# 2020-11-04

# 20190269

박재우

# Week09

https://github.com/jwoo9928/-Object-oriented-design/tree/master/week9

과제 1

- 코드 및 실행화면

```cpp
#include <iostream>

class MyData{
 int number;
 std::string strNumber;
public:
 MyData(int data, std::string str): number(data), strNumber(str){}
//Operator conversion
 operator int(){return number;}
 operator std::string(){return strNumber;}

//Unary operator
 int operator++(int){ //postfix operation (indicated by dummy 'int')
  return number++;
  //?1?¿
 }
 int operator++(){ //prefix operation
   return ++number;
 }

 friend std::ostream& operator<<(std::ostream&, MyData &);
};

// non-member operator<< function
std::ostream& operator<<(std::ostream& os, MyData & md){
  return os << "This number is: " << md.strNumber << "\n";
}

int main(){
 MyData mydata(1, "one");

 std::string strNum = mydata;
 int intNum = mydata;

 std::cout << strNum << std::endl; // one
 std::cout << intNum << std::endl; // 1
 std::cout << mydata++ << std::endl; // 1
 std::cout << ++mydata << std::endl; // 3
 std::cout << mydata;

}
```

```
→  week9 git:(master) ✗ g++ homework_09_01.cpp
→  week9 git:(master) ✗ ./a.out
one
1
1
3
This number is: one
```

- 과제 수행 과정

  별 어려운 것 없이 operator++을 각각 원하는 결과에 맞게 연산뒤에 return을 시켜준다.

- 결과 분석

  과제에서 원하는 결과와 같이 원하는대로 결과가 제대로 출력되었다.

- 새로 알게 된 점

  이전에 배웠던 연산자 오버로딩을 사용해서 복습하는 과제였다.

과제 2

- 코드 및 실행화면

```
#include <iostream>
#include <string>

class Employee {
protected:
    std::string name;
    int age;

public:
    Employee(std::string name, int age) : name(name), age(age) {}

    virtual void showInfo() { std::cout << "Name:" << name << ", Age: " << age << std::endl; }
};

class Manager : public Employee {
    int managerBonus;
public:
    Manager(int managerBonus, std::string name, int age) : managerBonus(managerBonus), Employee(name, age) {}

    void showInfo() { std::cout << "Manager Name:" << name << ", Age: " << age << ", managerBonus:" << managerBonus << std::endl; }
};

class Intern : public Employee {
    std::string majorName;
public:
    Intern(std::string major, std::string name, int age) : majorName(major), Employee(name, age) {}

    void showInfo() { std::cout << "Intern Name:" << name << ", Age: " << age << ", Major:" << majorName << std::endl; }
};

class Janitor : public Employee {
    int salary;
public:
    Janitor(int salary, std::string name, int age) : salary(salary), Employee(name, age) {}

    void showInfo() { std::cout << "Janitor Name:" << name << ", Age: " << age << ", Salary:" << salary << std::endl; }
};

int main() {

    Employee** employeelist = new Employee * [6];

    employeelist[0] = new Manager(200, "James", 33);
    employeelist[1] = new Manager(150, "Chulsoo", 50);

    employeelist[2] = new Intern("security", "Minsu", 24);
    employeelist[3] = new Intern("HCI", "Yong", 19);

    employeelist[4] = new Janitor(100, "Black", 90);
    employeelist[5] = new Janitor(200, "White", 100);

    employeelist[0]->showInfo();
```

```
-       employeelist[1]->showInfo();
-
-       employeelist[2]->showInfo();
-       employeelist[3]->showInfo();
-
-       employeelist[4]->showInfo();
-       employeelist[5]->showInfo();
-
-       getchar();
-       return 0;
-   }
```

```
→  week9 git:(master) ✗ g++ homework_09_02.cpp
→  week9 git:(master) ✗ ./a.out
Manager Name:James, Age: 33, managerBonus:200
Manager Name:Chulsoo, Age: 50, managerBonus:150
Intern Name:Minsu, Age: 24, Major:security
Intern Name:Yong, Age: 19, Major:HCI
Janitor Name:Black, Age: 90, Salary:100
Janitor Name:White, Age: 100, Salary:200
```

- 과제 수행 과정

  부모 class의 함수를 virtual로 만들어 가상함수로 만들어주고

  자식 class에서 부모의 함수를 재정의 해주어 오버라이딩 해준다.

- 결과 분석

  역시 pdf의 예시대로 정확하게 부모클래스의 함수를 자식클래스에서 재정의를 통해 오버라이딩을 통해 원하는 출력결과가 나왔다.