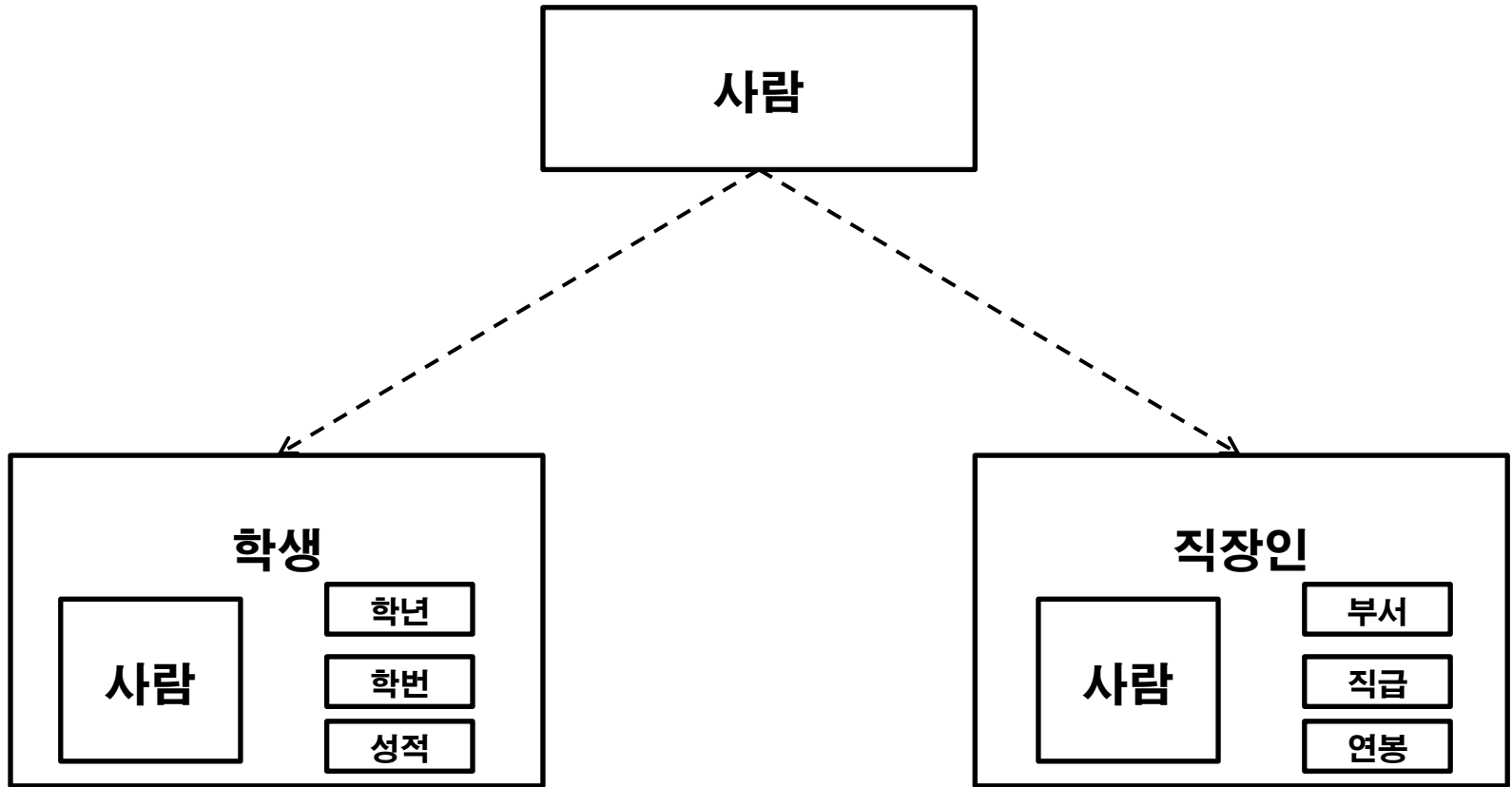
사람

부서

직급

연봉

자식 class(Child Class)



# 상속

```
#include <iostream>
using namespace std;
```

```
class A
{
```

```
public:
```

```
    A()
    {
```

```
        cout << "A Class 생성" << endl;
```

```
    }
```

```
};
```

```
class B : public A
{
```

```
    public:
    B()
    {
```

```
        cout << "B Class 생성" << endl;
```

```
    }
```

```
};
```

```
void main()
{
```

```
    B b;
```

```
}
```

