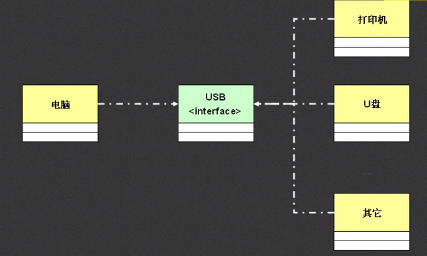
# 工厂设计模式



**两个无关的类之间的连接一定需要依靠接口来完成，有接口定义出操作的标准**

* + - 1. **首先定义接口！！！**

**public** **interface** IUSB {

**public** **void** work();// usb设备工作

**public** **void** exit();// 退出USB设备

}

* + - 1. **定义电脑**

**public** **class** Computer {

**public** **void** plugin(USB usb) {// 接收USB设备

usb.work();// 调用usb接口方法，由子类实现

usb.exit();// 完事之后退出

}

* + - 1. **定义设备**

**public** **class** Flash **implements** USB {

@Override

**public** **void** work() {

System.***out***.println("u盘开始拷贝");

}

@Override

**public** **void** exit() {

System.***out***.println("u盘结束拔出");

}

}

**public** **class** Print **implements** USB {

@Override

**public** **void** work() {

System.***out***.println("打印机开始打印");

}

@Override

**public** **void** exit() {

System.***out***.println("打印机结束工作");

}

}

**4.测试**

@Test

**public** **void** test() {

Computer com = **new** Computer();

com.plugin(**new** Flash());

com.plugin(**new** Print());

}

u盘开始拷贝

u盘结束拔出

打印机开始打印

打印机结束工作

**而在此工程中，电脑和打印机还是彼此独立的类，没有直接定义联系，但有标准联系**

**interface** Fruit {

**public** **void** eat();

}

**class** Apple **implements** Fruit {

@Override

**public** **void** eat() {

System.***out***.println("吃苹果");

}

}

**class** Orange **implements** Fruit {

@Override

**public** **void** eat() {

System.***out***.println("吃橘子");

}

}

**class Factory {**

**public static Fruit getInstance(String className) {**

**if ("apple".equals(className)) {**

**return new Apple();**

**}**

**if ("orange".equals(className)) {**

**return new Orange();**

**}**

**If(“cherry”.equals(className)){**

**Return new Cherry();**

**}**

**return null;**

**}**

**}**

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Fruit f1 = Factory.*getInstance*("orange");

f1.eat();

Fruit f2 = Factory.*getInstance*("apple");

f2.eat();

}

}

**当增加新水果时，只需要修改工厂类，而不修改客户端！！！**

# 二、代理设计模式

**在本结构中，一个接口定义了两个子类，一个子类完成核心的真实业务主题，而另一个子类要为真实核心业务提供所有的辅助功能**

# 

**interface** NetWork {// 网络连接

**public** **void** browse(String url);

}

**class** RetNetWork **implements** NetWork {// 真正上网

@Override

**public** **void** browse(String url) {

System.***out***.println("访问站点：" + url);

}

}

**class** ProxyNetWork **implements** NetWork {

**private** NetWork net;// 保证真正的上网连接

**public** ProxyNetWork(NetWork net) {

**this**.net = net;

}

**public** Boolean check(String url) {// 检查网站

**if** (url.contains("whitehouse")) {

System.***out***.println("请浏览合法网站");

**return** **false**;

}

**return** **true**;// 检查合格

}

@Override

**public** **void** browse(String url) {

**if** (**this**.check(url)) {

**this**.net.browse(url);

}

}

**public** **void** record(String url) {

System.***out***.println("上网记录:" + url);

}

}

**public** **class** one {

**public** **static** **void** main(String[] args) {

NetWork nw = **new** ProxyNetWork(**new** RetNetWork());

nw.browse("www.weibo.com");

nw.browse("www.whitehouse.com");

}

}

访问站点：www.weibo.com

请浏览合法网站

访问站点：www.whitehouse.com

请浏览合法网站