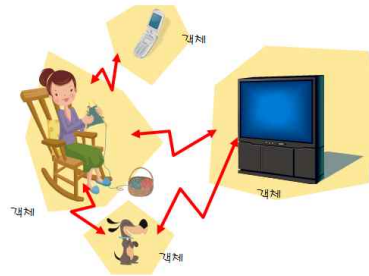


## 제9장 상속



1/40

## 이번 장에서 학습할 내용

- 상속이란?
- 접근 제어 지정자
- 상속에서의 생성자와 소멸자
- 재정의 (오버라이딩)
- 다중 상속

상속을 코드를 재사용하기 위한 중요한 기법입니다.

2/40

## 상속이란?

- 상속: 기존에 존재하는 유사한 클래스로부터 속성과 동작을 이어받고 자신이 필요한 기능을 추가하는 기법



상속



상속을 이용하면 쉽게 재산을 모을 수 있는 것처럼 소프트웨어도 쉽게 개발할 수 있다.

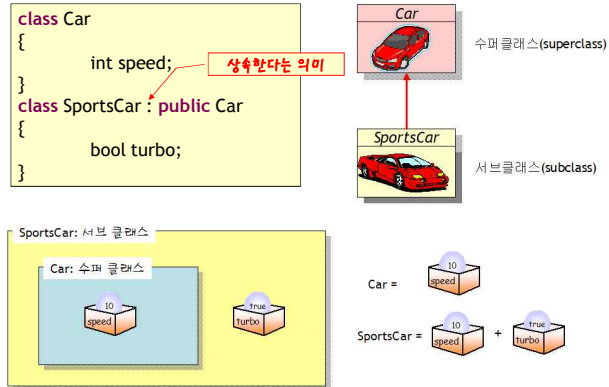
3/40

## 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.

4/40

## 상속



5/40

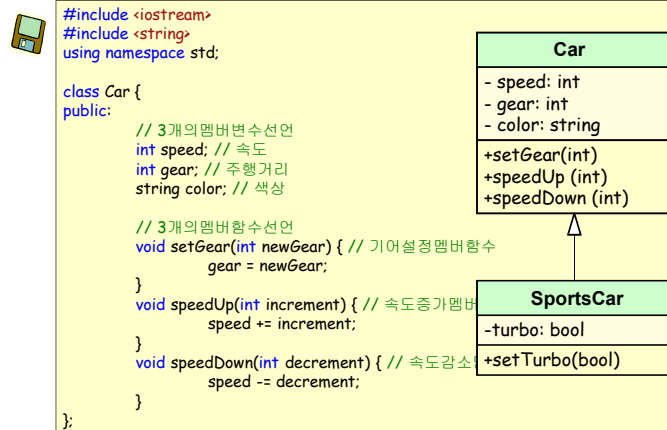
## 상속의 예

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

- 참고
- 수퍼 클래스 == 부모 클래스(parent class) == 베이스 클래스(base class)
  - 서브 클래스 == 자식 클래스(child class) == 파생된 클래스(derived class)

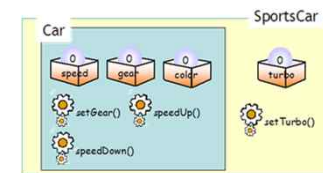
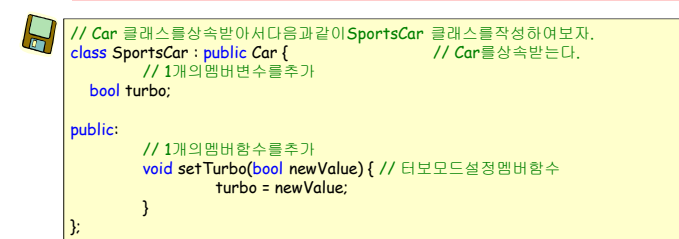
6/40

