**单例模式**

1、单例类只能有一个实例。  
2、单例类必须自己创建自己的唯一实例。  
3、单例类必须给所有其他对象提供这一实例。

1. **public** **static** Singleton getInstance() {
2. **if** (singleton == **null**) {
3. **synchronized** (Singleton.**class**) {
4. **if** (singleton == **null**) {
5. singleton = **new** Singleton();
6. }
7. }
8. }
9. **return** singleton;
10. }
11. **public** **class** Singleton {
12. **private** **static** **class** LazyHolder {
13. **private** **static** **final** Singleton INSTANCE = **new** Singleton();
14. }
15. **private** Singleton (){}
16. **public** **static** **final** Singleton getInstance() {
17. **return** LazyHolder.INSTANCE;
18. }
19. }

//线程安全，不需同步

代理

Class simpleProxy{

Private Interface proxied;

Public SimpleProxy(Interface proxied){

This.proxied = proxied;

}

Public void doSomethid(){

Proxied.doSomethind();

}

}

动态代理

Interface proxy = (Interface)Proxy.newProxyInstance(

Interface.class.getClassLoader();//一个类加载器

New Class[]{Interface.class}//代理实现的列表接口 ，这里只希望代理Interface

New DynamicProxyHandler(real)) //InvocationHandler的一个实现);

Proxy.doSomething()

InvocationHandler主要实现 invoke(Object proxy,Method method,Object [] args)和它的构造器

//实现

构造方法(要代理的对象){

Proxied = 要代理的对象;

}

invoke(Object proxy,Method method,Object [] args){

method.invoke(proxied,args);

}