7장. 상속(Inheritance)의 이해

부산대학교
김미경
ddosun@pusan.ac.kr

7-1 상속으로 들어가기에 앞서서

- 상속을 정복하기 위한 접근 방식
  - 첫 째 : 문제의 제기
  - 둘 째 : 기본 개념 소개
  - 셋 째 : 문제의 해결

- 급여 담당자의 요구사항
  - 급여 관리를 위해 직원 정보의 저장
    - 고용직(Permanent): 연봉제(매달 급여가 결정되어 있다)
  - 매달 지불되어야 할 급여 정보 확인
  - EmployeeManager1.cpp

```cpp
/////*   EmployeeManager1.cpp */////
#include <iostream>
#include <cstring>
using std::endl;using std::cout;

class Permanent
{
private:
    char name[20];
    int salary;
public:
    Permanent(const char* _name, int sal);
    const char* GetName();
    int GetPay();
};
Permanent::Permanent(const char* _name, int sal) {
    strcpy(name, _name);
    salary=sal;
}
const char* Permanent::GetName()
{
    return name;
}
int Permanent::GetPay()
{
    return salary;
}
```

```cpp
class Department
{
private:
    Permanent* empList[10];
    int index;
public:
    Department(): index(0) { };
    void AddEmployee(Permanent* emp);
    void ShowList(); // 급여 리스트 출력.
};

void Department::AddEmployee(Permanent* emp)
{
    empList[index++]=emp;
}
void Department::ShowList() // 급여 리스트 출력.
{
    for(int i=0; i<index; i++)
    {
        cout<<"name: "<<empList[i]->GetName()<<endl;
        cout<<"salary: "<<empList[i]->GetPay()<<endl;
        cout<<endl;
    }
}
```

```cpp
int main()
{
    //직원을 관리하는 CONTROL 클래스
    Department department;

    //직원 등록.
    department.AddEmployee(new Permanent("KIM", 1000));
    department.AddEmployee(new Permanent("LEE", 1500));
    department.AddEmployee(new Permanent("JUN", 2000));

    //최종적으로 이번달에 지불해야할 급여는?
    department.ShowList();
    return 0;
}
```

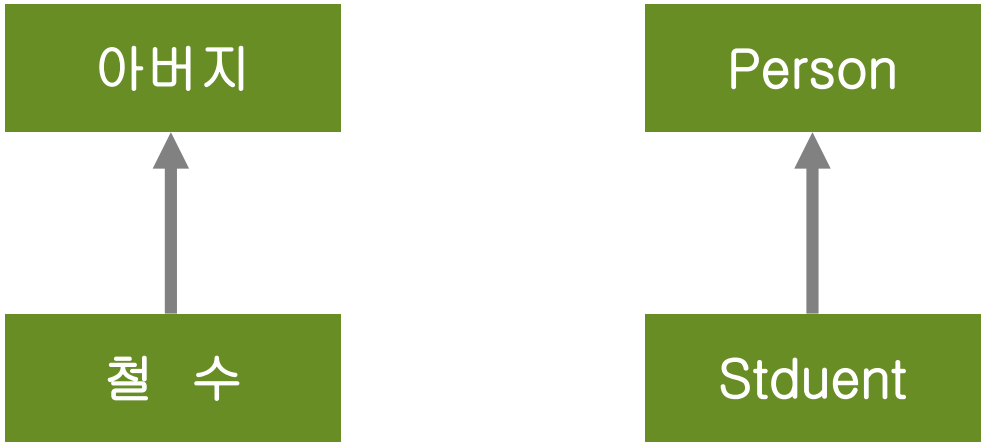
7-1 상속으로 들어가기에 앞서서

- 요구 사항의 변경
  - 급여의 형태 다양화
  - 판매직(Sales Person): 연봉제 + 인센티브제
  - 임시직(Temporary): 일한 시간 × 시간당 급여

- 요구 사항의 변경을 위해 해야 하는 일들
  - EmployeeManager1.cpp
  - Department 클래스의 대대적인 변경

7-2 상속의 기본 개념

- 상속의 예1
  - "철수는 아버지로부터 좋은 목소리와 큰 키를 물려 받았다."
- 상속의 예2
  - "Student 클래스가 Person 클래스를 상속한다."



