객체지향 프로그래밍 실습과제보고서 #9

강의: 객체지향 프로그래밍
교수님: 최경주 교수님
제출일: 2019 년 5 월 13 일
제출자: 김예원
학과: 소프트웨어학과

실습과제 #9: 객체지향방식의 성적처리프로그램 #2 작성

학생의 정보를 수정하는 Modify 함수를 만들고, 각 클래스에 생성자와 소멸자를 추가해 생성자와 소멸자의 원리를 파악한다

*과제 9-1

1. Subject, Student Class에 Modify() 멤버함수 추가

*Subject::Modify() 소스코드

```
void Subject::Modify() { //Subject Class의 멤버변수 값 수정
	cout << "교과목명 : ";
	InputValue(m_name);
	cout << "학점 : ";
	InputValue(m_hakjum);
	cout << "등급  : ";
	InputValue(m_grade);

}
```

*Student::Modify() 소스코드

```
void Student::Modify() {
	Subject c_Sub;
	string Type;
	cout << endl << endl;
	cout << "수정(학생정보/과목정보/모두) : ";
	InputValue(Type);
	cout << endl;

	if (Type == "학생정보") { //학생정보 수정
		cout << "<학생 정보 수정>" << endl;
		cout << "이름 : ";
		InputValue(m_name);
		cout << "학번 : ";
		InputValue(m_hakbun);
		cout << endl;
```

```
	}
	else if (Type == "과목정보") { //과목 정보 수정
		int i;
		for (i = 0; i < m_subnum; i++) {
			m_sub[i].Modify();
		}
		cout << endl;
	}
	else { //모두 수정
		cout << "<학생 정보 수정>" << endl;
		cout << "이름 : ";
		InputValue(m_name);
		cout << "학번 : ";
		InputValue(m_hakbun);
		cout << endl;

		cout << "<과목 정보 수정>" << endl;
		for (int i = 0; i < m_subnum; i++)
		{
			m_sub[i].Modify();

		}
	}

}
```

Student Class의 Modify()의 경우, 과목 정보 수정할 때 Subject Class의 Modify() 함수를 내부적으로 사용했다. 사용자가 입력하는 학생정보, 과목정보, 모두 문자열에 따라 수정하는 부분이 다르다.

2. Subject Class의 생성자

*Subject Class 내부

```
class Subject { //과목 클래스
protected:
(중략)
public:
(중략)
	//생성자 소멸자
	Subject();
```

```
        Subject(string name, int hakjum, string grade);
        Subject(const Subject& sub);
        ~Subject();
}
```

성적처리 프로그램#2에서는 멤버변수를 초기화하는 디폴트 생성자인 Subject()와 입력받은 정보를 멤버변수에 할당하는 Subject(string name, int hakjum, string grade), 소멸자인 ~Subject() 사용한다. 이번 프로그램에서 ~Subject()는 아무런 역할을 하지 않는다.

***Student Class 내부**

```
class Student { //과목 클래스
protected:
(중략)
public:
(중략)
        //생성자 소멸자
        Student();
        Student(string name, int hakbun, int subnum, Subject *sub);
        Student(const Student& std);
        ~Student();

}
```

성적처리 프로그램#2에서는 멤버변수를 초기화 하는 Student()와 입력 받은 정보를 멤버변수에 입력해주는 Student(string name, int hakbun, int subnum, Subject *sub), 그리고 동적할당 해제를 해주는, 소멸자 ~Student()를 사용한다.

