2020-10-11

20190269

박재우

Week06

https://github.com/jwoo9928/-Object-oriented-design/tree/master/week6

과제 1

1. 코드 및 실행화면

```cpp
#include <iostream>

class My_cat {
    char* name;
    int weight;

public:
    My_cat();
    My_cat(const char* name);
    My_cat(const My_cat& cat);
    ~My_cat();

    //eat 0L0 0000
    My_cat &eat(const int n);

    void show_status() const;
};

My_cat::~My_cat() {     //Destructor
    if (name) delete[] name;
}

//eat 0L000 00000g000.
My_cat& My_cat::eat(const int n) {
    this->weight +=n;
    return *this;
}
```

```
➜  WEEK06_CODE g++ homework_06_01.cpp -std=c++11
➜  WEEK06_CODE ./a.out
 Weight : 10
 Weight : 14
 Weight : 18
 Weight : 33
```

2. 과제 수행 과정

Cat(4).cat(5).cat(6)처럼 계속해서 cat함수를 호출하기위해 eat의 return type을 My_cat 참조형 으로 한뒤,

Weight를 this로 호출하여 인자로 받은 n을 더해준다.

그후 this로 my_cat객체를 호출하여 에스터리스크를 통해 주소값이 아닌 객체를 리턴시킨다.

3. 결과분석

   Ppt에서 제시한 결과와 똑같이 출력된다.

   따라서 코드에 문제가 없는것으로 분석된다.

   또한 eat함수 사용시 생성자를 통해 재생성 되지 않고 참조하여 실행되는 것을 알 수 있다.

4. 새로 알게된점

   기존에 c언어처럼 리턴타입을 My_cat으로 할 경우 생성자를 총해 재생성되어

```
Copy constructor invocation !
```

   가 뜨게 되는데 참조를 통해 코드를 구현하면

   재생성 없이 만들 수 있다는 것을 알게 되었다.

과제 2

1. 코드 및 실행화면

   -완성된 training_02.cpp

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

class Animal {
private:
    std::string name;

public:
    Animal() {};
    Animal(std::string name) : name(name) {};

    void showName() {
        std::cout << "Name is " << name << std::endl;
    }

    //operator+
    Animal operator+(Animal &ref) {
        return Animal(name+ref.name);
    }

};

int main() {
```

```
    Animal cat("Nabi");
    cat.showName();
    Animal dog("Jindo");
    dog.showName();

    Animal catDog = dog + cat; //������ ��
    catDog.showName();

    dog.showName();

    getchar();
    return 0;
}
```

training_02.cpp 와 training_03.cpp 실행화면

```
➜  WEEK06_CODE g++ training_02.cpp -std=c++11
➜  WEEK06_CODE ./a.out
Name is Nabi
Name is Jindo
Name is JindoNabi
Name is Jindo
^[[A^[[A^[[A^[[A^C
➜  WEEK06_CODE g++ training_03.cpp -std=c++11
➜  WEEK06_CODE ./a.out
Name is Nabi
Name is Jindo
Name is JindoNabi
Name is JindoNabi
^C
➜  WEEK06_CODE
```

2. 과제 수행 과정

   연산자 오버로딩를 사용하여 객체끼리 더할경우
   객체의 name끼리 더하도록 코드를 제작했다.

3. 결과분석(02와 03의 출력이 다른 이유)

   02는 dog의 name과 cat의 name을 더해 새로운 Animal 객체를 생성하여 return을 해주어 catDog이라는 객체에 새로운 객체가 할당되었다. 따라서 dog의 name에는 아무런 변화가 없다.
   하지만 03는 dog의 name에 cat의 name을 더해준 뒤,

This.를 사용하여 dog객체 자기 자신을 반환했기에
catDog 객체에 dog 객체가 할당되고 dog객체 또한 더한값으로 변한다. 따라서 dog를 출력하면 jindoNabi가 출력된다.

4. 새로 알게된점
   연산자 오버로딩을 이용해서 객체끼리의 값을 따로 함수생성을 통하지 않고 계산을 할 수 있다는 것을 알게되었다.