7-2 상속의 기본 개념

```
AAA() call!
BBB() call!
a : 0
b : 0

AAA() call!
BBB(int j) call!
a : 0
b : 50

BBB 소멸자 call
AAA 소멸자 call
BBB 소멸자 call
AAA 소멸자 call
```

첫 째 : 메모리 공간 할당

둘 째 : Base 클래스의 생성자 실행

셋 째 : Derived 클래스의 생성자 실행

```
1  #include <iostream>
2  using std::endl;using std::cout;
3
4  class AAA  {  //Base 클래스
5      int a;
6  public:
7      AAA(){
8          cout<<"AAA() call!"<<endl;
9          a=0;
10     }
11     AAA(int i){
12         cout<<"AAA(int i) call!"<<endl;
13         a = i;
14     }
15     ~AAA() {
16         cout << "AAA 소멸자 call" <<endl;
17     }
18     void showData(){
19         cout << "a : "  << a <<endl;
20     }
21 };
```

```
22 class BBB : public AAA {   //Derived 클래
23     int b;
24 public:
25     BBB(){
26         cout<<"BBB() call!"<<endl;
27         b=0;
28     }
29     BBB(int j) : AAA(){
30         cout<<"BBB(int j) call!"<<endl;
31         b=j;
32     }
33     ~BBB() {
34         cout << "BBB 소멸자 call" <<endl;
35     }
36     void showData(){
37         AAA::showData();
38         cout << "b : "  << b <<endl;
39     }
40 };
```

BBB(int j):AAA(j)로 바뀌면 ?

```
41 int main(void)
42 {
43     BBB b1;
44     b1.showData();
45     cout << endl;
46
47     BBB b2(50);
48     b2.showData();
49     cout << endl;
50 }
```

7-2 상속의 기본 개념

```
class AAA     //Base 클래스
{
public:
    AAA(){  }
    AAA(int i){  }
};
```

```
class BBB : public AAA    //Derived 클래스
{
public:
    BBB(){  }
    BBB(int j){  }
};

int main(void)
{
   BBB b2;
   return 0;
}
```

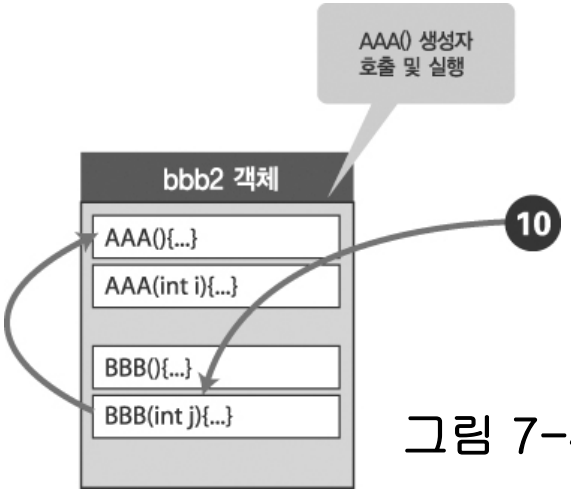


그림 7-4

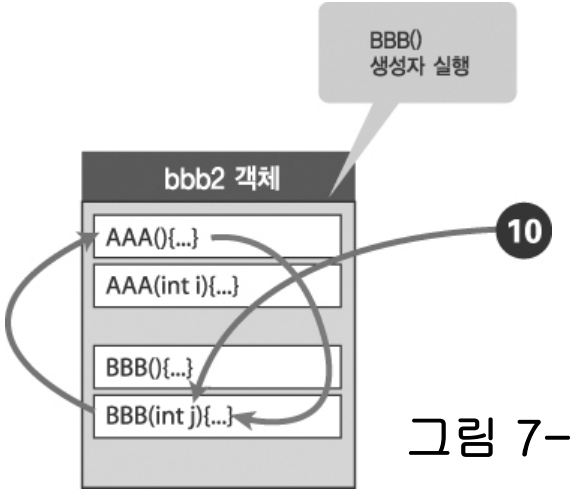


그림 7-5

7-2 상속의 기본 개념

```
class Person
{
    int age;
    char name[20];
public:
    int GetAge() const {   }
    const char* GetName() const {   }
    Person(int _age=1, char* _name="noname") {   }
};
```

