代理模式

资料： C++代理模式

01.简述

`代理模式（Proxy Pattern）`为其他对象提供了一种代理，以控制对这个对象的访问。在某些情况下，一个对象不适合或者不能直接引用另一个对象，而代理对象可以在客户端和目标对象之间起到中介的作用。

02.优缺点

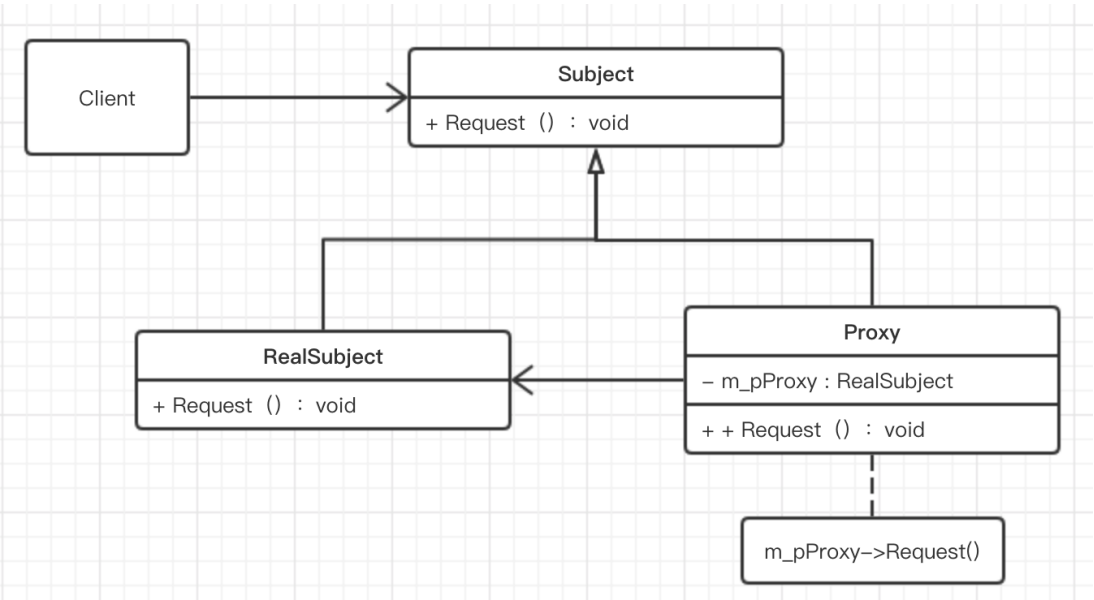
优点

- 代理模式能将代理对象与真正调用的对象分离，在的一定程度上降低了系统的耦合度。
- 在客户端和目标对象之间，代理起到了一个中介作用，这样可以保护目标对象，在对目标对象调用之前，代理对象也可以进行其他操作。

缺点

- 因为引入了代理对象，因此有些类型的代理模式可能会造成请求的处理速度变慢。
- 该模式引入了另一个抽象层，可能会存在真实主题被客户访问，并且客户还可能会访问代理类，这可能会导致不同的行为。

03.UML机构图



- Subject（抽象主题）：声明了RealSubject与Proxy的共同接口，定义功能。
- RealSubject（真实主题）：通常执行具体的业务逻辑，Proxy控制对它的访问。
- Proxy（代理）：持有一个RealSubject的指针或者引用，可以在需要时将请求转发给RealSubject，以起到代理的作用。
- Clicent（客户端）：通过Proxy间接地与RealSubject进行交互。

注意：RealSubject与Proxy都实现了Subject定义的接口，这允许Clicent可以像处理Proxy一样处理RealSubject.

04.代码

```
//
//  T5_20190120.h
//  DesignPattern
//
//  Created by shadot on 2019/1/20.
//  Copyright © 2019 shadot. All rights reserved.
//

#ifndef T5_20190120_h
#define T5_20190120_h

//代理模式

#include <iostream>

class Subject
{
public:
    virtual void Recharge(int money) = 0;
};

class RealSubject : public Subject
{
public:
    void Recharge(int money){
        std::cout << "RealSubject : " << money << std::endl;
    }
};

class Proxy : public Subject
{
public:
    void Recharge(int money){
```

```
        if (money >= 50){
            if (m_pRealSubject == nullptr){
                m_pRealSubject = new RealSubject();
                m_pRealSubject->Recharge(money);
            }
        }
        else{
            std::cout << "Proxy : " << money << " < 50 !" << endl;
        }
    }
private:
    RealSubject* m_pRealSubject;
};


#endif /* T5_20190120_h */

    //T4_20190120 测试代码
    {
        Proxy* proxy = new Proxy();
        proxy->Recharge(20);
        proxy->Recharge(100);
    }

```

```
Proxy : 20 < 50 !
RealSubject : 100
Program ended with exit code: 0
```