## Car 클래스



7/40

## SportsCar 클래스



8/40

## SportsCar 클래스

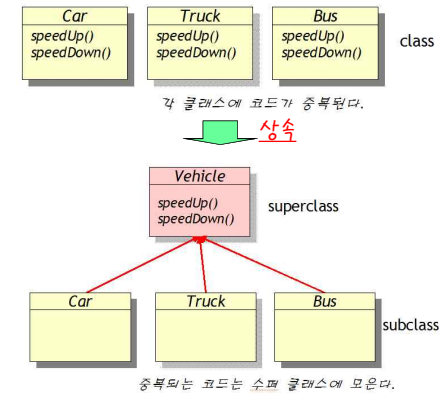


```
int main()
{
    SportsCar c;
    c.color = "Red";           // 부모클래스멤버변수접근
    c.setGear(3);              // 부모클래스멤버함수호출
    c.speedUp(100);            // 부모클래스멤버함수호출
    c.speedDown(30);           // 부모클래스멤버함수호출
    c.setTurbo(true);          // 자체멤버함수호출
    return 0;
}
```

자식 클래스는 부모 클래스의 변수와 함수를 마치 자기 것처럼 사용할 수 있다.

9/40

## 상속은 왜 필요한가?

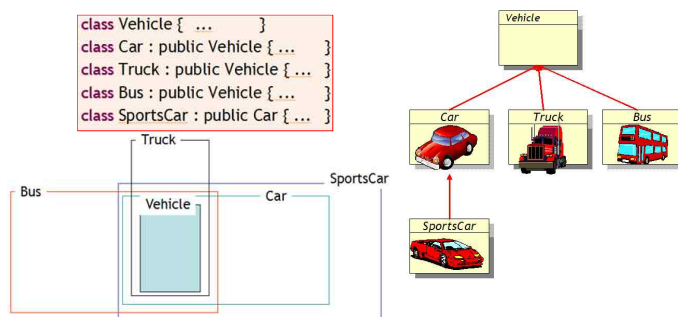


10/40

## 상속 계층도

- 상속은 여러 단계로 이루어질 수 있다.

```
class Vehicle { ... }
class Car : public Vehicle { ... }
class Truck : public Vehicle { ... }
class Bus : public Vehicle { ... }
class SportsCar : public Car { ... }
```



11/40

## 상속은 is-a 관계

- 상속은 is-a 관계
  - 자동차는 탈 것이다. (**Car is a Vehicle**).
  - 사자, 개, 고양이는 동물이다.
- has-a(포함) 관계는 상속으로 모델링을 하면 안 된다.
  - 도서관은 책을 가지고 있다(**Library has a book**).
  - 거실은 소파를 가지고 있다.

12/40

## 중간 점검 문제

1. 상속은 왜 필요한가?
2. 사자, 호랑이, 고양이를 상속 계층 구조를 이용하여 표현하여 보자.

13/40

## 접근 제어 지정자



14/40

## 예제

```
#include <iostream>
#include <string>

using namespace std;
class Employee {
    int rrrn; // 주민등록번호
protected:
    int salary; // 월급
public:
    string name; // 이름
    void setSalary(int salary);
    int getSalary();
};

void Employee::setSalary(int salary) {
    this->salary = salary;
}

int Employee::getSalary() {
    return salary;
}

class Manager : public Employee {
    int bonus;
public:
    Manager(int b=0) : bonus(b) {}
    void modify(int s, int b);
    void display();
};

void Manager::modify(int s, int b) {
    salary = s;
    bonus = b;
}

void Manager::display() {
    cout << "봉급: " << salary
    << "보너스: " << bonus << endl;
}

int main() {
    Manager m;
    m.setSalary(2000);
    m.modify(1000, 500);
    m.display();
}
```

15/40

## 중간 점검 문제

1. protected가 의미하는 바는 무엇인가?
2. 부모 클래스에서 private로 선언된 변수를 자식 클래스에서 사용할 수 있는가?
3. 자식 클래스의 객체 obj를 통하여 부모 클래스에서 protected로 선언된 변수 x를 사용할 수 있는가? 즉 obj.x = 10;과 같은 문장을 작성할 수 있는가?

16/40

## 상속에서의 생성자와 소멸자

- 자식 클래스의 객체가 생성될 때 당연히 자식 클래스의 생성자는 호출된다. 이때에 부모 클래스 생성자도 호출될까?