**B class**

**A class**

# 상속

```
#include <iostream>
using namespace std;
```

```
class A
{
public:
```

```
    A()
    {
```

```
        cout << "A의 생성자 호출" << endl;
```

```
    }
```

```
    ~A()
    {
```

```
        cout << "A의 소멸자 호출" << endl;
```

```
    }
```

```
};
```

```
class B : public A
```

```
{
```

```
public:
```

```
    B()
    {
```

```
        cout << "B의 생성자 호출" << endl;
```

```
    }
```

```
    ~B()
    {
```

```
        cout << "B의 소멸자 호출" << endl;
```

```
    }
```

```
};
```

```
void main()
```

```
{
```

```
    B b;
```

```
}
```

**B class**

**A class**

## 상속의 특징

1. is ~ a( ~ 은 ~이다.) 의 관계  
Person Class / Student Class

Ex.

사람 , 학생 Class

사람은 학생이다 ( X )

학생은 사람이다 ( O )





## 상속의 특징

1. has ~ a( ~ 은 ~ 을 가지고 있다.) 의 관계  
Person Class / Phone Class

Ex.

사람 , Phone Class

핸드폰은 사람을 가지고있다 ( X )

사람은 핸드폰을 가지고있다 ( O )



# 상속

```
#include <iostream>
using namespace std;
```

```
class A
{
private:
    int m_ia;
protected:
    int m_ib;
public:
    int m_ic;
    void Test()
    {
        m_ia = 10;
        m_ib = 10;
        m_ic = 10;
    }
};

class B : public A
{
    void Test()
    {
        m_ia = 10;
        m_ib = 10;
        m_ic = 10;
    }
};
```

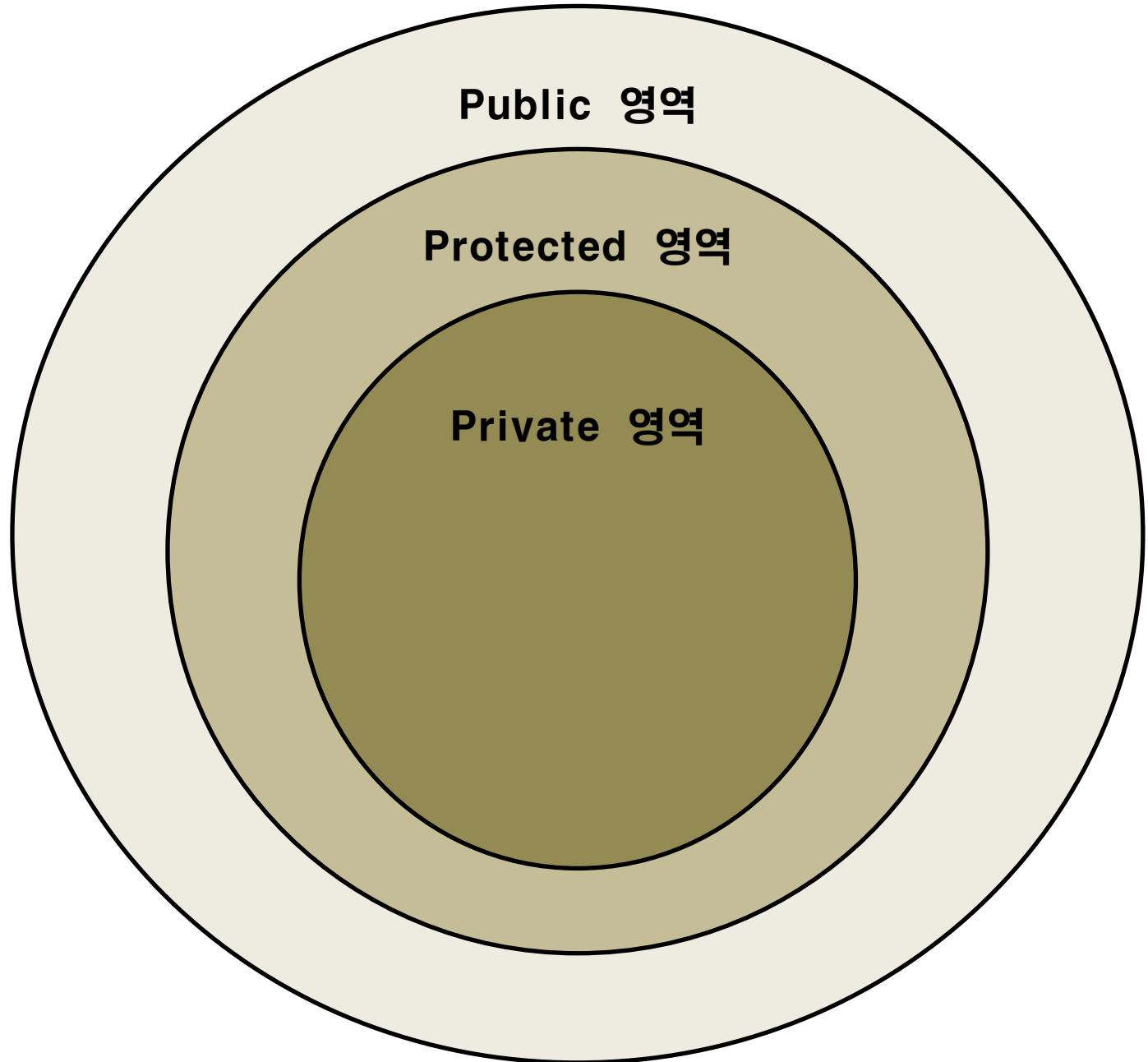
```
class C : public B
{
    void Test()
    {
        m_ia = 10;
        m_ib = 10;
        m_ic = 10;
    }
};

void main()
{
    B b;
    b.m_ia = 10;
    b.m_ib = 10;
    b.m_ic = 10;
}
```