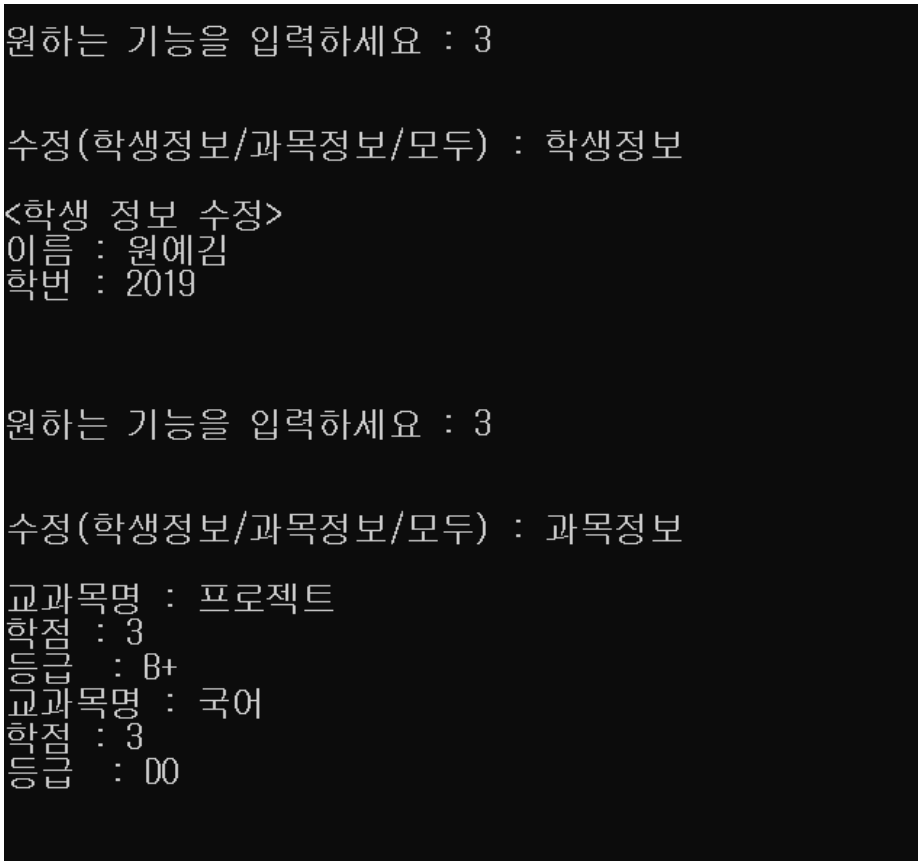
3. 성적처리 프로그램#2 실행 화면

```
=====메뉴=====
1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 정보 수정
4. 프로그램 종료


원하는 기능을 입력하세요 : 1

*1 번째 학생 이름과 학번을 입력하세요.
이름 : 김예원
학번 : 2018038032
수강한 과목 수를 입력 : 2

* 수강한 과목 2개와 각 교과목명, 과목학점, 과목등급을 입력하세요
교과목명 : 기초프로젝트
과목학점수 : 3
과목등급<A+ ~ F> : A+


* 수강한 과목 2개와 각 교과목명, 과목학점, 과목등급을 입력하세요
교과목명 : 액션잉글리시
과목학점수 : 3
과목등급<A+ ~ F> : C+
```

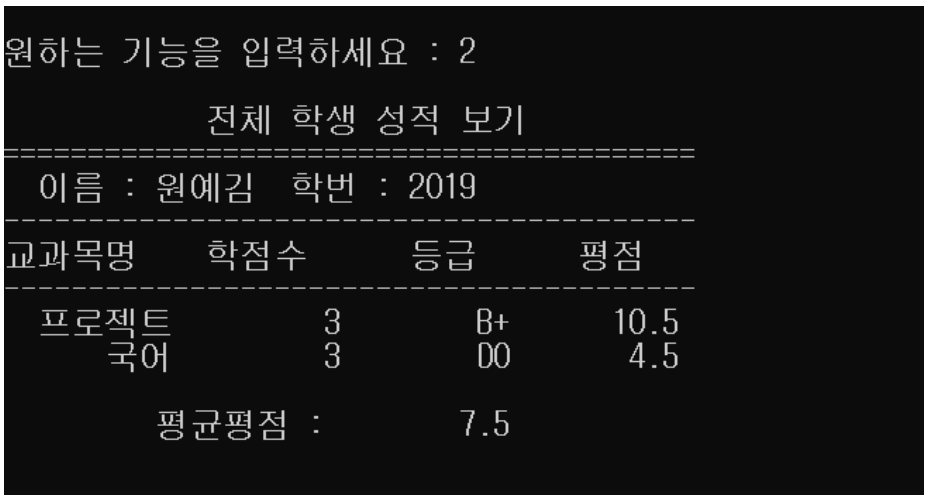
한 명의 학생에게 두 과목에 대한 정보를 입력 받는다.

```
원하는 기능을 입력하세요 : 2

            전체 학생 성적 보기
=========================================
  이름 : 김예원  학번 : 2018038032
-----------------------------------------
교과목명     학점수       등급       평점
-----------------------------------------
기초프로젝트          3         A+       13.5
액션잉글리시          3         C+        7.5

         평균평점 :        10.5
```

입력 받은 정보를 확인한다.



학생정보(이름, 학번)와 입력 받은 2개의 과목정보(교과목명, 학점, 등급)를 수정한다.