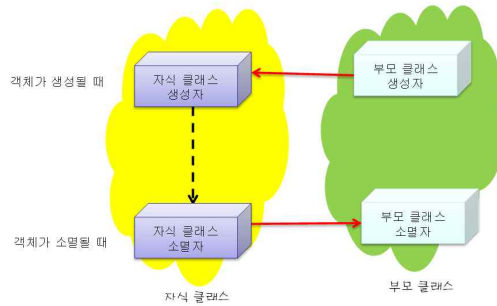


그림 8.10 상속에서 생성자와 소멸자의 호출

17/40

## 예제

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

```
class Shape {
    int x, y;
public:
    Shape() {
        cout << "Shape 생성자()" << endl;
    }
    ~Shape() {
        cout << "Shape 소멸자()" << endl;
    }
};
```

```
class Rectangle : public Shape {
    int width, height;
public:
    Rectangle() {
        cout << "Rectangle 생성자()" << endl;
    }
    ~Rectangle() {
        cout << "Rectangle 소멸자()" << endl;
    }
};

int main() {
    Rectangle r;
    return 0;
}
```

Shape 생성자()  
Rectangle 생성자()  
Rectangle 소멸자()  
Shape 소멸자()  
계속하려면 아무 키나 누르십시오 . . .

18/40

## 부모 생성자의 명시적 호출

```
class Shape {
    ...
    Shape(int xloc, int yloc) : x(xloc), y(yloc) {
        cout << "Shape 생성자(xloc, yloc)" << endl;
    }
};

class Rectangle : public Shape {
    ...
    Rectangle(int x=0, int y=0, int w=0, int h=0) : Shape(x, y) {
        width = w;
        height = h;
        cout << "Rectangle 생성자(x, y, w, h)" << endl;
    }
};
```

```
int main() {
    Rectangle r(0, 0, 100, 100);
    return 0;
}
```

Shape 생성자(xloc, yloc)  
Rectangle 생성자(x, y, w, h)  
Rectangle 소멸자()  
Shape 소멸자()

19/40

## 중간 점검 문제

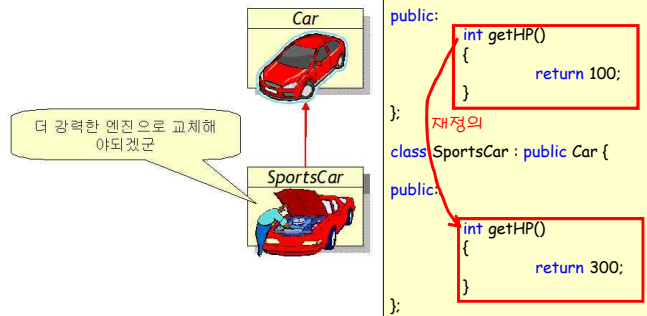
- 상속에서 자식 클래스의 생성자와 부모 클래스의 생성자 중에서 함수의 몸체가 먼저 실행되는것은?
- 상속에서 자식 클래스의 소멸자와 부모 클래스의 소멸자 중에서 함수의 몸체가 먼저 실행되는것은?



20/40

## 재정의

- 재정의(overriding): 자식 클래스가 필요에 따라 상속된 멤버 함수를 다시 정의하는 것



21/40

## 예제

```

int main()
{
    SportsCar sc;
    cout << "마력: " << sc.getHP() << endl;
    return 0;
}
    
```

마력: 300  
계속하려면 아무 키나 누르십시오 . . .

```

int main()
{
    SportsCar sc;
    cout << "마력: " << sc.Car::getHP() << endl; // 100이 출력된다.
    return 0;
}
    
```

마력: 100  
계속하려면 아무 키나 누르십시오 . . .

