



# Object-Oriented Programming Report

Assignment 3-3

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2020202031
Name	Jaehyun Kim

## Program 1

### □ 문제 설명

두 item 의 가격과 할인율을 입력 받고, 할인율을 적용한 두 item 의 가격비교와 절약한 비용을 출력하는 프로그램을 작성하기 위해 필요한 class 들을 정의하는 문제입니다.

먼저 가격 정보를 담고 있는 Sale class 와 이를 상속받는 DiscountSale class 를 정의합니다. Sale class 에서 가장 중요한 메소드는 GetPrice()로 이는 상속 class 에서 private 변수 Price 를 참고할 수 있도록 해줍니다.

GetPrice 를 통해 DiscountSale class 에서 Bill(), Savings() 등 초기가격이 필요한 메소드들을 정의할 수 있습니다.

### □ 결과 화면

```

=====
Price Compare Program
=====
Insert item1 price: 15
Insert item2 price: 20

-----

Insert discount percent: 20
-----

Result:
Discount price of item1 is cheaper.
Saving discount price is $3

```

15 와 20 의 20 퍼센트는 각각 3, 4 다. 15 와 20 의 할인된 금액은 12, 16 이 된다. 그러므로 item1 의 가격이 더 싸다고 할 수 있고, item1 의 할인된 금액 3 이 함께 출력된다.

```

=====
Price Compare Program
=====
Insert item1 price: 2
Insert item2 price: 4

-----

Insert discount percent: 50
-----

Result:
Discount price of item1 is cheaper.
Saving discount price is $1

```

2 와 4 의 50 퍼센트는 각각 1, 2 다. 2 와 4 의 할인된 금액은 1, 2 가 된다. 그러므로 item1 의 가격이 더 싸다고 할 수 있고, item1 의 할인된 금액 1 이 함께 출력된다.

## □ 고찰

DiscountSale class 의 operater<를 정의 하던 과정에서 매개변수로 const DiscountSale& second 를 선언했습니다. second 의 price 를 참고하기 위해 GetPrice()메소드를 호출했는데 에러가 뜨는 것을 확인했고, 알아본 결과 class 에 const 가 붙으면 const 로 선언된 메소드만을 호출할 수 있다는 것을 알게되어, GetPrice() 메소드에 const 를 추가해 이를 가능하도록 만들었습니다.



## Program 2

### □ 문제 설명

사람의 이름과 나이 정보를 저장하고 있는 Person 객체를 상속 받는 Student, Professor class 를 정의하는 문제입니다. Student 는 Person 을 상속 받으므로 이름, 나이 정보가 존재하고, 추가로 학번, 전공, 학년 정보도 존재합니다. Professor 또한 Person 을 상속 받으므로 이름, 나이 정보가 존재하며 추가로 학번, 전공 정보가 존재합니다. 이름과 나이는 Person 에 정의된 메소드들로 초기화 및 참조했습니다. virtual void say() = 0; 으로 say() 메소드가 순수 가상함수로 선언됐기 때문에 Student 와 Professor class 에도 각각 say()메소드를 정의해줬습니다.

### □ 결과 화면

```

=====Insert Professor Info=====
name: Dongkyu Sim
age: 35
ProfessorNum: 1988050512
major: Computer Science

=====Insert Student Info=====
name: Jaehyun Kim
age: 23
StudentNum: 2020202031
major: Computer Science
Grade: 2

=====Professor Info=====
I'm a professor of KW University.
My name is Dongkyu Sim.
I'm 35 years old
My Professor Number is 1988050512
I'm majoring in Computer Science.

=====Student Info=====
I'm a student of KW University.
My name is Jaehyun Kim.
I'm 23 years old and I'm a Sophomore.
My Professor Number is 2020202031
I'm majoring in Computer Science.

```

교수와 학생의 정보를 입력하면 Say()메소드를 통해 정보들이 잘 출력되는 것을 확인할 수 있습니다. 특히 학생의 정보의 경우 학년에 따라 불리는 호칭이 다르므로 switch 문을 통해 각 학년 별로 다른 출력 값을 가지도록 했습니다.