수정한 과목정보와 학생 정보를 확인한다.

```
원하는 기능을 입력하세요 : 3


수정(학생정보/과목정보/모두) : 모두

<학생 정보 수정>
이름 : 진예원
학번 : 20182018

<과목 정보 수정>
교과목명 : 영어
학점 : 2
등급  : C+
교과목명 : 수학
학점 : 3
등급  : F
```

학생의 모든 정보를 한번에 수정한다.

```
원하는 기능을 입력하세요 : 2

           전체 학생 성적 보기
========================================
  이름 : 진예원  학번 : 20182018
----------------------------------------
교과목명    학점수      등급      평점
----------------------------------------
     영어         2        C+         5
     수학         3         F         0

        평균평점 :       2.5


원하는 기능을 입력하세요 : 4

C:₩Users₩김예원₩source₩repos₩객프과제9₩Debug₩객프과제9.e
xe(10700 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

수정에 성공한 것을 확인하고, 소멸자가 호출되어 정상적으로 종료된 것을 확인한다.

*과제 9-2

1. 생성자와 소멸자 테스트

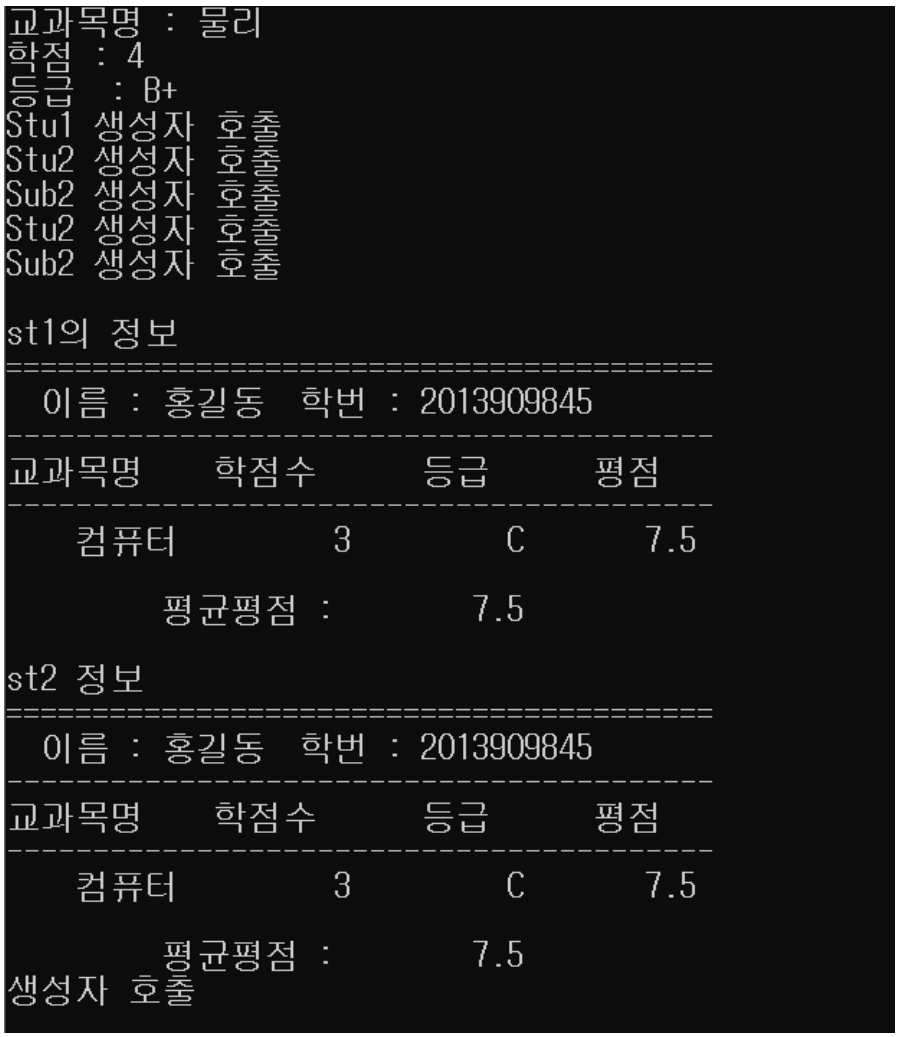
*소스코드

