객체지향설계 #Week05

제출일 : 20.10.04

학번 이름 : 201902721 유찬희

GitHub 주소 : https://github.com/HanCiHu/OOP

1. homework01.

```
18      static GlobalClass* instance(){
19          if (s_instance == NULL){
20              s_instance = new GlobalClass;
21          }
22          return s_instance;
23      }
24  };
```

<< instance 함수 코드

```
➜  week8 git:(master) ✗ g++ homework_08_01.cpp
➜  week8 git:(master) ✗ ./a.out
main: global_ptr is 0
foo: global_ptr is 1
bar: global_ptr is 2
```

<< 실행결과

- foo 함수와 bar 함수에서 instance함수 자체를 하나의 GlobalClass의 객체로 사용하고 있기 때문에 리턴값을 GlobalClass로 해서 함수를 작성하였다.
- 만약, GlobalClass안의 instance가 없다면 instance함수에서 null값을 리턴해주게 되고 그렇게 되면 null값을 참조하여 프로그램이 비정상적으로 종료될수 있기 때문에 instance가 null인경우 예외처리를 하였다.

2. homework02.

```cpp
#include<iostream>

class Strategy{
public:
    virtual int doOperation(int num1, int num2) = 0;
};

class OperationAdd : public Strategy{
    public:
        int doOperation(int num1, int num2) override{
            return num1 + num2;
        }
};

class OperationMultiply : public Strategy{
    public:
        int doOperation(int num1, int num2) override{
            return num1 * num2;
        }
};

class OperationSubstract : public Strategy{
    public:
        int doOperation(int num1, int num2) override {
            return num1 - num2;
        }
};

class Context{
    private:
        Strategy *strategy = 0;

    public:
        Context(Strategy *strategy){
            this->strategy = strategy;
        }
        int executeStrategy(int num1, int num2){
            return strategy->doOperation(num1,num2);
        }
};
```

<< Strategy, Operation, Context 클래스

```cpp
int main(){
    Context *context = new Context(new OperationAdd());
    std::cout << context->executeStrategy(10,5) << std::endl;
    context = new Context(new OperationMultiply());
    std::cout << context->executeStrategy(10,5) << std::endl;
    context = new Context(new OperationSubstract());
    std::cout << context->executeStrategy(10,5) << std::endl;

}
```

<< main 함수

```
→  week8 git:(master) ✗ g++ homework_08_02.cpp -std=c++17
→  week8 git:(master) ✗ ./a.out
15
50
5
→  week8 git:(master) ✗
```

<< 실행 결과

- 자바 파일에서 Strategy를 가상 클래스로 선언하고 OperationAdd, OperationMultiply, OperationSubtract가 각각 상속받았으므로 그 부분부터 구현하였다.
- 자바 파일의 코드처럼 메인함수에서 context 생성자의 매개변수 안에 operation객체를 넣어주기 위해 context클래스안의 생성자를 따로 만들어주었다.
- New operation()을 하면 포인터형으로 리턴이 되기 때문에 Context안의 strategy변수는 포인터형으로 선언해주었다.
- 메인함수는 자바 파일의 코드를 그대로 C++식으로 옮겨주었다.