Observer模式：重用，解耦合。又称发布-订阅模式。订阅的网络杂志、话费清单的实现类似于Observer模式。

简单的Observer模式的例子：该例子模拟移动公司（发布者）向消费者（观察者）发送话费通知。

例：程序observer\_test1

// Copyright 2016.刘珅珅

// author：刘珅珅

// Observer观察者模式

#include <iostream>

#include <list>

#include <string>

using namespace std;

class Subject;

// 订阅者（观察者）基类

class Observer {

public:

// 虚析构函数

virtual ~Observer() {}

virtual void Received(Subject\* sender) = 0;

protected:

Observer() {}

};

// 发布者基类

// 在基类中维护观察者队列

class Subject

{

public:

virtual ~Subject() {}

virtual void Attach(Observer\* object);

virtual void Detach(Observer\* object);

virtual void Notify();

private:

list<Observer\*> observers; // 观察者队列

};

// 加入观察者队列

void Subject::Attach(Observer\* object)

{

observers.push\_back(object);

}

// 从观察者队列移除

void Subject::Detach(Observer\* object)

{

observers.remove(object);

}

// 通知所有订阅者

void Subject::Notify()

{

for (list<Observer\*>::iterator iter = observers.begin();

iter != observers.end(); ++iter)

{

static\_cast<Observer\*>(\*iter)->Received(this);

}

}

// 具体的发布者类

// 发布者一般只有一个

class Mobile : public Subject

{

public:

Mobile() { check\_list\_ = "";}

void SimulateCheckList();

string GetCheckList();

private:

string check\_list\_; // 话费清单

};

void Mobile::SimulateCheckList()

{

cout << "--------话费清单--------" << endl;

check\_list\_ = "市话话费：17.52\t\t长途话费57.00\n短信：5.00\n";

Notify();

}

string Mobile::GetCheckList()

{

return check\_list\_;

}

// 具体的观察者类

class Jerry : public Observer

{

public:

void Received(Subject\* sender);

};

void Jerry::Received(Subject\* sender)

{

cout << "Jerry收到话费通知:" << endl

<< static\_cast<Mobile\*>(sender)->GetCheckList() << endl;

}

// 另一个具体观察者类

class Anco : public Observer

{

public:

void Subscribe(Subject\* subject);

void Unsubscribe(Subject\* subject);

void Received(Subject\* sender);

};

void Anco::Received(Subject\* sender)

{

cout << "Anco收到话费通知:" << endl

<< static\_cast<Mobile\*>(sender)->GetCheckList() << endl;

}

// 自助订阅

void Anco::Subscribe(Subject\* subject)

{

subject->Attach(this);

}

// 取消订阅

void Anco::Unsubscribe(Subject\* subject)

{

subject->Detach(this);

}

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

{

Mobile mobile;

Jerry jerry;

Anco anco;

// 订阅

// 两种订阅方式本质上向Subject

// 维护的observers链表加一条记录

mobile.Attach(&jerry);

anco.Subscribe(&mobile);

// 发布者发布消息

cout << "\t[月底了...移动该送话费清单了]" << endl;

mobile.SimulateCheckList();

return 0;

}

输出结果为：

[月底了...移动该送话费清单了]

--------话费清单--------

Jerry收到话费通知:

市话话费：17.52 长途话费57.00

短信：5.00

Anco收到话费通知:

市话话费：17.52 长途话费57.00

短信：5.00

从上例的程序可以看出，当发布者mobile改变状态时，所有已注册的观察者都会得到通知。