## □ 고찰

cin >> 함수는 표준입력으로 받은 입력에서 개행문자와 공백을 제외하고 나머지만 인식합니다. 하지만 cin.getline()은 cin 에서 건너뛴 공백, 개행문자들을 그대로 읽어들이 원하는 입력 값을 입력하지 못하고 공백, 개행문자가 자동으로 입력되는 상황이 연출됩니다. 이에 cin.ignore()을 통해 버퍼에 남아있는 공백, 개행문자를 비워 cin.getline 이 올바르게 입력되도록 했습니다.

## Program 3

## □ 문제 설명

3x3 행렬에 대하여 연산자 +=, -=, \*를 재정의 하는 문제입니다. 연산자 오버로딩을 통해 행렬에 대한 연산자로 재정의 해줬습니다. +=와 -=의 경우 행렬의 각 요소를 1 대 1로 덧셈, 뺄셈을 진행하고 연산자 실행주체가 되는 객체에 그 결과가 저장되며 반환 값으로 그 객체를 반환했습니다. \*의 경우 행렬끼리의 곱셈 정의대로 곱셈연산을 진행하였고 임시객체를 반환하여 다른 객체에서 연산결과를 저장하거나 임시객체를 통해 print()메소드도 출력할 수 있도록 했습니다.

## □ 결과 화면

```
=====Input matrix1=====
9 7 5
1 2 7
0 8 5

=====Input matrix2=====
6 6 9
3 7 2
9 6 1

=====matrix1+=matrix2=====
15 13 14
4 9 9
9 14 6

=====matrix1-=matrix2=====
9 7 5
1 2 7
0 8 5

=====matrix1*matrix2=====
120 133 100
75 62 20
69 86 21
```

```
=====Input matrix1=====
2.3 5.1 1.6
6.6 8.5 2.3
1.2 1.5 2.1

=====Input matrix2=====
5.4 0.3 5.5
1.5 2.2 0.9
6.6 7.7 9.3

=====matrix1+=matrix2=====
7.7 5.4 7.1
8.1 10.7 3.2
7.8 9.2 11.4

=====matrix1-=matrix2=====
2.3 5.1 1.6
6.6 8.5 2.3
1.2 1.5 2.1

=====matrix1*matrix2=====
30.63 24.23 32.12
63.57 38.39 65.34
22.59 19.83 27.48
```

matrix1 에 matrix2 를 덧셈한 후 다시 matrix2 를 뺄셈하므로 matrix1 이 초기값으로 돌아가는 것을 확인할 수 있다.

$$\begin{pmatrix} 9 & 7 & 5 \\ 1 & 2 & 7 \\ 0 & 8 & 5 \end{pmatrix} + \begin{pmatrix} 6 & 6 & 9 \\ 3 & 7 & 2 \\ 9 & 6 & 1 \end{pmatrix} = \begin{pmatrix} 15 & 13 & 14 \\ 4 & 9 & 9 \\ 9 & 14 & 6 \end{pmatrix}$$
$$\begin{pmatrix} 15 & 13 & 14 \\ 4 & 9 & 9 \\ 9 & 14 & 6 \end{pmatrix} - \begin{pmatrix} 6 & 6 & 9 \\ 3 & 7 & 2 \\ 9 & 6 & 1 \end{pmatrix} = \begin{pmatrix} 9 & 7 & 5 \\ 1 & 2 & 7 \\ 0 & 8 & 5 \end{pmatrix}$$
$$\begin{pmatrix} 2.3 & 5.1 & 1.6 \\ 6.6 & 8.5 & 2.3 \\ 1.2 & 1.5 & 2.1 \end{pmatrix} + \begin{pmatrix} 5.4 & 0.3 & 5.5 \\ 1.5 & 2.2 & 0.9 \\ 6.6 & 7.7 & 9.3 \end{pmatrix} = \begin{pmatrix} 7.7 & 5.4 & 7.1 \\ 8.1 & 10.7 & 3.2 \\ 7.8 & 9.2 & 11.4 \end{pmatrix}$$
$$\begin{pmatrix} 7.7 & 5.4 & 7.1 \\ 8.1 & 10.7 & 3.2 \\ 7.8 & 9.2 & 11.4 \end{pmatrix} - \begin{pmatrix} 5.4 & 0.3 & 5.5 \\ 1.5 & 2.2 & 0.9 \\ 6.6 & 7.7 & 9.3 \end{pmatrix} = \begin{pmatrix} 2.3 & 5.1 & 1.6 \\ 6.6 & 8.5 & 2.3 \\ 1.2 & 1.5 & 2.1 \end{pmatrix}$$
$$\begin{pmatrix} 2.3 & 5.1 & 1.6 \\ 6.6 & 8.5 & 2.3 \\ 1.2 & 1.5 & 2.1 \end{pmatrix} * \begin{pmatrix} 5.4 & 0.3 & 5.5 \\ 1.5 & 2.2 & 0.9 \\ 6.6 & 7.7 & 9.3 \end{pmatrix} = \begin{pmatrix} 30.63 & 24.23 & 32.12 \\ 63.57 & 38.39 & 65.34 \\ 22.59 & 19.83 & 27.48 \end{pmatrix}$$