Base class

Derived class

```
class Student: public Person
{
    char major[20]; //전공
public:
    Student(char* _major){   }
    const char* GetMajor() const {   }
    void ShowData() const {   }
};
```

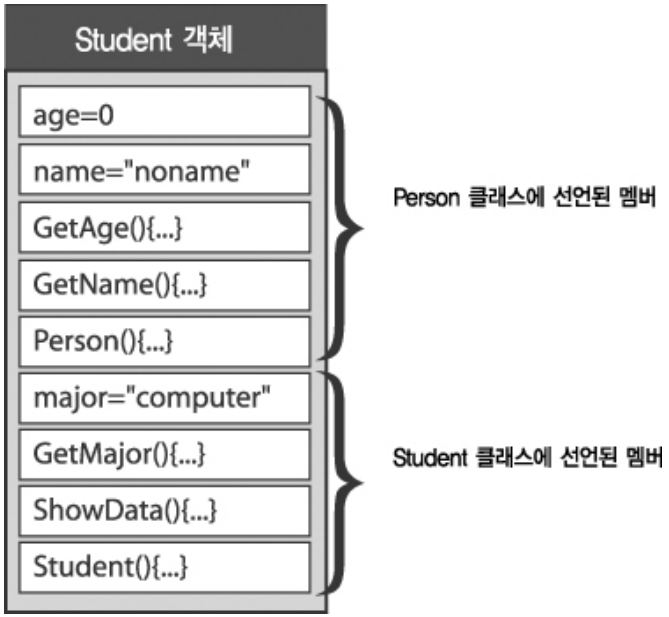


그림 7-1

7-2 상속의 기본 개념

```
1  /*   BasicInheri1.cpp*/
2  #include <iostream>
3  using std::endl;using std::cout;
4
5  class Person
6  {
7      int age;
8      char name[20];
9  public:
10
11     int GetAge() const {
12         return age;
13     }
14     const char* GetName() const {
15         return name;
16     }
17
18     Person(int _age=1, char* _name="noname"){
19         age=_age;
20         strcpy(name, _name);
21     }
22 };
```

```
24 class Student: public Person
25 {
26     char major[20]; //전공
27 public:
28     Student(char* _major){
29         strcpy(major, _major);
30     }
31     const char* GetMajor() const {
32         return major;
33     }
34     void ShowData() const {
35         cout<<"이 름 : "<<GetName()<<endl;
36         cout<<"나 이 : "<<GetAge()<<endl;
37         cout<<"전 공 : "<<GetMajor()<<endl;
38     }
39 };
40 int main(void)
41 {
42     Student Kim("computer");
43     Kim.ShowData();
44
45     return 0;
46 }
```

```
이 름 : noname
나 이 : 1
전 공 : computer
```

눈세: 결과➔원하는 값으로 초기화 불가능

7-3 객체의 생성 및 소멸 과정

- 멤버 이니셜라이저 : 부모 클래스 변수의 값을 원하는 값으로 초기화 하기 위해
  - Base 클래스의 생성자 명시적 호출
  - BasicInheri2.cpp

```
BBB(int j) : AAA(j) {
    cout<<"BBB(int j) call!"<<endl;
}
```

7-3 객체의 생성 및 소멸 과정

- basicInheri1.cpp의 student 생성자를 다음과 같이 고쳐서
- basicInheri2.cpp 로 저장➔error
- 해결은? ➔멤버이니셜라이즈 통해 부모클래스의 생성자 호출

```
Student(int _age, char* _name, char* _major){
    age=_age;
    strcpy(name, _name);
    strcpy(major, _major);
}
```

basicInheri3.cpp

```
이름: Hong Gil Dong
나이: 20
전공: computer
```

```
Student(int _age, char* _name, char* _major)
    : Person(_age, _name)
{
    strcpy(major, _major);
}
```

7-3 객체의 생성 및 소멸 과정

- 소멸자의 호출 순서

```
class AAA     //Base 클래스
{
public:
    AAA(){
        cout<<"AAA() call!"<<endl;
    }
    ~AAA(){cout<<"~AAA() call!"<<endl;
     }
};
```

```
class BBB : public AAA     //Derived 클래스
{
public:
    BBB(){
        cout<<"BBB() call!"<<endl;
    }
    ~BBB(){
         cout<<"~BBB() call!"<<endl;
     }
};
```

