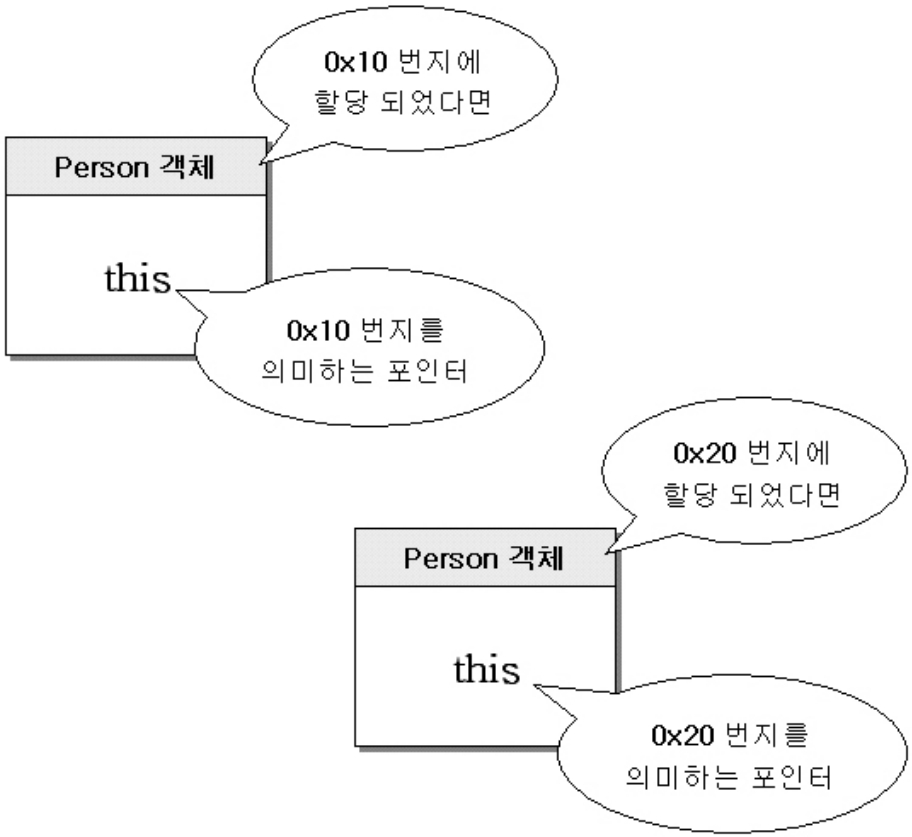
4장. 클래스의 완성_추가

담당교수 : 김미경

4-5 this 포인터

- this 포인터의 의미
  - this_under.cpp가 말해준다!



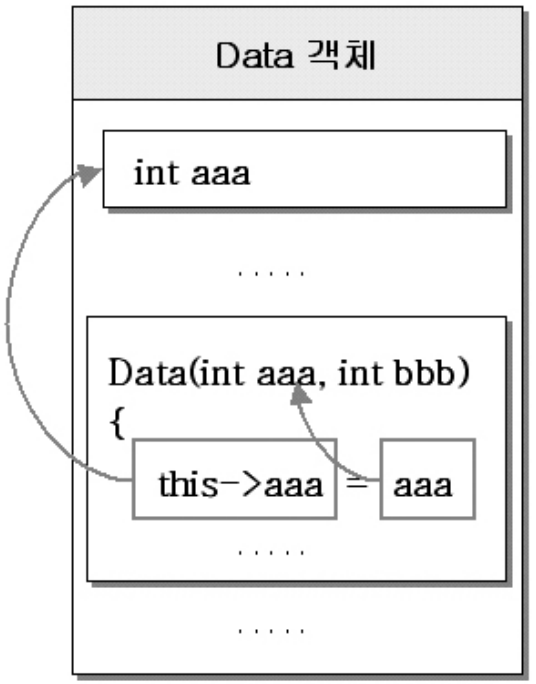
4-5 this 포인터

- **this 포인터의 용도**
  - **연산자 오버로딩에서 유용하게 사용**

```
class Data
{
    int aaa;
    int bbb;

public :
    Data(int aaa, int bbb) {
        //aaa=aaa;
        this->aaa=aaa;

        //bbb=bbb;
        this->bbb=bbb;
    }
    void printAll() {
        cout<<aaa<<" "<<bbb<<endl;
    }
};
```



