软件构造实验报告三

实验名称：适配器模式编程实现

实验时间：2019.4.17

学号：E21614061

姓名：徐奕

所在院系：计算机科学与技术学院

所在专业：软件工程

【实验目的和要求】

a）熟悉并理解适配器模式的原理与方法

b）熟练掌握适配器模式的代码与方法

【实验原理】

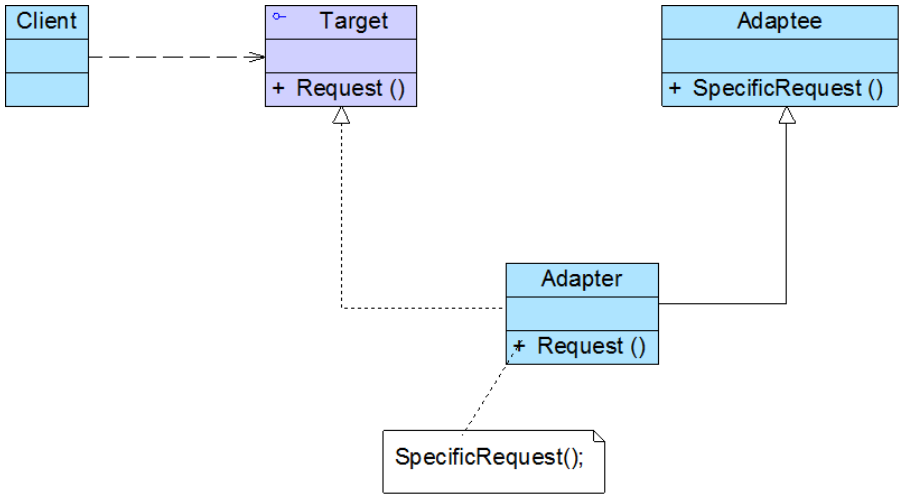
适配器模式：将一个类的接口转换成客户希望的另一个接口。适配器模式让那些接口不兼容的类可以一起工作

适配器模式的别名为包装器(Wrapper)模式，它既可以作为类结构型模式，也可以作为对象结构型模式。在适配器模式定义中所提及的接口是指广义的接口，它可以表示一个方法或者方法的集合。

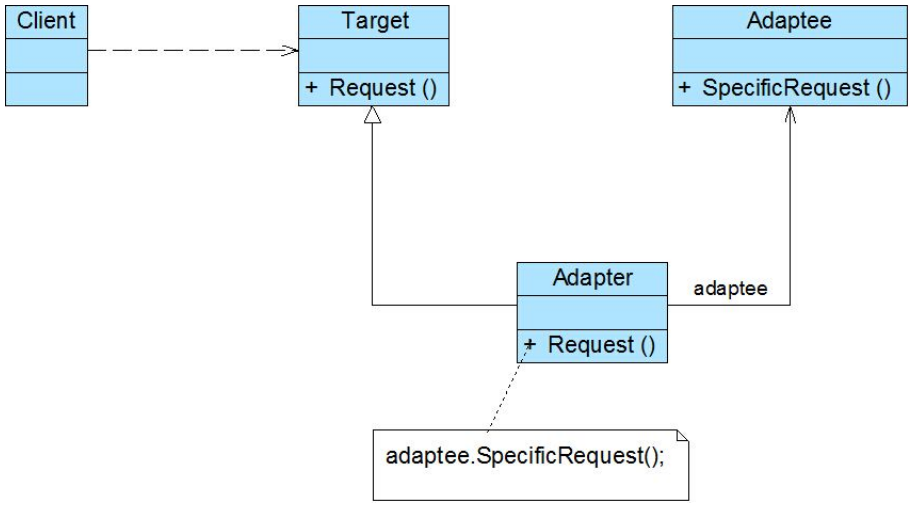
结构：

- 客户端使用的 Target 类需要使用一个已经存在的接口 Adaptee 类，可以用两种方法实现：
- 1、构造 Adapter 类继承 Target 类，并实现 Adaptee 接口（适配器模式的类版本）
- 2、将一个 Adaptee 实例作为 Adapter 的组成部分（适配器模式的对象版本）

类适配器模式结构图：



对象适配器结构图：



适配器模式的适用性：

- 想使用一个已经存在的类，但它的接口不符合需求。
- 想创建一个可以复用的类，该类可以与其它不相关的类或不可预见的类（即那些接口可能不一定兼容的类）协同工作。
- 想使用一些已经存在的子类，但是不可能对每一个都进行子类化以匹配它们的接口。对象适配器可以适配它的父类接口。