- **첫째 : Derived 객체 소멸자 호출**
- **둘째 : Base 객체 소멸자 호출**
- **셋째 : 메모리 반환**

7-4 protected 멤버

- **protected 멤버**
  - 상속 관계에 놓여있을 경우 접근을 허용
  - 그 이외는 private 멤버와 동일
  - BasicInheri4.cpp

```
class AAA
{
private:  int a;
protected:  int b;
};
class BBB : public AAA
{
public:
    void SetData(){
        a=10;  // private 멤버. Error!
        b=20;  // protected 멤버
    }
};
```

```
int main(void)
{
    AAA aaa;
    // private 멤버. 따라서 Compile Error.
    aaa.a=10;

    // protected 멤버. 따라서 Compile Error.
    aaa.b=20;
    return 0;
};
```

protected

```
1  /*   BasicInheri4.cpp*/
2  #include <iostream>
3  using std::endl;using std::cout;
4  class Person
5  {
6  protected:
7      int age;
8      char name[20];
9  public:
10     int GetAge() const {
11         return age;
12     }
13     const char* GetName() const {
14         return name;
15     }
16
17     Person(int _age=1, char* _name="noname"){
18         age=_age;
19         strcpy(name, _name);
20     }
21 };
```

```
23 class Student: public Person
24 {
25     char major[20]; //전공
26 public:
27     Student(int _age, char* _name, char* _major)
28         : Person(_age, _name)
29     {
30         strcpy(major, _major);
31     }
32     const char* GetMajor() const {
33         return major;
34     }
35     void ShowData() const {
36         cout<<"이름 : "<<age<<endl;
37         cout<<"나이 : "<<name<<endl;
38         cout<<"전공 : "<<major<<endl;
39     }
40 };
41 int main(void){
42     Student Kim(20, "Hong Gil Dong", "computer");
43     Kim.ShowData();
44 };
```

7-5 세 가지 형태의 상속

- **접근 권한 변경**
  - Base 클래스의 멤버는 상속되는 과정에서 접근 권한 변경

상속 형태 / Base 클래스	public 상속	protected 상속	private 상속
public 멤버	public	protected	private
Protected 멤버	protected	protected	private
Private 멤버	접근 불가	접근 불가	접근 불가

7-5 세 가지 형태의 상속

```
class Base
{
private:
    int a;
protected:
    int b;
public:
    int c;
};

class Derived : public Base  // public 상속
{
   // EMPTY
};
```

```
int main(void)
{
   Derived object;
   return 0;
};
```

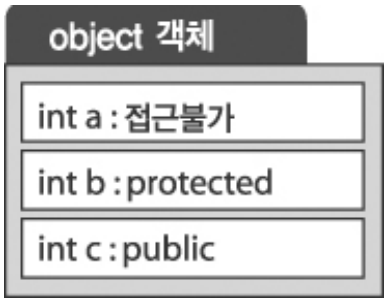


그림 7-8

7-6 상속을 하는 이유

- **문제의 도입**
  - 운송 수단에 관련된 클래스 정의
  - 자동차(car), 열차(train), 선박(ship), 비행기(Airplane) 등등

```
class Airpalne
{
    int passenger;              // 탑승 인원
    int baggage;                // 수하물의 무게
    int crew_man;          // 승무원 인원
public:
    Airpalne(int p=0, int w=0, int c=0);
    void Ride(int person);      //탑승하다.
    void Load(int weight)       //짐을 싣다.
    void TakeCrew(int crew)  //승무원 탑승
}
```

```
class Train
{
    int passenger;              //탑승 인원
    int baggage;                //수하물의 무게
    int length;               //열차의 칸 수
public:
    Train(int p=0, int w=0, int l=0);
    void Ride(int person);      //탑승하다.
    void Load(int weight);      //짐을 싣다.
    void SetLength(int len);    //열차의 칸 수 설정
};
```

- **문제의 해결**

```
class Vehicle
{
    int passenger;
    int baggage;
public:
    Vehicle(int person=0, int weight=0);
    void Ride(int person);              //탑승하다.
    void Load(int weight);              //짐을 싣다.
};
```