29 动态代理

- 1 概念

特点：无侵入式的给代码增加额外的功能

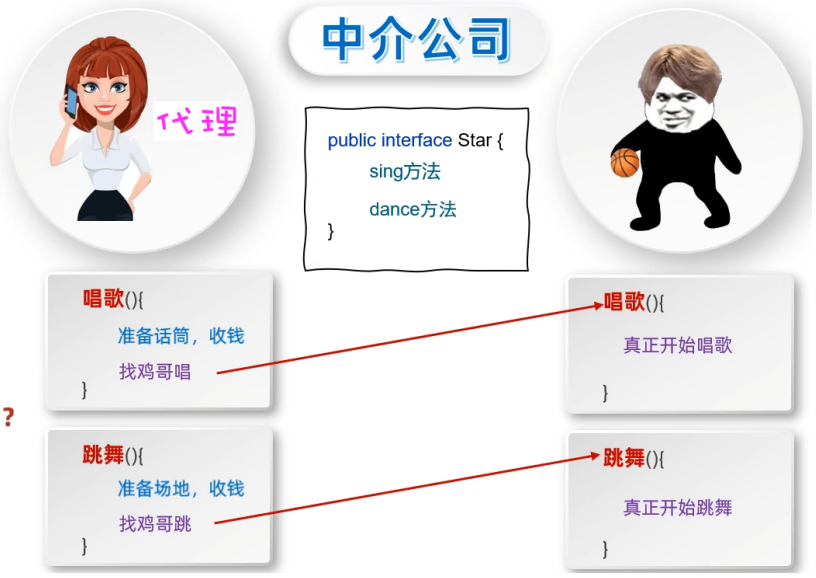
程序为什么需要代理？代理长什么样？

对象如果嫌身上干的事太多的话，可以通过代理来转移部分职责。

对象有什么方法想被代理，代理就一定要有对应的方法

中介如何知道要派有唱歌、跳舞方法的代理呢？

接口



- 2 总结

1. 为什么需要代理?

代理可以无侵入式的给对象增强其他的功能

2. 代理长什么样?

代理里面就是对象要被代理的方法

3. Java通过什么来保证代理的样子?

通过接口保证，后面的对象和代理需要实现同一个接口
接口中就是被代理的所有方法

- 3 代码实现
  - 3.1 JavaBean 类(BigStar)的两个方法

```
//唱歌
@Override
public String sing(String name){
    System.out.println(this.name + "正在唱" + name);
    return "谢谢";
}

//跳舞
@Override
public void dance(){
    System.out.println(this.name + "正在跳舞");
}
```

- 3.2 接口(中介公司)

```
public interface Star {

    //我们可以把所有想要被代理的方法定义在接口当中

    //唱歌
    public abstract String sing(String name);

    //跳舞
    public abstract void dance();


}
```

- 3.3 代理对象
  - java.lang.reflect.Proxy类：提供了为对象产生代理对象的方法：

```
public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces, InvocationHandler h)
```

参数一：用于指定用哪个类加载器，去加载生成的代理类

参数二：指定接口，这些接口用于指定生成的代理长什么，也就是有哪些方法

参数三：用来指定生成的代理对象要干什么事情

```
public class ProxyUtil {

需求：
  外面的人想要大明星唱一首歌
  1. 获取代理的对象
     代理对象 = ProxyUtil.createProxy(大明星的对象);
  2. 再调用代理的唱歌方法
     代理对象.唱歌的方法();
```

```
    public static Star createProxy(BigStar bigStar){
       /* java.lang.reflect.Proxy类：提供了为对象产生代理对象的方法：

        public static Object newProxyInstance(ClassLoader loader, Class<?>[] interfaces, InvocationHandler h)
        参数一：用于指定用哪个类加载器，去加载生成的代理类
        参数二：指定接口，这些接口用于指定生成的代理长什么，也就是有哪些方法
        参数三：用来指定生成的代理对象要干什么事情*/

        Star star = (Star) Proxy.newProxyInstance(
                ProxyUtil.class.getClassLoader(),//参数一：用于指定用哪个类加载器，去加载生成的代理类
                new Class[]{Star.class},//参数二：指定接口，这些接口用于指定生成的代理长什么，也就是有哪些方法
                //参数三：用来指定生成的代理对象要干什么事情
                new InvocationHandler() {
                    @Override
                    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
                        /*
                        * 参数一：代理的对象
                        * 参数二：要运行的方法 sing
                        * 参数三：调用sing方法时，传递的实参
                        * */
                        if("sing".equals(method.getName())){
                            System.out.println("准备话筒，收钱");
                        }else if("dance".equals(method.getName())){
                            System.out.println("准备场地，收钱");
                        }
                        //去找大明星开始唱歌或者跳舞
                        //代码的表现形式：调用大明星里面唱歌或者跳舞的方法
                        return method.invoke(bigStar,args);
                    }
                }
        );
        return star;

    }
}
```

- **3.4 测试**

```
/*
    需求：
        外面的人想要大明星唱一首歌
         1. 获取代理的对象
            代理对象 = ProxyUtil.createProxy(大明星的对象);
         2. 再调用代理的唱歌方法
            代理对象.唱歌的方法("只因你太美");
 */


    //1. 获取代理的对象
    BigStar bigStar = new BigStar( name: "鸡哥");
    Star proxy = ProxyUtil.createProxy(bigStar);

    //2. 调用唱歌的方法
    String result = proxy.sing( name: "只因你太美");
    System.out.println(result);
```