7. 상속(inheritance)의 이해

- 상속의 기본 개념
- 상속의 생성자, 소멸자
- protected 멤버

Jong Hyuk Park

상속의 기본 개념

Jong Hyuk Park

상속의 기본 개념

❖ 상속의 예1

- "철수는 아버지로부터 좋은 목소리와 큰 키를 물려 받았다."

❖ 상속의 예2

- "Student 클래스가 Person 클래스를 상속한다."

파생 클래스 (derived class)

- **상속(inheritance)**
  - 한 클래스가 다른 클래스에서 정의된 속성(자료,함수)를 이어받아 그대로 사용
  - 이미 정의된 클래스를 바탕으로 필요한 기능을 추가하여 정의
  - 소프트웨어 재사용 지원

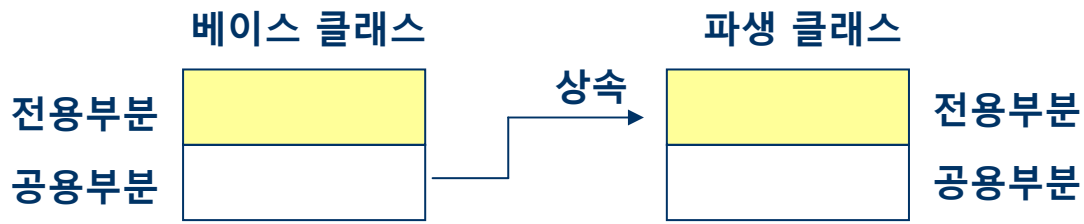
- **파생 클래스는 C++에서 각 클래스의 속성을 공유하고 물려받는 객체지향 프로그래밍의 상속(inheritance)을 구현한 것**
  - 이미 만들어진 기존의 클래스를 베이스 클래스(base class) 또는 부모 클래스(parent class)
  - 이를 상속 받아 새로 만들어지는 클래스를 파생 클래스(derived class)라 함

공용부분 상속

```
class 파생클래스명 : public[private] 베이스클래스명  {
        ……
};
```

public

private



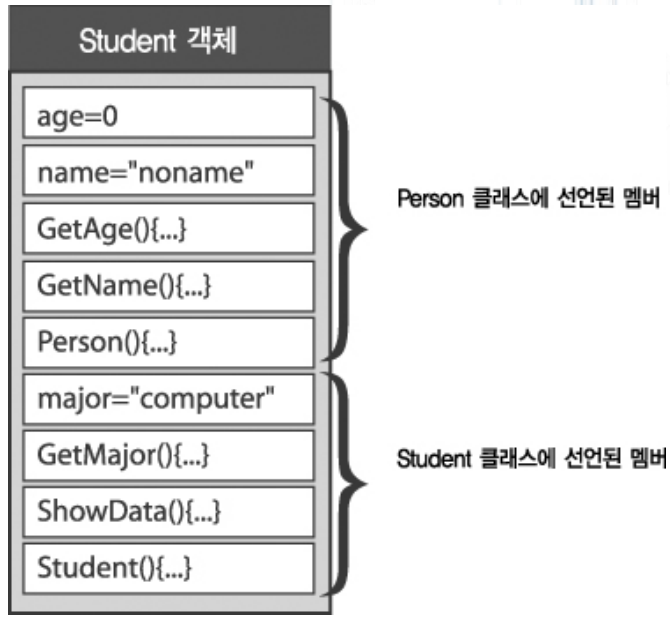
베이스 클래스와 파생 클래스 예

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

Base class

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

Derived class



public 상속 예 1

```
#include <iostream>
using std::endl;
using std::cout;

class Person
{
	int age;
	char name[20];
public:

	int GetAge() const {
		return age;
	}
	const char* GetName() const {
		return name;
	}

	Person(int _age=1, char* _name="noname"){
		age=_age;
		strcpy(name, _name);
	}
};
```

```
class Student: public Person
{
	char major[20]; //전공
public:
	Student(char* _major){
		strcpy(major, _major);
	}
	const char* GetMajor() const {
		return major;
	}
	void ShowData() const {
		cout<<"이름: "<<GetName()<<endl;
		cout<<"나이: "<<GetAge()<<endl;
		cout<<"전공: "<<GetMajor()<<endl;
	}
};

int main(void)
{
	Student Kim("computer");
	Kim.ShowData();

	return 0;
};
```