22/40

## 재정의의 조건

- 부모 클래스의 멤버 함수와 동일한 시그니처를 가져야 한다.
- 즉 멤버 함수의 이름, 반환형, 매개 변수의 개수와 데이터 타입이 일치하여야 한다.

```

class Animal {
public:
    void makeSound()
    {
    }
};
    
```

재정의가 아님

```

class Dog : public Animal {
public:
    int makeSound()
    {
    }
};
    
```

23/40

## 재정의와 중복 정의

- 중복 정의: 같은 이름의 멤버 함수를 여러 개 정의하는 것
- 재정의: 부모 클래스에 있던 상속받은 멤버 함수를 다시 정의하는 것



24/40

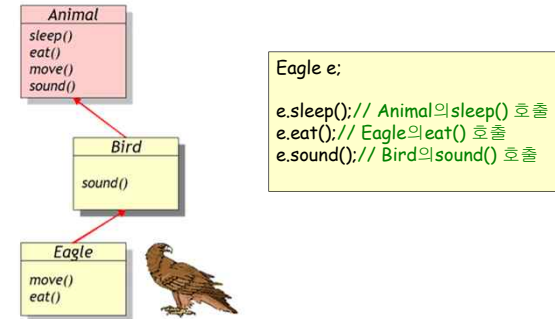
## 멤버 변수 재정의

```
class Car {
public:
    int speed;
    int gear;
    string color;
    Car(): speed(0), gear(1), color("white") {}
    void setSpeed(int s){ speed = s; }
    int getSpeed(){ return speed; }
};

class SportsCar : public Car {
public:
    int speed;
    int gear;
    string color;
    SportsCar(): speed(100), gear(3), color("blue") {}
};
```

재정의 -> 가능하지만 혼란을 일으킴!

## 재정의된 멤버 함수의 호출 순서



## 부모 클래스의 멤버 호출

```
class ParentClass {
public:
    void print() {
        cout << "부모클래스의 print() 멤버함수" << endl;
    }
};

class ChildClass : public ParentClass {
public:
    int data;
    void print() { // 멤버함수 오버라이딩
        ParentClass::print();
        cout << "자식클래스의 print() 멤버함수" << endl;
    }
};

int main()
{
    ChildClass obj;
    obj.print();
    return 0;
}
```

부모 클래스의 함수 호출!

부모 클래스의 print() 멤버 함수  
 자식 클래스의 print() 멤버 함수  
 계속하려면 아무 키나 누르십시오 ...

## 상속의 3가지 유형

```
class 자식클래스 : public 부모클래스 {
{
    ...
}
```

public로 상속

	public로 상속	protected로 상속	private로 상속
부모 클래스의 public 멤버	->public	->protected	->private
부모 클래스의 protected 멤버	->protected	->protected	->private
부모 클래스의 private 멤버	접근 안됨	접근 안됨	접근 안됨



## 중간 점검 문제

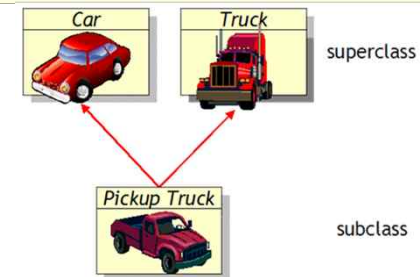
1. 부모 클래스에 **public** 변수 **x**가 있다. 부모 클래스를 **protected**로 상속받은 자식 클래스에서는 **x**의 접근 권한이 무엇으로 변경되는가?



29/40

## 다중 상속

```
class Sub : public Sup1, public Sup2
{
    ...// 추가된 멤버
    ...// 재정의된 멤버
}
```



30/40

## 예제

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

class PassengerCar {
public:
    int seats; // 정원
    void set_seats(int n){ seats = n; }
};

class Truck {
public:
    int payload; // 적재하중
    void set_payload(int load){ payload = load; }
};

class Pickup : public PassengerCar, public Truck {
public:
    int tow_capability; // 견인능력
    void set_tow(int capa){ tow_capability = capa; }
};

void main() {
    Pickup my_car;
    my_car.set_seats(4);
    my_car.set_payload(10000);
    my_car.set_tow(30000);
}
```

31/40

## 다중 상속의 문제점



