**方法头（method header）是指方法的修饰符（modifier）、返回值类型（return value type）、方法名（method name）和方法的参数（parameter）。**

**在其他某些语言中，方法称为过程（procedure）或函数（function）。带返回值的方法称为函数，返回值类型为void的方法称为过程。**

**我们经常会说“定义方法”和“声明变量”，定义是指被定义的条目是什么，而声明通常是指为被声明的条目分配内存来存储数据。**

**为了使用方法，必须调用（call 或invoke）它。**

**方法能够带来代码的共享和重用，可以在其他类中调用它。如果创建了一个新类，可以通过使用“类名.方法名”来调用。**

**每当调用一个方法时，系统会创建一个活动记录（也称为活动框架），用于保存方法中的参数和变量。活动记录置于一个内存区域中，称为调用堆栈（call stack）。调用堆栈也称为执行堆栈，运行时堆栈，或者一个机器堆栈，常简称为“堆栈”。**

**当一个方法调用另一个方法时，调用者的活动记录保持不动，一个新的活动记录被创建用于被调用的新方法。一个方法结束返回到调用者时，其相应的活动记录也被释放。**

**void方法不需要return语句，但它能用于终止方法并返回到方法的调用者。它的语法是：return;**

**这种用法很少，但是对于改变void方法中的正常流程控制是很有用的。**

**实参必须与方法签名中定义的参数在次序和数量上匹配，在类型上兼容。类型兼容是指不需要经过显式的类型转换，实参的值就可以传递给形参，例如，将int型的实参值传递给double型形参。**

**当调用带参数的方法时，实参的值传递给形参，这个过程称为按值传递（pass-by-value）。**

**使用方法可以减少冗余的代码，提高代码的复用性。方法可以用来模块化代码，以提高程序的质量。**

**模块化使得代码易于维护和调试，并且使得代码可以被重用。**

**重载方法**

**重载方法使得你可以使用同样的名字来定义不同方法，只要它们的签名是不同的。**

**重载方法可以使得程序更加清楚，以及更加具有可读性。执行同样功能但是具有不同参数类型的方法应该使用同样的名字。**

**被重载的方法必须具有不同的参数列表。不能基于不同修饰符或返回值类型来重载方法。**

**有时调用一个方法时，会有两个或更多可能的匹配，但是，编译器无法判断哪个是最精确的匹配。这称为歧义调用（ambiguous invocation）。歧义调用会产生一个编译错误。**

**变量的作用于（scope of a variable）是指变量可以在程序中引用的范围。**

**生成随机字符**

**字符使用整数来编码，产生一个随机字符就是产生一个随机整数。**

**所有的数字操作符都可以应用到char操作数上。如果一个操作数是数字或字符，那么char型操作数就会被转换成数字。这样（int）'a'到（int）'z'之间的随机整数就可以简化成**

**'a'+Math.random()\*('z'-'a'+1)**

**这样，随机的小写字母就是：**

**(char)('a'+Math.random()\*('z'-'a'+1))**

**由此，可以生成任意两个字符ch1和ch2之间的随机字符，其中ch1<ch2**

**(char)(ch1+Math.random()\*(ch2-ch1+1))**

**方法抽象（method abstraction）是通过将方法的使用和它的实现分离来实现的。用户在不知道方法是如何实现的情况下，就可以使用方法。方法的实现细节封装在方法内，对使用该方法的用户来说是隐藏的。这就称为信息隐藏（information hiding）或封装（encapsulation）。如果决定改变方法的实现，但只要不改变方法签名，用户的程序就不受影响。**

**当实现一个大型程序时，可以使用自顶向下或自底向上的编码方法。不要一次性编写完整个程序。**