public 상속 예 2

```
#include <iostream>
using std::endl;
using std::cout;

class Point {                  // 베이스클래스 Point
        int px,py;
public:
        void setpt(int x, int y) { px = x; py = y; }
        int  getx()                    { return px; }
        int  gety()                    { return py; }
};

class Circle : public Point {
                        // Point의 파생클래스 Circle
        float r;
public:
        void setrd(float d)              { r = d;  }
        float area()               { return 3.14*r*r; }
};
```

```
int main()
{
 Circle c;

  c.setpt(10,20);  // 베이스클래스의 멤버 호출
  c.setrd(12.5);

  // 베이스 클래스의 멤버 호출
  cout << "Center of Circle : x = " << c.getx();
  cout << ", y = " << c.gety() << endl;
  cout << "Area of circle   : " << c.area() << endl;

  return 0;
}
```

```
Center of Circle : x = 10, y = 20
Area of circle   : 490.625
```

베이스 클래스 Point의 공용멤버 setpt(), getx(), gety() 함수를 Point의 파생 클래스인 Circle이 공용멤버로 상속받아서 객체 c가 호출한 예이다.

private 상속 예

```
#include <iostream>
using std::endl;
using std::cout;

class Point {                    // 베이스 클래스 Point
        int px,py;
public:
        void setpt(int x, int y) { px = x; py = y; }
        int  getx()                    { return px; }
        int  gety()                    { return py; }
};

class Circle : private Point {
                // Point의 파생 클래스 Circle
        float r;
public:
        void setrd(float d)              { r = d;  }
        float area()               { return 3.14*r*r; }
        void center(int x, int y)       { setpt(x,y); }
              // 베이스 클래스의 멤버 setpt() 호출

        void showxy()
          {  cout << "Center of Circle : x = " << getx();
              cout << ", y = " << gety() << '₩n';
           }
             // 베이스 클래스의 멤버 getx(),gety() 호출
};
```

```
int main()
{
    Circle c;

    c.center(10,20);
    c.setrd(12.5);

    c.showxy();
    cout << "Area of circle    : " << c.area() << '₩n';

     return 0;
}
```

```
Center of Circle : x = 10, y = 20
Area of circle    : 490.625
```

베이스 클래스 Point의 공용멤버 setpt(), getx(), gety() 함수를Point의 파생 클래스인 Circle이 전용멤버로 상속받았기 때문에 클래스의 외부에서는 이 함수들에 접근할 수 없으므로 클래스의 멤버함수인 center(),showxy() 함수를 통하여 이 함수들을 상속받아 호출한 예이다.

클래스 외부의 주 프로그램에서 c.setpt()와 같이 호출하면 setpt()는 객체 c의 전용멤버이므로 에러가 발생한다.

상속의 생성자, 소멸자

Jong Hyuk Park

생성자, 소멸자

- 베이스 클래스, 파생 클래스가 생성자, 소멸자를 갖는 경우에 생성자는 파생되는 순서에 따라 호출되고 소멸자는 파생된 순서의 반대로 호출
  - 파생 클래스의 객체가 생성될 때 먼저 베이스 클래스의 생성자가 호출되어 실행된 후 파생 클래스의 생성자가 호출되어 실행
  - 소멸자인 경우에는 실행 순서가 생성자와는 반대로 파생 클래스의 소멸자가 실행된 후 베이스 클래스의 소멸자가 실행

- 베이스 클래스의 생성자에 인수를 전달하고자 하는 경우 파생 클래스의 생성자를 정의할 때 베이스 클래스에 전달될 인수를 다음과 같이 기술

```
파생클래스_생성자명(인수리스트) : 베이스클래스_생성자명(인수리스트) {
        .....
};
```

상속의 생성자 예 1

```
#include <iostream>
using std::endl;
using std::cout;

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

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

