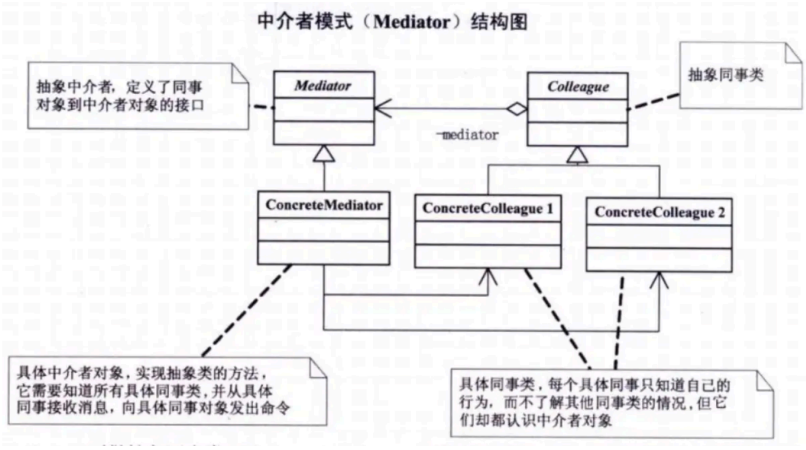
动机: 在软件构建过程中,经常会出现多个对象互相关联交互情况,对象之间常常会维持一种复杂的引用关系,如果遇到一些需求的更改,这种直接的引用关系将面临不断的变化. 在这种情况下,我们可使用一个"中介对象"来管理对象间的关联关系,避免相互交互的对象之间的紧耦合引用关系,从而更好地抵御变化.

定义:用一个中介对象来封装(封装变化)一系列的对象交互. 中介者使各对象不需要显式的相互引用(编译时依赖-->运行时依赖),从而使其耦合松散(管理变化),而且可以独立地改变它们之间的交互.

要点总结:将多个对象间复杂的关联关系解耦,中介者模式将多个对象间的控制逻辑进行集中管理,变"多个对象互相关系"为"多个对象和一个中介者关联",简化了系统的维护,抵御了可能的变化. 随着控制逻辑的复杂化. Mediator具体对象的实现可能相当复杂. 这时候可以对Mediator对象进行分解处理. 外观模式是解耦系统间(单向)的对象关联关系; 中介者模式是解耦系统内各个对象之间(双向)的关联关系.




```
#include <iostream>
using namespace std;
class Collegue;
class Mediator{
public:
    virtual ~Mediator(){}
    virtual void SendMsg(string strMsg,Collegue *pCollegue) = 0;
};
class Collegue{
protected:
    Mediator *pMediator;
public:
    Collegue(Mediator* pM){
        pMediator = pM;
    }
};
class ConcretCollegueA : public Collegue{
public:
    ConcretCollegueA(Mediator *pM):Collegue(pM){}
    void SendMsg(string strMsg){
        pMediator->SendMsg(strMsg,this);
    }
    void Notyfy(string strMsg){
        cout << "ConcretCollegueA recevie " << strMsg << endl;
    }
};
class ConcretCollegueB : public Collegue{
public:
```

```
    ConcretCollegueB(Mediator *pM):Collegue(pM){}
    void SendMsg(string strMsg){
        pMediator->SendMsg(strMsg,this);
    }
    void Notyfy(string strMsg){
        cout << "ConcretCollegueB recevie " << strMsg << endl;
    }
};
class ConcretMediator : public Mediator{
private:
    ConcretCollegueA *pA;
    ConcretCollegueB *pB;
public:
    void Set(ConcretCollegueA* a){
        pA = a;
    }
    void Set(ConcretCollegueB* b){
        pB = b;
    }
    void SendMsg(string strMsg,Collegue *pCol){
       if(pCol == pA) pB->Notyfy(strMsg);
       else pA->Notyfy(strMsg);
    }
};
int main(){
    ConcretMediator *pM = new ConcretMediator();
    ConcretCollegueA *pA = new ConcretCollegueA(pM);
    ConcretCollegueB *pB = new ConcretCollegueB(pM);
    pM->Set(pA);
    pM->Set(pB);
    pA->SendMsg("Hello ");
    pB->SendMsg("World! ");
    delete pM;
    delete pA;
    delete pB;
    return 0;
}
```

```
#include<iostream>
using namespace std;
class Mediator;
class Person{
protected:
    Mediator *m_pMediator;
public:
    virtual void SetMediator(Mediator* pM) = 0;
    virtual void GetMsg(string strMsg) = 0;//从中介获取消息
     virtual void SendMsg(string strMsg) = 0;//向中介发消息
};
class Mediator{//抽象中介
public:
    virtual ~Mediator(){}
    virtual void Send(string strMsg, Person *pPerson) = 0;
```

```
    virtual void SetA(Person *pA) = 0;
    virtual void SetB(Person *pB) = 0;
};
class Renter : public Person{
public:
    void SetMediator(Mediator* pM) {
        m_pMediator = pM;
    }
    void GetMsg(string strMsg){
        cout << "租客收到消息: " << strMsg << endl;
    }
    void SendMsg(string strMsg) {
        m_pMediator->Send(strMsg,this);
    }
};
class Landlord: public Person{
public:
    void SetMediator(Mediator* pM) {
        m_pMediator = pM;
    }
    void GetMsg(string strMsg){
        cout << "房东收到消息: " << strMsg << endl;
    }
    void SendMsg(string strMsg) {
        m_pMediator->Send(strMsg,this);
    }
};
class HoseMediator : public Mediator{
private:
    Person *m_pA;
    Person *m_pB;
public:
    HoseMediator(){
        m_pA = NULL;
        m_pB = NULL;
    }
    void SetA(Person *pA){
        m_pA = pA;
    }
    void SetB(Person *pB){
        m_pB = pB;
    }
    void Send(string strMsg, Person *pPerson){
        if(m_pA == pPerson){
            m_pB->GetMsg(strMsg);
        }
        else{
            m_pA->GetMsg(strMsg);
        }
    }
};
int main(){
    Mediator *pM = new HoseMediator();
    Person * pR = new Renter();
```

```
    Person * pL = new Landlord();
    pM->SetA(pR);
    pM->SetB(pL);
    pR->SetMediator(pM);
    pL->SetMediator(pM);
    pR->SendMsg("我想租房! 可有?");
    pL->SendMsg("这有一套性价比高的一房一厅待出租,欢迎看房" );
    return 0;
}
```

```
192:DesignPattnsStudy weishichun$ ls Mediator
Mediator_1.cpp          Mediator_2.cpp          Mediator中介者模式.pdf
192:DesignPattnsStudy weishichun$ g++ -o Mediator1.out Mediator_1.cpp
192:DesignPattnsStudy weishichun$ ./Mediator1.out
ConcretCollegueB recevie Hello
ConcretCollegueA recevie World!
192:DesignPattnsStudy weishichun$ g++ -o Mediator2.out Mediator_2.cpp
192:DesignPattnsStudy weishichun$ ./Mediator2.out
房东收到消息：我想租房！可有？
租客收到消息：这有一套性价比高的一房一厅待出租,欢迎看房
192:DesignPattnsStudy weishichun$
```