01

02

03



## Overriding(오버라이딩)

- 부모class의 함수의 처리동작이 마음에 들지 않을 경우 **동일한 이름**으로 자식class가 만들어 **재정의** 한다.

# 상속

```
#include <iostream>
using namespace std;

class Mammal
{
public:
    void speak(int cnt)
    {
        cout << cnt << "번 짫다" << endl;
    }
    void speak()
    {
        cout << "짫다" << endl;
    }
};

class Dog : public Mammal
{
public:
    void speak()
    {
        cout << "멍멍" << endl;
    }
};

void main()
{
    Mammal dongmul;
    Dog jindo;
    dongmul.speak();
    dongmul.speak(3);
    jindo.speak();
    //jindo.speak(5);
}
```

## UpCasting(업캐스팅)

- 자식Class객체의 **주소값**을 **부모Class 포인터 변수**에 담아 사용한다.
- 여러 자식Class의 **부모가 동일** 할 경우 해당 부모 Class에 여러 자식Class를 **모아** 일괄 처리한다. (ex. 부모 : 동물 자식 : 고양이, 강아지, 원숭이)

# 상속

```
#include <iostream>
using namespace std;

class Mammal
{
public:
    void speak(int cnt)
    {
        cout << cnt << "번 짫다" << endl;
    }
    void speak()
    {
        cout << " 짫다" << endl;
    }
};

class Dog : public Mammal
{
public:
    void speak()
    {
        cout << "멍멍" << endl;
    }
};

void main()
{
    Mammal* ptr;
    Dog jindo;
    ptr = &jindo;
    ptr->speak();
    ptr->speak(5);
}
```

## Quiz

- **Login Class를 만들어 Computer Class 와 함께 상속관계를 만들어 사용하시오.**



(실행파일)