```
int main(void)
{
    cout << "객체 1 생성 " << endl;
    BBB bbb1;

    cout << "객체 2 생성 " << endl;
    BBB bbb2(10);

    return 0;
}
```

```
객체 1 생성
AAA() call!
BBB() call!
객체 2 생성
AAA(int i) call!
BBB(int j) call!
```

상속의 생성자 예 2

```
#include <iostream>
using std::endl;
using std::cout;

class Person
{
	int age;
	char name[20];
public:

	int GetAge() const {
		return age;
	}
	const char* GetName() const {
		return name;
	}

	Person(int _age=1, char* _name="noname"){
		age=_age;
		strcpy(name, _name);
	}
};
```

```
class Student: public Person
{
	char major[20]; //전공
public:
	Student(int _age, char* _name, char* _major){
		age=_age;	// 컴파일 에러
		strcpy(name, _name);	// 컴파일 에러
		strcpy(major, _major);
	}
	const char* GetMajor() const {
		return major;
	}
	void ShowData() const {
		cout<<"이름: "<<GetName()<<endl;
		cout<<"나이: "<<GetAge()<<endl;
		cout<<"전공: "<<GetMajor()<<endl;
	}
};

int main(void)
{
	Student Kim(20, "Hong Gil Dong", "computer");
	Kim.ShowData();

	return 0;
};
```

상속의 생성자 예 3

```
#include <iostream>
using std::endl;
using std::cout;

class Person
{
	int age;
	char name[20];
public:

	int GetAge() const {
		return age;
	}
	const char* GetName() const {
		return name;
	}

	Person(int _age=1, char* _name="noname"){
		age=_age;
		strcpy(name, _name);
	}
};
```

```
class Student: public Person
{
	char major[20]; //전공
public:
	Student(int _age, char* _name, char* _major)
		: Person(_age, _name)
	{
		strcpy(major, _major);
	}
	const char* GetMajor() const {
		return major;
	}
	void ShowData() const {
		cout<<"이름: "<<GetName()<<endl;
		cout<<"나이: "<<GetAge()<<endl;
		cout<<"전공: "<<GetMajor()<<endl;
	}
};
int main(void)
{
	Student Kim(20, "Hong Gil Dong", "computer");
	Kim.ShowData();
	return 0;
};
```

상속의 소멸자 예

```
#include <iostream>
using std::endl;
using std::cout;

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

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

```
int main(void)
{
    BBB bbb;

    return 0;
}
```

```
AAA() call!
BBB() call!
~BBB() call!
~AAA() call!
```

protected 멤버

Jong Hyuk Park

protected 멤버 상속 (1)

- 파생 클래스의 멤버함수는 베이스 클래스의 공용멤버에 직접 접근이 가능하지만 **베이스 클래스의 전용멤버**에 대해서는 접근할 수 없음

- 파생 클래스의 멤버함수가 베이스 클래스의 전용멤버를 상속받아 자유로이 사용을 위해서는 다음 두 가지 방식 사용
  - 프렌드 함수를 사용
  - 베이스 클래스 정의 시 protected 키워드를 사용한 보호부분을 정의

- 보호부분에 있는 멤버는 전용멤버와 같이 외부에서는 직접 접근할 수 없지만 파생 클래스의 멤버함수에서는 직접 접근이 가능
  - 보호부분 멤버가 파생 클래스에서 public으로 상속 받으면 파생 클래스의 protected 멤버가 됨
  - private으로 상속 받으면 파생 클래스에서 전용부분 멤버가 됨

protected 멤버 상속 (2)

```
class 클래스명 {
[private:]              // 전용부분 멤버 정의
        .....           // 클래스 외부 접근 및 상속 안됨
protected:              // 보호부분 멤버 정의
        .....           // 클래스 외부 접근 안됨, 상속 됨
public:                 // 공용부분 멤버 정의
        .....           // 클래스 외부 접근 및 상속 됨


};
```

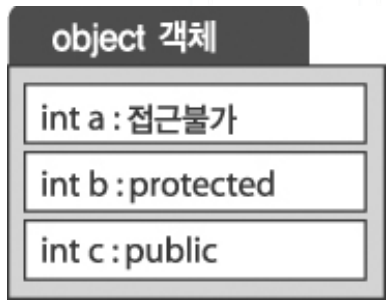
상속 형태 / Base 클래스	public 상속	protected 상속	private 상속
public 멤버	public	protected	private
Protected 멤머	protected	protected	private
Private 멤버	접근 불가	접근 불가	접근 불가