【实验内容】

分别利用类版本和对象版本的适配器模式模拟实现 ps2 接口和 usb 接口的转换。

我们手中有个 ps2 插头的设备，但是主机上只有 usb 插头的接口，实现一个适配器将 ps2 接口转换为 usb 接口。其中，**ps2 接口表示为：**

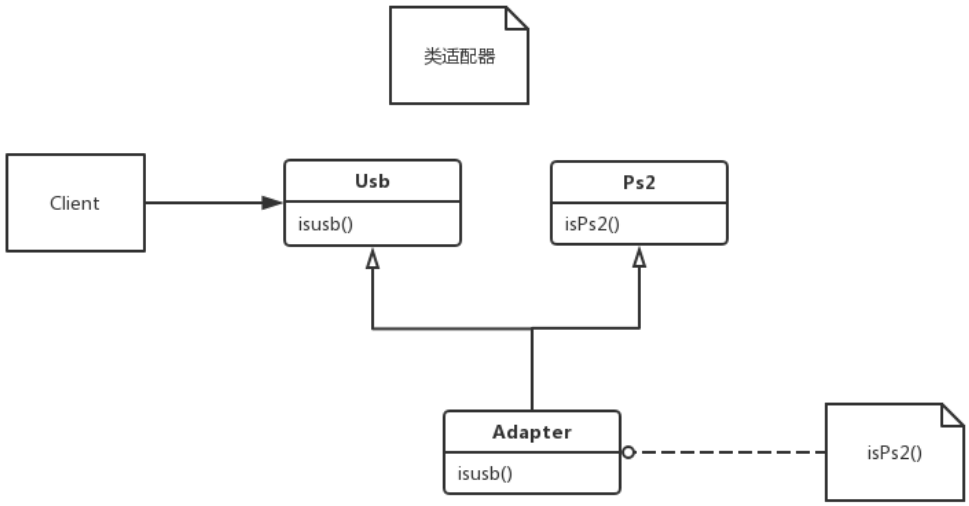
```
class Ps2{

    virtual void isPs2();

}
```

Usb 接口表示为：

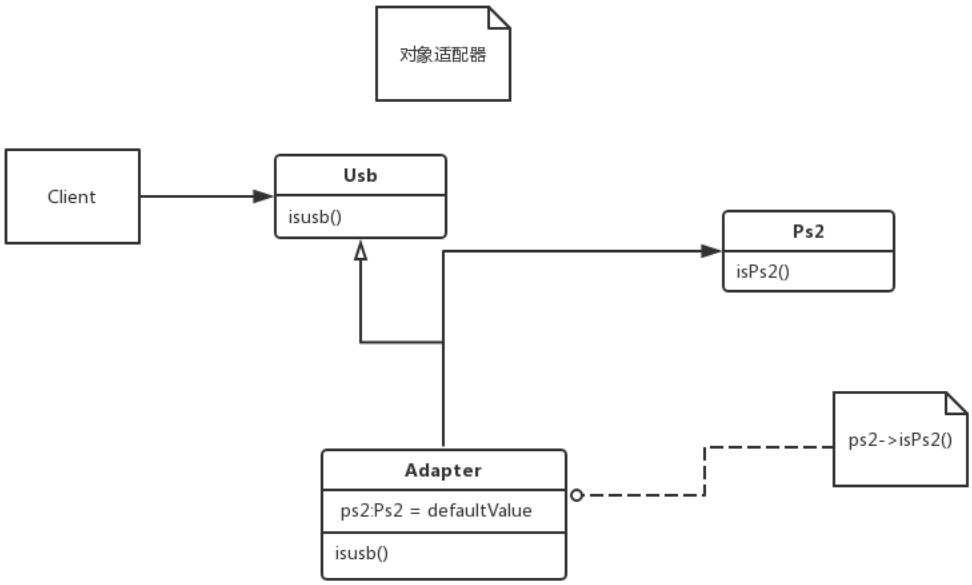
```
class Usb{

    Virtual void isusb();

}。
```

【实验 UML 图】

类适配器模式 UML 图：



对象适配器 UML 图：



【实验代码与函数】

- target 为：usb
- Adaptee 为：Ps2
- Adapter 类公有继承 target，私有继承 Adaptee

类适配器模式代码：

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

class Ps2{ //adaptee
public:
    void isPs2(){
        cout<<"Ps2"<<endl;
    };
};

class Usb{ //target
public:
    virtual void isusb(){
        cout<<"USB 接口"<<endl;
    };
};

class Adapter: public Usb, private Ps2{
    void isusb(){
        isPs2();
    }
};



int main(){
    Usb* usb = new Adapter();
    usb->isusb();
    return 0;
}
```

对象适配器代码：

```
#include<iostream>

using namespace std;

class Ps2{ //adaptee
public:
    void isPs2(){
        cout<<"Ps2"<<endl;
    };
};

class Usb{ //target
public:
    virtual void isusb(){
        cout<<"USB接口"<<endl;
    };
};

class Adapter: public Usb{
public:
    void isusb(){
        ps2->isPs2();
    }
private:
    Ps2 * ps2 = new Ps2();
};

int main(){
    Usb* usb = new Adapter();
    usb->isusb();
    return 0;
}
```

【实验结果】

两个均调用的为 SpecificRequest();

```
Ps2
```

```
Ps2
```

【实验总结】

①本次实验掌握并编码了适配器模式的类与对象的两种不同方法。

②在实现类适配器和对象适配器时，注意，对象适配器模式中的“目标接口”和“适配者类”的代码同类适配器模式一样，只要修改适配器类和客户端的代码即可。（如下图）

```
int main(){
    Usb* usb = new Adapter();
    usb->isusb();
    return 0;
}
```