# Facade 패턴

## 타입

Structural Pattern

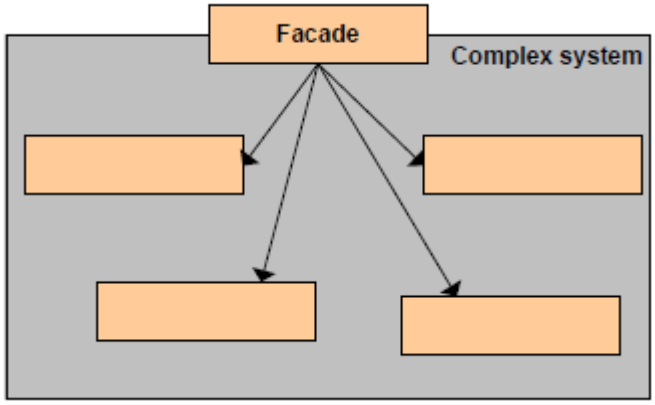
## 문제

클래스가 많고 인터페이스가 복잡하다.

## 해결

여러 개의 클래스 또는 인터페이스를 순차적으로 호출하는 클래스를 만든다.

## 클래스 다이어그램




## 예제

facade.cpp

```
/*Facade is one of the easiest patterns I think... And this is very
simple example.

Imagine you set up a smart house where everything is on remote. So to
turn the lights on you push lights on button - And same for TV,
AC, Alarm, Music, etc...

When you leave a house you would need to push a 100 buttons to make
sure everything is off and are good to go which could be little
annoying if you are lazy like me
```

```
so I defined a Facade for leaving and coming back. (Facade functions
represent buttons...) So when I come and leave I just make one
call and it takes care of everything...

*/

#include <string>
#include <iostream>

using namespace std;

class Alarm
{
public:
    void alarmOn()
    {
        cout<<"Alarm is on and house is secured"<<endl;
    }

    void alarmOff()
    {
        cout<<"Alarm is off and you can go into the house"<<endl;
    }
};

class Ac
{
public:
    void acOn()
    {
        cout<<"Ac is on"<<endl;
    }

    void acOff()
    {
        cout<<"AC is off"<<endl;
    }
};

class Tv
{
public:
    void tvOn()
    {
        cout<<"Tv is on"<<endl;
    }

    void tvOff()
    {
        cout<<"TV is off"<<endl;
    }
```

```
};

class HouseFacade
{
    Alarm alarm;
    Ac ac;
    Tv tv;

public:

    HouseFacade(){}

    void goToWork()
    {
        ac.acOff();
        tv.tvOff();
        alarm.alarmOn();
    }

    void comeHome()
    {
        alarm.alarmOff();
        ac.acOn();
        tv.tvOn();
    }

};

int main()
{
   HouseFacade hf;

   //Rather than calling 100 different on and off functions thanks to
facade I only have 2 functions...
   hf.goToWork();
   hf.comeHome();
}
```

# 참고

http://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/Design_Patterns#Facade