*绑定指的是一个**方法**的调用与**方法**所在的类(方法主体)关联起来。*

静态绑定：

前期绑定：在程序执行前方法已经被绑定（也就是说在编译过程中就已经知道这个方法到底是哪个类中的方法），此时由编译器或其它连接程序实现。例如：C。

可以理解为程序编译期的绑定：java当中的方法只有final，static，private和构造方法（非静态方法）是前期绑定。

private：不能被继承，类自身的对象来调用。

final方法虽然可以被继承，但不能被重写（由此我们可以知道将方法声明为final类型，一是为了防止方法被覆盖，二是为了有效地关闭java中的动态绑定)。

static：static方法不可以被子类继承，但是不能被子类重写（覆盖），但是可以被子类隐藏。（这里意思是说如果父类里有一个static方法，它的子类里如果没有对应的方法，那么当子类对象调用这个方法时就会使用父类中的方法。而如果子类中定义了相同的方法，则会调用子类的中定义的方法。唯一的不同就是，当子类对象上转型为父类对象时，不论子类中有没有定义这个静态方法，该对象都会使用父类中的静态方法。因此这里说静态方法可以被隐藏而不能被覆盖。这与子类隐藏父类中的成员变量是一样的。隐藏和覆盖的区别在于，子类对象转换成父类对象后，能够访问父类被隐藏的变量和方法，而不能访问父类被覆盖的方法）

动态绑定：

后期绑定：在运行时根据具体对象的类型进行绑定。若一种语言实现了后期绑定，同时必须提供一些机制，可在运行期间判断对象的类。

动态绑定的过程：

1. 虚拟机提取对象的实际类型的方法表；
2. 虚拟机搜索方法签名；
3. 调用方法。

方法表是动态调用的核心，也是 Java 实现动态调用的主要方式。它被存储于方法区中的类型信息，包含有该类型所定义的所有方法及指向这些方法代码的指针。

JAVA 虚拟机调用一个类方法时（静态方法），它会基于对象引用的类型(通常在编译时可知)来选择所调用的方法。
当虚拟机调用一个实例方法时，它会基于对象实际的类型(只能在运行时得知)来选择所调用的方法，这就是动态绑定。
与方法不同，在处理java类中的成员变量（实例变量和类变量）时，并不是采用运行时绑定，而是一般意义上的静态绑定。