

**Overloading** 在一个类中可以定义多个同名方法，各个方法的参数表一定不同。但修饰词可能相同，返回值也可能相同。

在程序的编译过程中根据变量类型来找相应的方法。因此也有人认为 **overloading** 是编译时的多态，以后我们还会学到运行时多态。

为什么会存在 **overloading** 技术呢？作为应对方法的细节。

利用类型的差异来影响对方法的调用。

吃（）可以分为吃肉，吃菜，吃药，在一个类中可以定义多个吃方法。

构造方法也可以实现 **overloading**。例：

```
public void teach();  
public void teach(int a);  
public void teach(String a){}为三种不同的方法。
```

**Overloading** 方法是从低向高转。

Byte—short—float—int—long—double。

在构造方法中，**this** 表示本类的其他构造方法：

```
student();  
student(string n){  
    this();//表示调用 student()  
}
```

如果调用 **student(int a)** 则为 **this(int a)**。

**特别注意：**用 **this** 调用其他构造方法时，**this** 必须为第一条语句，然后才是其他语句。

**This** 表示当前对象。

```
Public void printNum(){  
    Int number=40;  
    System.out.println(this.number);  
}
```

此时打印的是实例变量，而非局部变量，即定义在类中而非方法中的变量。

**This.number** 表示实例变量。

谁调用 **this.number** 那么谁即为当前(**this**)对象的 **number** 方法。

封装：使对象的属性尽可能私有，对象的方法尽可能的公开。用 **private** 表示此成员属性为该类的私有属性。

**Public** 表示该属性（方法）公开；

**Private** 表示该属性（方法）为只有本类内部可以访问（类内部可见）。

（想用 **private** 还要用 **set** 和 **get** 方法供其他方法调用，这样可以保证对属性的访问方式统一，并且便于维护访问权限以及属性数据合法性）

如果没有特殊情况，属性一定私有，方法该公开的公开。

如果不指明谁调用方法，则默认为 **this**。

区分实例变量和局部变量时一定要写 **this**。

## 11.29

继承：