**다중상속**

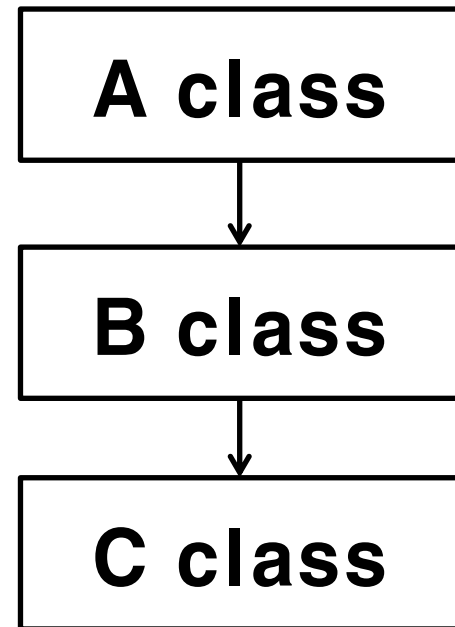
## 다중상속

- 부모 class가 **둘 이상**으로 상속 받은 경우
- 기본적으로 **C++에서만 지원**하며 다른 언어에서는 지양하는 편

## » 상속의 형태

- 해당 자식 Class가 다른 Class의 부모가 되는 경우

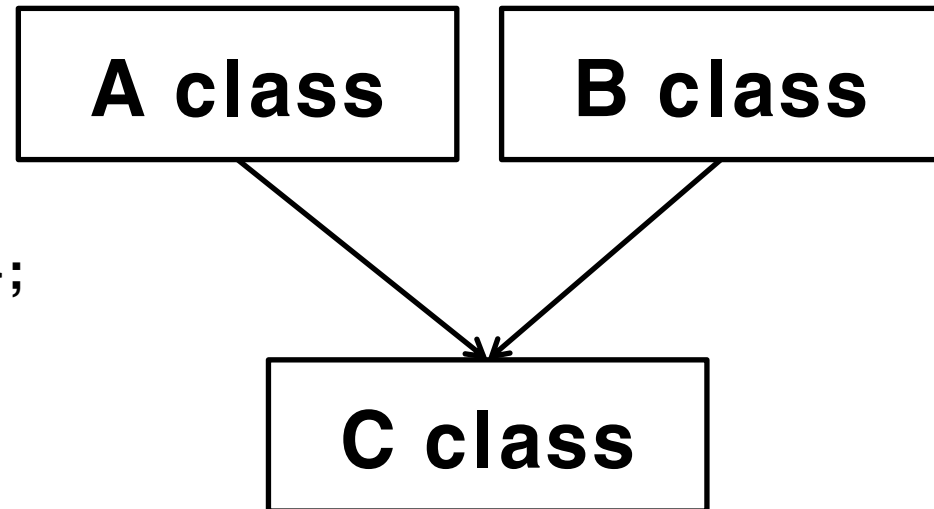
```
class A : {};  
class B : public A{};  
class C : public B{};
```



## » 상속의 형태

- 해당 class 의 부모가 2개 이상 인 경우

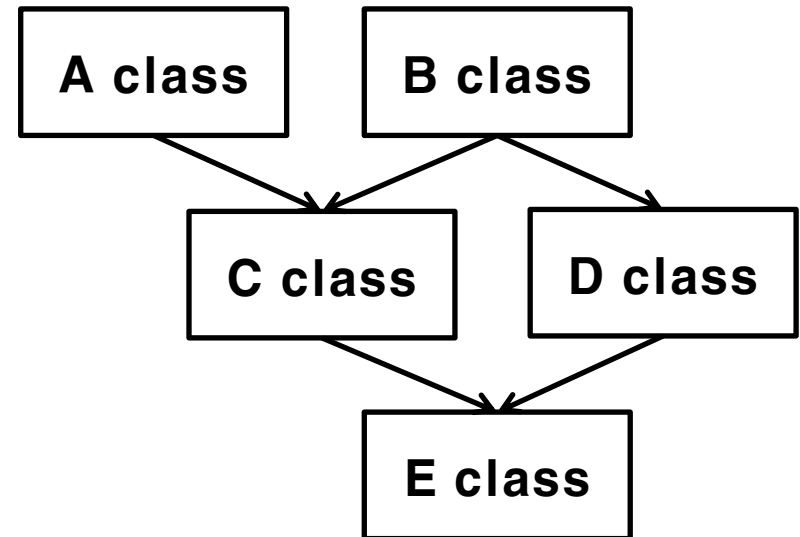
```
class A : {};  
class B : {};  
class C : public A,public B {};
```



## » 상속의 형태

- 위의 두 가지가 혼합된 형태

```
class A : {};  
class B : {};  
class C : public A,public B {};  
class D : public B{};  
class E : public C,public D {};
```



```
#include <iostream>
using namespace std;

class A
{
public:
    void func1()
    {
        cout << "A함수 입니다." << endl;
    }
};

class B
{
public:
    void func2()
    {
        cout << "B함수 입니다." << endl;
    }
};

class C : public A, public B
{
public:
    void func3()
    {
        func1();
        func2();
    }
};

void main()
{
    C c;
    c.func3();
}
```

```
#include <iostream>
using namespace std;

class A
{
public:
    void func()
    {
        cout << "A함수 입니다." << endl;
    }
};

class B
{
public:
    void func()
    {
        cout << "B함수 입니다." << endl;
    }
};

class C : public A, public B
{
public:
    void func3()
    {
        func();
        func();
    } //에러
};

void main()
{
    C c;
    c.func3();
}
```

```
#include <iostream>
using namespace std;
```

```
class A
{
public:
    void func()
    {
        cout << "A함수 입니다." << endl;
    }
};

class B
{
public:
    void func()
    {
        cout << "B함수 입니다." << endl;
    }
};

class C : public A, public B
{
public:
    void func3()
    {
        A::func();
        B::func();
    } //에러
};

void main()
{
    C c;
    c.func3();
}
```