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;
};
```



protected 멤버 상속 예 1

```
class AAA
{
private:
    int a;
protected:
    int b;
};

class BBB: public AAA
{
public:
    void SetData() {
        a=10;  // private 멤버, 따라서 에러
        b=10; // protected 멤버, OK!
    }
};
```

```
int main(void)
{
    AAA aaa;
    aaa.a=10;              // private 멤버,
     따라서 에러
    aaa.b=20; // protected 멤버, 따라서
     에러!!

    BBB bbb;
    bbb.SetData();

    return 0;
}
```

protected 멤버 상속 예 2

```
#include <iostream>
using std::endl;
using std::cout;

class Person
{
protected:
	int age;
	char name[20];
public:

	int GetAge() const {
		return age;
	}
	const char* GetName() const {
		return name;
	}

	Person(int _age=1, char* _name="noname"){
		age=_age;
		strcpy(name, _name);
	}
};
```

```
class Student: public Person
{
	char major[20]; //전공
public:
	Student(int _age, char* _name, char* _major){
		age=_age;
		strcpy(name, _name);
		strcpy(major, _major);
	}
	const char* GetMajor() const {
		return major;
	}
	void ShowData() const {
		cout<<"이름: "<<GetName()<<endl;
		cout<<"나이: "<<GetAge()<<endl;
		cout<<"전공: "<<GetMajor()<<endl;
	}
};
int main(void)
{
	Student Kim(20, "Hong Gil Dong", "computer");
	Kim.ShowData();
	return 0;
};
```

상속을 하는 이유

❖ 문제의 도입

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

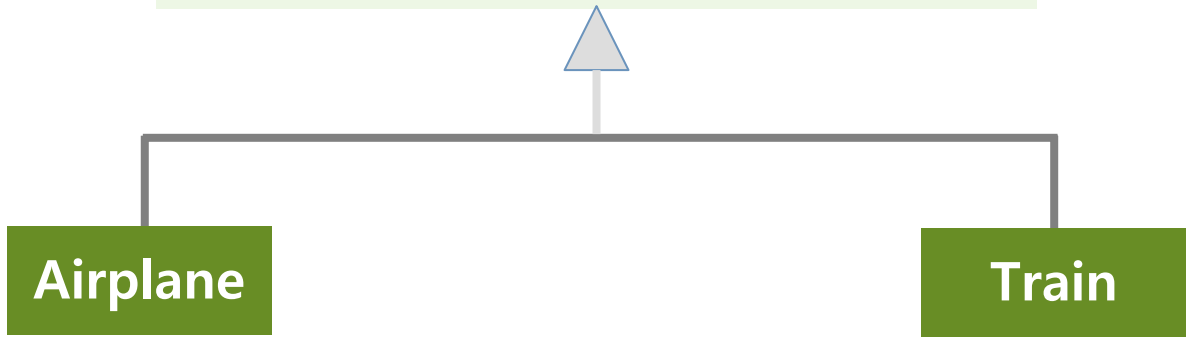
```
class Airpalne
{
    int passenger;              // 탑승 인원
    int baggage;                // 수하물의 무게
    int crew_man;                   // 승무원
인원
public:
    Airpalne(int p=0, int w=0, int c=0);
    void Ride(int person);      //탑승하다.
    void Load(int weight)       //짐을 싣다.
    void TakeCrew(int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

상속을 하는 이유

❖ 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.
};
```



과제

1. 연습문제 7-2: Person 클래스 프로그램

2. 7-8절의 은행계좌 관리 과제 프로그램에 다음을 추가하여라
   - 고객의 전화번호 정보를 추가하여라. 전화번호는 다음과 같이 private으로 선언하라.
     char *phone;
   - (이름,전화번호)와 계좌번호로 잔액조회를 하는 Inquire() 오버로딩 함수를 작성하여라.
     - void Inquire(char *name, char *phone);
     - void Inquire(int id);


Q & A
Jong Hyuk Park