행렬의 곱셈 또한 결과가 잘 출력됨을 확인할 수 있다.

## □ 고찰

함수에서 class 를 반환형으로 하여 지역변수로 선언된 객체를 반환하면 함수가 종료되면서 지역변수 객체 또한 동시에 해제되어 참조를 하지 못하는 것으로 알고 있었는데 이번 과제에서 operator\*을 정의하면서 class 를 반환형으로 지역변수 객체를 반환하면 임시객체라는 것이 생성되어 함수를 호출한 한 줄이 끝나기 전까지는 임시객체가 유효하게 작동한다는 것을 알게 되었습니다. 이를 통해 임시객체로 print()메소드를 실행시켜 \*연산의 결과값을 출력했습니다.

## Program 4

### □ 문제 설명

학생들의 수학 영어 과학 점수 평균을 저장하는 노드 Score class 를 만들고 이를 관리하는 StudentScoreList 를 정의하는 문제입니다. StudentScoreList::Insert(Score\* pScore)의 경우 Score 노드를 오름차순으로 삽입해야하기 때문에 기존에 존재하던 노드들과 pScore 를 비교하며 적절한 위치에 pScore 를 삽입했습니다. 이 때 다양한 예외사항들을 처리하기 위해 pScore 가 들어갈 위치가 제일 앞일 경우, 제일 뒤일 경우, 중간일 경우 3 가지로 나누어 삽입을 진행했습니다.

PrintList 의 경우 오름차순으로 출력할 땐 pHead 부터 pTail 까지 차례로 출력했고

내림차순으로 출력할 땐 pTail 부터 pHead 까지 차례로 출력했습니다.

## □ 결과 화면

```
Input the number of students: 10

=====Input Math Score=====
->12 45 67 95 42 77 99 34 53 100
average: 62.4

=====Input English Score=====
->56 36 27 86 45 42 57 79 77 99
average: 60.4

=====Input Science Score=====
->18 82 84 91 23 94 49 19 28 100
average: 58.8
Ascending or Descending (1 or 0)1
58.8 60.4 62.4
```

처음에 학생의 수를 입력 받습니다. 수학 영어 과학 점수를 학생 수만큼 입력 받고 평균을 내서 Score 노드를 생성하고 이를 StudentScoreList 에 Insert 해줍니다. 오름차순으로 출력하기 위해 1 을 입력했고 그 결과 평균들이 오름차순으로 출력되는 것을 확인할 수 있습니다.

```
Input the number of students: 10

=====Input Math Score=====
->28 83 85 96 37 26 95 49 99 100
average: 69.8


=====Input English Score=====
->14 64 93 38 84 85 76 73 27 95
average: 64.9

=====Input Science Score=====
->18 98 95 96 94 38 28 39 69 99
average: 67.4
Ascending or Descending (1 or 0)0
69.8 67.4 64.9
```

처음에 학생의 수를 입력 받습니다. 수학 영어 과학 점수를 학생 수만큼 입력 받고 평균을 내서 Score 노드를 생성하고 이를 StudentScoreList 에 Insert 해줍니다. 내림차순으로 출력하기 위해 0 을 입력했고 그 결과 평균들이 내림차순으로 출력되는 것을 확인할 수 있습니다.

## □ 고찰





단순 연결리스트의 경우 노드를 탐색할 때 한 방향으로만 탐색이 가능했는데 양방향 연결리스트를 사용할 경우 양쪽 끝에서 원하는 방향으로 탐색이 가능한 점이 매력적이었던 것 같습니다. 양 방향으로 탐색 가능하다는 점 덕분에 오름차순으로 정렬된 리스트를 오름차순 혹은 내림차순으로 출력하는데 한결 수월하게 진행할 수 있었던 것 같습니다.