**代理模式的定义：**对其他对象提供一种代理以控制对这个对象的访问。

对于某一功能来说（这些功能对应为一些接口），一个类A继承这些接口并提供这个功能的实现。但是 不能直接调用类A的对象来使用这些接口定义的功能，而是新建一个类B来调用类A的对象。同过类B来调用类A实现的功能，并决定何时以及如何调用这些功能。 这里的类A和类B分别即为被代理类与代理类。

1.一个类（被代理类）拥有某些功能（方法,这些方法被定义在一个功能接口中），这些功能的方法需要通过另一个类（代理类）来执行。即用户不能直接使用被代理类对象来执行其提供的方法，需要通过一个中介（代理类）来执行。

2.代理类可以在被代理类提供的功能的基础上做一些扩展（比如在执行被代理类方法前后打印一些日志）。而用户在通过代理类对象使用被代理类提供的功能时，同样可以使用这些扩展的功能。

3.代理类不关心如何实现，它只对这些方法的运行进行管理。它就像是一个管家，或者说是一个包装类，对被代理类进行包装和管理。

**静态代理的特点是：** 一个代理类与 一类 特定的被代理类绑定（指继承了相同的功能接口的实现类），在编译时就已确定其间的绑定关系。因而需要为每一类被代理类写一个代理类。他们之间有这种种绑定关系是因为代理类与被代理类需要实现相同的功能接口。

**动态代理的特点是：**静态代理的代理类与被代理类进行绑定（具体来说是与那些接口绑定），这样代码耦合度高。动态代理实现了使用同一个代理类来代理所有不同的被代理类（即使被代理类实现的接口不同）。他的技术基础是Java反射。通过反射来动态创建被代理类对象。动态代理在运行时才确定与哪一类被代理类绑定。在运行时通过反射动态创建代理类对象来代理不同的代理类。

动态代理与静态代理的区别在于代理类是否绑定了被代理类实现的接口。静态代理的代理类实现了被代理类同样的接口，绑定关系在编译时就已经确定。代码耦合度高。动态代理的代理类不指定任何接口，他通过反射动态获取被代理类对象的结构并创建被代理类的代理对象。而且动态代理的代理类对被代理类功能的扩展是面向切面编程（aop）的基本原理。