```
void main() {
	Subject sub1("컴퓨터", 3, "C");
	Subject sub2(sub1);
	cout << "₩n" << "sub1 정보" << "₩n";
	sub1.PrintData();
	cout << "₩n" << "sub2 정보" << "₩n";
	sub2.PrintData();
	sub2.Modify();

	Student st1;
	Student st2("홍길동", 2013909845,1, &sub1);
	Student st3("김성령", 2015909845, 1, &sub2);

	st1 = st2;
	cout << "₩n" << "st1의 정보" << "₩n";
	st1.PrintData();

	cout << "₩n" << "st2 정보" << "₩n";
	st2.PrintData();
	st2.Modify();
	cout << "₩n" << "수정된 st2 정보" << "₩n";
	st2.PrintData();
	st1.PrintData();

	cout << "₩n" << "st3 정보" << "₩n";
	st3.PrintData();

}
```

***첫 실행화면**

`Subject sub1("컴퓨터", 3, "C"); Subject sub2(sub1);`로 호출한 생성자가 호출되었음을 확인할 수 있다.

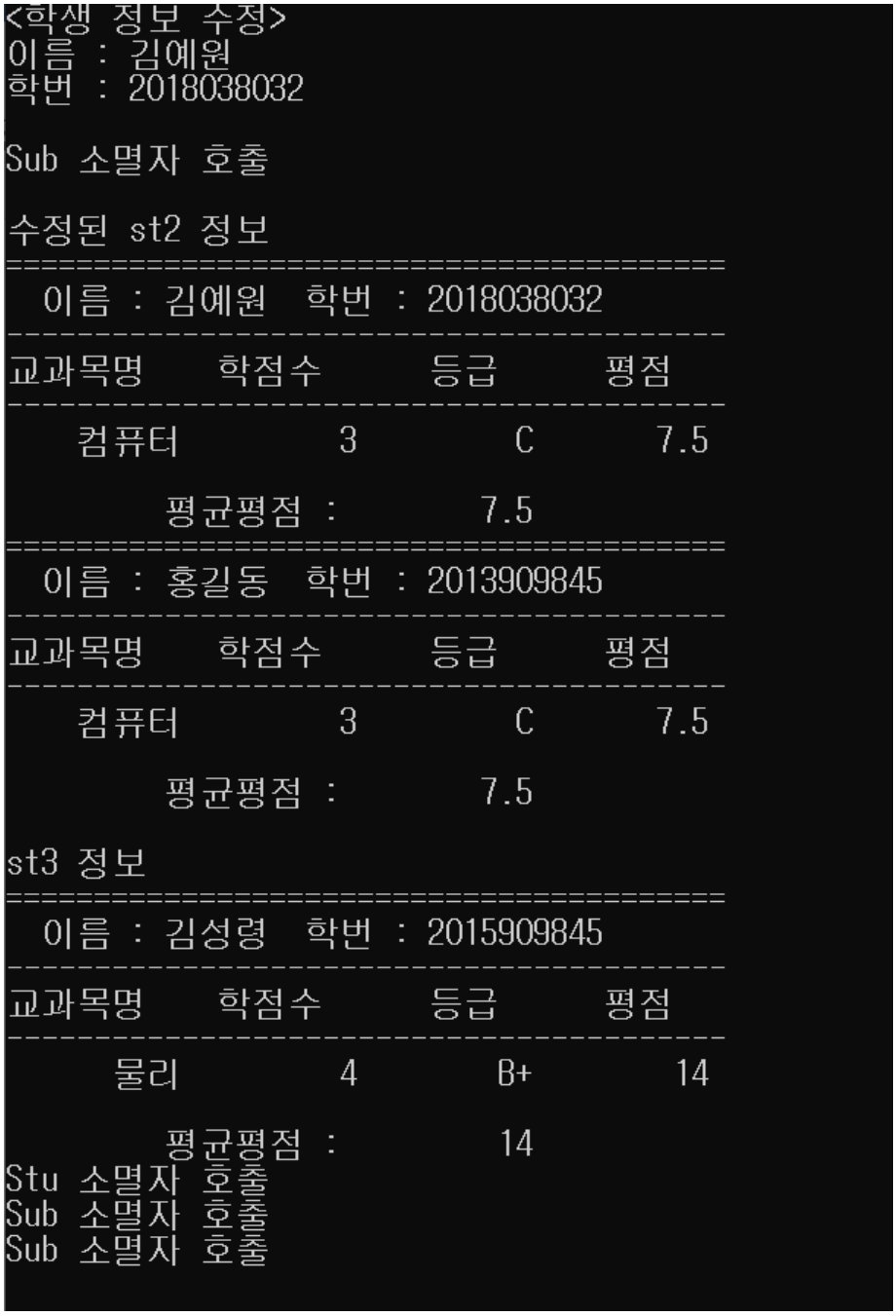
***정보를 입력한 이후**



Student의 생성자들이 호출됨을 확인할 수 있고, st1과 st2의 복사가 성공적으로

이루어졌음을 확인할 수 있다.

***Modify() 함수로 학생 정보를 수정한 결과**



수정에 앞서 Sub 소멸자가 나타났고 수정된 st2 정보와 st1에 복사된 st2의 이전 정보, st3의 정보가 출력됨을 확인할 수 있다. 또한 Stu 소멸자가 동적할당을 해제하고 두 과목 정보를 소멸하는 Sub 소멸자가 호출됨을 알 수 있다.

***Modify()함수로 과목정보를 출력한 결과**