# 다중상속

```
#include <iostream>
using namespace std;

class A
{
public:
    A()
    {
        cout << "A함수 생성자." << endl;
    }
};

class B : public A
{
public:
    B()
    {
        cout << "B함수 생성자." << endl;
    }
};

class C : public A
{
public:
    C()
    {
        cout << "C함수 생성자." << endl;
    }
};

class D : public B, public C
{
public:
    D()
    {
        cout << "D함수 생성자." << endl;
    }
};

void main()
{
    D d;
}
```

## Quiz

- **성별,나이,이름을 저장하고 출력하는 Person Class 와 학년,반,번호 를 저장하고 출력하는 School Class를 만들고 위의 두Class를 상속받은 Student Class를 만들어 출력하시오.**  
**(단 Person Class와 School Class의 멤버함수는 Student Class에서만 호출가능 )**



(실행파일)

# 가상함수

## 가상함수

- 자식의 주소를 부모 포인터에 **업캐스팅** 했을 시 **오버라이딩**된 함수를 사용하게 해주는 방법
- 부모class 에서 작성한다.
- 소멸자 함수도 virtual 을 써주어야 한다.
- 형식 : virtual 반환자료형 함수이름(매개변수)

01

```
#include <iostream>
using namespace std;
```

02

```
class Bumo
{
```

03

```
public:
```

```
    void func()
```

```
    {
```

```
        cout << "Bumo함수 입니다." << endl;
```

```
    }
```

```
};
```

```
class Jasic : public Bumo
```

```
{
```

```
public:
```

```
    void func()
```

```
    {
```

```
        cout << "Jasic함수 입니다." << endl;
```

```
    }
```

```
};
```

```
void main()
```

```
{
```

```
    Jasic ob;
```

```
    Bumo* mom = &ob;
```

```
    mom->func();
```

```
}
```

01

```
#include <iostream>
using namespace std;
```

02

```
class Bumo
{
```

03

```
public:
    virtual void func()
    {
        cout << "Bumo함수 입니다." << endl;
    }
};
class Jasic : public Bumo
{
public:
    void func()
    {
        cout << "Jasic함수 입니다." << endl;
    }
};
void main()
{
    Jasic ob;
    Bumo* mom = &ob;
    mom->func();
}
```

# •가상함수예제 참고

## 순수가상함수

- 부모 class에서는 **사용하지 않는 함수** 이지만 자식 class는 무조건 동일한 이름으로 함수를 **오버라이딩**을 적용하게끔 **강제**하는 방법
- 형식 : `virtual 반환자료형 함수이름(매개변수) = 0;`



# •순수가상함수예제 참고

## 가상상속

- 다중 상속으로 인한 부모class를 **여러 번 생성** 못하게끔 제어하는 방법

# 가상함수

```
#include <iostream>
using namespace std;

class A
{
public:
    A()
    {
        cout << "A함수 생성자." << endl;
    }
};

class B : public A
{
public:
    B()
    {
        cout << "B함수 생성자." << endl;
    }
};

class C : public A
{
public:
    C()
    {
        cout << "C함수 생성자." << endl;
    }
};

class D : public B, public C
{
public:
    D()
    {
        cout << "D함수 생성자." << endl;
    }
};

void main()
{
    D d;
}
```

# 가상함수

```
#include <iostream>
using namespace std;

class A
{
public:
    A()
    {
        cout << "A함수 생성자." << endl;
    }
};

class B : virtual public A
{
public:
    B()
    {
        cout << "B함수 생성자." << endl;
    }
};

class C : virtual public A
{
public:
    C()
    {
        cout << "C함수 생성자." << endl;
    }
};

class D : public B, public C
{
public:
    D()
    {
        cout << "D함수 생성자." << endl;
    }
};

void main()
{
    D d;
}
```

## Quiz

- RPG게임을 객체지향으로 설계 한 후에  
무기와 상점기능 추가.

`MapDraw::BoxErase();`

`MapDraw::MenuSelectCursor(); //설명`



(실행파일)

**Thank you**

강사 | 최정호