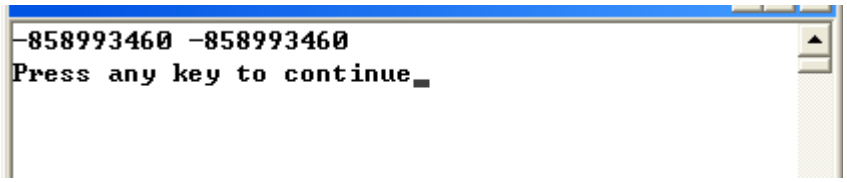
4-5 this 포인터

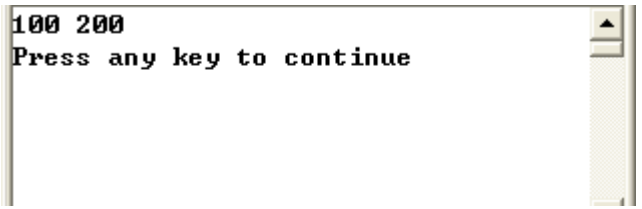
```
1  /*   this_under.cpp*/
2  #include <iostream>
3  using std::cout;
4  using std::endl;
5
6  class Person
7  {
8  public:
9      Person* GetThis(){
10         return this; //this 포인터를 리턴.
11     }
12 };
13
14 int main()
15 {
16     cout<<"***** p1의 정보 ******"<<endl;
17     Person *p1 = new Person();
18     cout<<"포인터 p1: "<<p1<<endl;
19     cout<<"p1의 this: "<<p1->GetThis()<<endl;
20
21     cout<<"***** p2의 정보 ******"<<endl;
22     Person *p2 = new Person();
23     cout<<"포인터 p2: "<<p2<<endl;
24     cout<<"p2의 this: "<<p2->GetThis()<<endl;
25
26     return 0;
27 }
```

```
***** p1의 정보 ******
포인터 p1: 00D785E0
p1의 this: 00D785E0
***** p2의 정보 ******
포인터 p2: 00D748B0
p2의 this: 00D748B0
```

4-5 this 포인터

```
1  //////* app_this.cpp*//////
2  #include <iostream>
3  using std::cout;using std::endl;
4
5  class Data
6  {
7      int aaa;
8      int bbb;
9  public :
10     Data(int aaa, int bbb) {
11         //aaa=aaa;
12         this->aaa=aaa;
13         //bbb=bbb;
14         this->bbb=bbb;
15     }
16     void printAll() {
17           cout<<aaa<<" "<<bbb<<endl;
18     }
19 };
20 int main(void)
21 {
22     Data d(100, 200);
23     d.printAll();
24     return 0;
25 }
```



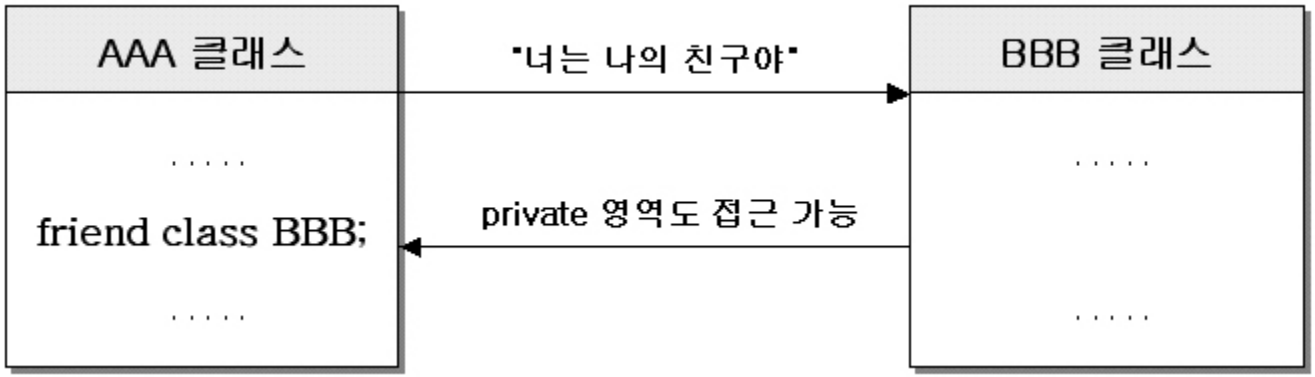


4-5 this 포인터

```
1  /* this_imsi.cpp */
2  #include <iostream>
3  using namespace std;
4
5  class A{
6      public:
7          int a;
8          void fct()
9          {
10             int a=20;
11             cout << "fct안의 a : " <<a << endl;
12             cout << "class A의 a : " << this->a << endl;
13         }
14 };
15 int main()
16 {
17     A* p = new A();
18     p->a = 10;
19     p->fct()     ;
20 }
```

4-6 friend 선언

- 전역 함수에 대한 friend 선언
  - friend1.cpp
  - 전역 함수에게 private 영역 접근 허용!
- class에 대한 friend 선언
  - friend2.cpp



4-6 friend 선언

```cpp
/*   friend1.cpp*/
#include <iostream>
using namespace std;
class Counter{
private:
    int val;
public:
    Counter() {
        val=0;
    }
    void Print() const {
        cout<<val<<endl;
    }
    friend void SetX(Counter& c, int val);  //friend 선언.
};

void SetX(Counter& c, int val){ // 전역함수.
    c.val=val;
}

int main(){
    Counter cnt;
    cnt.Print();

    SetX(cnt, 2002);
    cnt.Print();
    return 0;
}
```

```cpp
/*   friend2.cpp*/
#include <iostream>
using namespace std;
class AAA{
private:
    int data;
    friend class BBB;  // class BBB를 friend로 선언함!
};

class BBB{
public:
    void SetData(AAA& aaa, int val){
        aaa.data=val; //class AAA의 private 영역 접근!
    }
};

int main()
{
    AAA aaa;
    BBB bbb;

    bbb.SetData(aaa, 10);

    return 0;
}
```

4-6 friend 선언

- friend 선언의 유용성
  - 유용하지 않다!
  - 정보 은닉에 위배되는 개념
  - 연산자 오버로딩에서 유용하게 사용
  - 그 전에는 사용하지 말자!

- friend 선언으로만 해결 가능한 문제
  - 그런 것은 존재하지 않는다.
  - 연산자 오버로딩에서는 예외!