```
수정(학생정보/과목정보/모두) : 과목정보

교과목명 : 영어
학점 : 3
등급  : A+

Sub 소멸자 호출

수정된 st2 정보
=========================================
  이름 : 홍길동  학번 : 2013909845
-----------------------------------------
교과목명    학점수      등급      평점
-----------------------------------------
     영어         3        A+      13.5

        평균평점 :       13.5
=========================================
  이름 : 홍길동  학번 : 2013909845
-----------------------------------------
교과목명    학점수      등급      평점
-----------------------------------------
     영어         3        A+      13.5

        평균평점 :       13.5

st3 정보
=========================================
  이름 : 김성령  학번 : 2015909845
-----------------------------------------
교과목명    학점수      등급      평점
-----------------------------------------
     물리         4        B+        14

        평균평점 :         14
Stu 소멸자 호출
Sub 소멸자 호출
Sub 소멸자 호출
```

과목정보가 수정된 st1과 st2, st3가 출력됨을 확인할 수 있다.

***Modify() 함수로 모두 정보를 수정한 결과**

```
수정된 st2 정보
========================================
  이름 : 김예원  학번 : 2018038032
----------------------------------------
교과목명     학점수       등급       평점
----------------------------------------
      영어          3         A+       13.5

         평균평점 :        13.5
========================================
  이름 : 홍길동  학번 : 2013909845
----------------------------------------
교과목명     학점수       등급       평점
----------------------------------------
      영어          3         A+       13.5

         평균평점 :        13.5

st3 정보
========================================
  이름 : 김성령  학번 : 2015909845
----------------------------------------
교과목명     학점수       등급       평점
----------------------------------------
      물리          4         B+         14

         평균평점 :          14
Stu 소멸자 호출
Sub 소멸자 호출
Sub 소멸자 호출
```

과목정보와 학생정보가 수정됨을 알 수 있다.

느낀점

생성자와 소멸자가 사실 너무 생소한 개념이었다. 불러내지 않아도 스스로 호출된다는 점이 너무 신기하게 다가왔다. 처음에는 반드시 모두 호출 해야하는 줄 알고 디폴트 생성자를 호출하기 위해 갖은 노력들을 했었다. 하지만 어느 방법을 시도하던 오류 메시지가 나타났다. 슬슬 약이 오를 때쯤, 선배한테 물어보게 되었는데, 디폴트 생성자와 소멸자는 스스로 호출되니 프로그래머가 호출할 필요 없다는 것을 알게 되었다. 약이 올랐지만 너무 신기하고 좋은 기능이라는 생각이 들었다. 또한 오버로딩이 가능하다는 점이 생성자로써 명시적으로 표현될 수 있다는 점이 좋았다. 또한 깊은 복사와 얕은 복사에 대해서 공부하면서, 처음에는 깊은 복사가 어떤 것인지 잘 감이 오지 않았다. 깊은 복사는 메모리 공간을 따로 확보해줘서 복사된 것들 사이의 관련성을 떨어뜨려준다는 개념이 확실히 세워진 뒤에야 올바르게 코딩 시도를 할 수 있었다.

객체지향에 대해 배우면 배울수록 그동안 배운 절차지향 언어인 C언어 보다 간편하고 단순한 언어라는 생각이 드는 것 같다. 처음에는 왜 나는 이렇게 까지 밖에 못할까라는 생각을 하는 일이 종종 있었는데, 객체지향이 재밌게 느껴질수록 그런 생각보다는 재밌어서 좋다는 생각이 남는다.