Observer 패턴

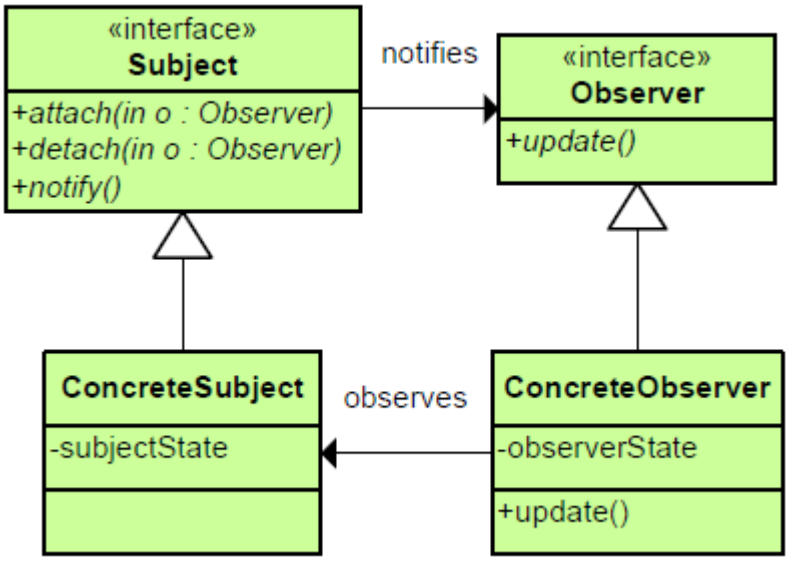
타입

Behavioral Pattern

문제

해결

클래스 다이어그램



예제

observer.cpp

```
#include <list>
#include <algorithm>
#include <iostream>
using namespace std;

// The Abstract Observer
class ObserverBoardInterface
{
public:
    virtual void update(float a,float b,float c) = 0;
```

```
};

// Abstract Interface for Displays
class DisplayBoardInterface
{
public:
    virtual void show() = 0;
};

// The Abstract Subject
class WeatherDataInterface
{
public:
    virtual void registerOb(ObserverBoardInterface* ob) = 0;
    virtual void removeOb(ObserverBoardInterface* ob) = 0;
    virtual void notifyOb() = 0;
};

// The Concrete Subject
class ParaWeatherData: public WeatherDataInterface
{
public:
    void SensorDataChange(float a,float b,float c)
    {
        m_humidity = a;
        m_temperature = b;
        m_pressure = c;
        notifyOb();
    }

    void registerOb(ObserverBoardInterface* ob)
    {
        m_obs.push_back(ob);
    }

    void removeOb(ObserverBoardInterface* ob)
    {
        m_obs.remove(ob);
    }
protected:
    void notifyOb()
    {
        list<ObserverBoardInterface*>::iterator pos = m_obs.begin();
        while (pos != m_obs.end())
        {
            ((ObserverBoardInterface*
)(*pos))->update(m_humidity,m_temperature,m_pressure);
            (dynamic_cast<DisplayBoardInterface*>(*pos))->show();
            ++pos;
        }
```

```
    }

private:
    float        m_humidity;
    float        m_temperature;
    float        m_pressure;
    list<ObserverBoardInterface* > m_obs;
};

// A Concrete Observer
class CurrentConditionBoard : public ObserverBoardInterface, public
DisplayBoardInterface
{
public:
    CurrentConditionBoard(ParaWeatherData& a):m_data(a)
    {
        m_data.registerOb(this);
    }
    void show()
    {
        cout<<"_____CurrentConditionBoard_____"<<endl;
        cout<<"humidity: "<<m_h<<endl;
        cout<<"temperature: "<<m_t<<endl;
        cout<<"pressure: "<<m_p<<endl;
        cout<<"_______________________________"<<endl;
    }

    void update(float h, float t, float p)
    {
        m_h = h;
        m_t = t;
        m_p = p;
    }

private:
    float m_h;
    float m_t;
    float m_p;
    ParaWeatherData& m_data;
};

// A Concrete Observer
class StatisticBoard : public ObserverBoardInterface, public
DisplayBoardInterface
{
public:
    StatisticBoard(ParaWeatherData&
a):m_maxt(-1000),m_mint(1000),m_avet(0),m_count(0),m_data(a)
    {
        m_data.registerOb(this);
    }
```

```
    void show()
    {
        cout<<"_______StatisticBoard________"<<endl;
        cout<<"lowest  temperature: "<<m_mint<<endl;
        cout<<"highest temperature: "<<m_maxt<<endl;
        cout<<"average temperature: "<<m_avet<<endl;
        cout<<"_______________________________"<<endl;
    }

    void update(float h, float t, float p)
    {
        ++m_count;
        if (t>m_maxt)
        {
            m_maxt = t;
        }
        if (t<m_mint)
        {
            m_mint = t;
        }
        m_avet = (m_avet * (m_count-1) + t)/m_count;
    }

private:
    float m_maxt;
    float  m_mint;
    float m_avet;
    int m_count;
    ParaWeatherData& m_data;
};


int main(int argc, char *argv[])
{

    ParaWeatherData * wdata = new ParaWeatherData;
    CurrentConditionBoard* currentB = new
CurrentConditionBoard(*wdata);
    StatisticBoard* statisticB = new StatisticBoard(*wdata);

    wdata->SensorDataChange(10.2, 28.2, 1001);
    wdata->SensorDataChange(12, 30.12, 1003);
    wdata->SensorDataChange(10.2, 26, 806);
    wdata->SensorDataChange(10.3, 35.9, 900);

    wdata->removeOb(currentB);

    wdata->SensorDataChange(100, 40, 1900);
```

```
    delete statisticB;
    delete currentB;
    delete wdata;

    return 0;
}
```

참고

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

From:
http://obg.co.kr/doku/ - **OBG WiKi**

Permanent link:
http://obg.co.kr/doku/doku.php?id=programming:design_pattern:observer

Last update: **2020/11/29 14:09**