```
class SuperA
{
public:
    int x;
    void sub(){
        cout << "SuperA의 sub()" << endl;
    }
};

class SuperB
{
public:
    int x;
    void sub(){
        cout << "SuperB의 sub()" << endl;
    }
};
```

32/40



## 다중 상속의 문제점

```
class Sub : public SuperA, public SuperB
{
};

int main()
{
    Sub obj;
    obj.x = 10;
    return 0;
}
```

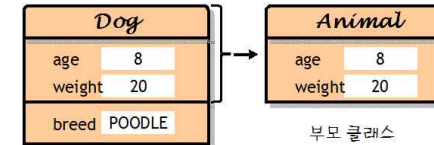
Obj.SuperA::x=10;

// obj.x는 어떤 부모클래스의 x를 참조하는가?

1> \multi\_inheri.cpp(27) : error C2385: 'x' 액세스가 모호합니다.  
 1> 기본 'SuperA'의 'x'일 수 있습니다.  
 1> 또는 기본 'SuperB'의 'x'일 수 있습니다.

33/40

## 예제



자식 클래스

부모 클래스

```
int main()
{
    Dog dog;
    dog.eat();
    dog.sleep();
    dog.speak();
    dog.wag();
    return 0;
}
```

Animal 생성자  
 Dog 생성자  
 Animal eat()  
 Animal sleep()  
 Dog speak()  
 Dog wag()  
 Dog 소멸자  
 Animal 소멸자

34/40

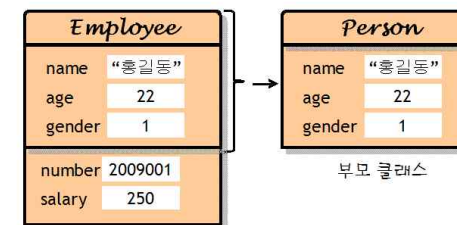
## 예제

```
enum BREED { YORKIE, POODLE, BULLDOG };
class Animal {
protected:
    int age; // 나이
    int weight; // 몸무게
public:
    Animal() {
        cout << "Animal 생성자\n";
    }
    ~Animal() {
        cout << "Animal 소멸자\n";
    }
    void speak() const {
        cout << "Animal speak()\n";
    }
    void sleep() const {
        cout << "Animal sleep()\n";
    }
    void eat() const {
        cout << "Animal eat()\n";
    }
};

class Dog : public Animal
{
private:
    BREED breed;
public:
    Dog() {
        cout << "Dog 생성자\n";
    }
    ~Dog() {
        cout << "Dog 소멸자\n";
    }
    void wag() {
        cout << "Dog wag()\n";
    }
    void bite() {
        cout << "Dog bite()\n";
    }
    void speak() const {
        cout << "Dog speak()\n";
    }
};
```

35/40

## 예제 #2



자식 클래스

부모 클래스

36/40

## 예제

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

class Person {
    string name;
    int age;
    bool gender;
public:
    Person(string n="", int a=0, bool g=true)
        : name(n), age(a), gender(g) { }
    void setName(string s) { name = s; }
    string getName() const { return name; }
    void setAge (int a) { age = a; }
    int getAge() const { return age; }
    void setGender (bool g) { gender = g; }
    bool getGender() const { return gender; }
};
```

37/40

## 예제

```
class Employee : public Person {
    int number;
    int salary;
public:
    Employee(string n="", int a=0, bool g=true, int num=0, int s=0)
        : Person(n, a, g), number(num), salary(s) { }
    void display() const;
    void setNumber (int n) { number = n; }
    int getNumber() const { return number; }
    void setSalary (int s) { salary = s; }
    int getSalary() const { return salary; }
};

void Employee::display() const
{
    cout << this->getName() << endl;
    cout << this->getAge() << endl;
    cout << this->getGender() << endl;
    cout << this->getNumber() << endl;
    cout << this->getSalary() << endl;
}
```

38/40

## 예제

```
int main()
{
    Employee e("김철수", 26, true, 2010001, 2800);
    e.display();
    return 0;
}
```

```
김철수
26
1
2010001
2800
계속하려면 아무 키나 누르십시오 . . .
```

39/40

## Q & A



40/40