**为对象定义类**

**类为对象定义属性和行为。**

**面向对象程序设计（OOP）就是使用对象进行程序设计。对象（object）代表现实世界中可以明确标识的一个实体。例如，一个学生、一张桌子、一个圆。每个对象都有自己独特的标识、状态和行为。**

**一个对象的状态（state，也称为特征（property）或属性（attribute））是由具有当前值的数据域来表示的。**

**一个对象的行为（behavior，也称为动作（action））是由方法定义的。调用对象的一个方法就是要求对象完成一个动作。**

**可以从一个类中创建多个实例。创建实例的过程称为实例化（instantiation）。对象和实例经常是可以互换的。**

**java使用变量定义数据域，使用方法定义动作。**

**类还提供了一种称为构造方法的（constructor）的特殊类型的方法，调用它可以创建一个新对象。构造方法本身是可以完成任何动作的，但是设计构造方法是为了完成初始化动作，例如：初始化对象的数据域。**

**类的模板和对象的图示可以使用统一建模语言（Unified Modeling Language，UML）的图形符号进行标准化。这种方法称为UML类图。**

**使用构造方法构造对象**

**构造方法在使用new操作符创建对象的时候被调用。**

**构造方法三个特殊性：**

**1.构造方法必须具备和所在类相同的名字。**

**2.构造方法没有返回值类型，甚至连void都没有。**

**3.构造方法是在创建一个对象使用new操作符时调用的。构造方法的作用是初始化对象。**

**构造方法也可以重载，易于用不同的初始数据值来构造对象。**

**一个类可以不定义构造方法。在这种情况下，类中隐含定义一个方法体为空的无参构造方法。这个构造方法称为默认构造方法，当且仅当类中没有明确定义任何构造方法时才会自动提供它。**

**如果一个引用类型的数据域没有引用任何对象，那么这个数据域就有一个特殊的java值null。null同true和false一样都是一个直接量。true和false是boolean类型直接量，而null是引用类型直接量。**

**引用类型数据域的默认值是null，数值类型数据域的默认值是0，boolean类型数据域的默认值是false，而char类型数据域的默认值是'\u0000'。但是java没有给方法中的局部变量赋默认值，例如**

**int x；**

**//编译错误，方法中的局部变量没有被初始化**

**System.out.println("x is "+x);**

**NullPointerException是一种常见的运行时错误，当调用值为null的引用变量上的方法时会发生此类异常。在通过引用变量调用一个方法之前，确保先将对象引用赋值给这个变量。**

**基本类型变量在内存中存储的是一个基本类型值，而引用类型变量存储的是一个引用，它指向对象在内存中的位置。**

**如果你认为不再需要某个对象时，可以显式地给该对象的引用变量赋null值。如果该对象没有被任何引用变量所引用，java虚拟机将自动回收它所占的空间。**

**Date类**

**可以使用Date类中的无参构造方法为当前的日期和时间创建一个实例，它的getTime()方法返回自从GMT时间1970年1月1日算起至今流逝的时间，它的toString()方法返回日期和时间的字符串。**

**Random类**

**可以使用Math.random()获取一个0.0到1.0（不包括1.0）之间的随机double型值。另一种产生随机数的方法是使用java.util.Random类，它可以产生一个int、long、double、float和boolean型值。**

**创建一个Random对象时，必须制定一个种子或者使用默认的种子。种子是一个用于初始化一个随机数字生成器的数字。无参构造方法使用当前已经逝去的时间作为种子，创建一个Random对象。如果这两个Random对象有相同的种子，那它们将产生相同的数列。**

**Random()--------以当前时间作为种子创建一个Random对象**

**Random(seed:long)------以一个特定值作为种子创建Random对象**

**nextInt():int--------返回一个随机的int值**

**nextInt(n:int):int---------返回一个0到n（不含n）之间的随机int值**

**nextLong():long---------返回一个随机的long值**

**nextDouble():double----------返回一个0.0到1.0（不含1.0）之间的随机double值**

**nextFloat():float--------类似**

**nextBoolean():boolean------返回一个随机的boolean值**

**Random Random1=new Random();**

**int i=random.nextInt();**

**静态变量、常量和方法**

**静态变量被类中所有对象所共享。静态方法不能访问类中的实例成员。**

**实例变量是绑定到类的某个特定实例的，它是不能被同一个类的不同对象所共享的。**

**如果想让一个类的所有实例共享数据，就要使用静态变量（static variable),也称为类变量（class variable）。静态变量将变量值存储在一个公共的内存地址。因为它是公共的地址，所以如果某一个对象修改了静态变量的值，那么同一个类的所有对象都会受到影响。**

**java支持静态方法和静态变量，无须创建类的实例就可以调用静态方法。**

**类中的常量是被该类的所有对象共享的。因此，常量应该声明为final static，例如**

**final static double PI=3.1415926535;**

**PI是一个定义在Math中的一个常量，可以使用Math.PI来访问。**

**使用“类名.方法名（参数）”的方式调用静态方法，使用“类名.静态变量”的方式调用静态变量。这会提高可读性。**

**实例方法可以调用实例方法和静态方法，以及访问实例数据域或者静态数据域。**

**静态方法可以调用静态方法以及访问静态数据域。**

**然而，静态方法不能调用实例方法或者访问实例数据域，因为静态方法和静态数据域不属于某个特定的对象。**

**可见性修饰符**

**可见性修饰符可以用于确定一个类以及它的成员的可见性。**

**可以在类、方法和数据域前使用public修饰符，表示它们可以被任何其他的类访问。如果没有使用这些特定的修饰符，则默认可以被同一个包中的任何一个类访问。这称作包私有（package-private）或包内访问（package-access）。**

**包可以用来组织类。为了完成这个目标，需要在程序中首先出现下面这行语句，在这行语句之前不能有注释也不能有空白：**

**package packageName;**

**如果定义类时没有声明包，就表示把它放在默认包中。**

**private修饰符限定方法和数据域只能在它自己的类中被访问。**

**私有的修饰符将访问权限限定在它自己的类内，默认修饰符将访问权限限定在包内，而公共的修饰符可以无限制的访问。**

**修饰符private只能应用在类的成员上。修饰符public可以应用在类或类的成员上。在局部变量上使用public和private都会导致编译错误。**

**大多数情况下，构造方法应该是公共的，但如果想防止用户创建类的实例，就该使用私有构造方法。例如，因为Math类中的所有数据域和方法都是静态的，所以没必要创建Maht类的实例。**

**private Math(){**

**}**

**数据域封装**

**将数据域设为私有保护数据，并且使类易于维护。**

**为了避免对数据域的直接修改，应该使用private修饰符将数据域声明为私有的，这称为数据域封装（data field encapsulation）。**

**在定义私有数据域的类外的对象是不能访问这个数据域的。但是经常会有客户端需要存取、修改数据域的情况。为了能够访问私有数据域，可以提供一个get方法返回数据域的值。为了能够更新一个数据域，可以提供一个set方法给数据域设置新值。get方法也被称为访问器（accessor），而set方法被称为修改器（mutator）。**

**get方法有如下签名：**

**public returnType getPropertyName（）**

**如果返回类型是boolean型，习惯上如下定义get方法：**

**public boolean isPropertyName（）**

**set方法有如下签名：**

**public void setPropertyName（dataType propertyValue）**

**给方法传递一个对象，是将对象的引用传递给方法。**

**java只有一种参数传递方式：值传递。**

**引用上的传值在语义上最好描述为传共享（pass-by-sharing），也就是说，在方法中引用的对象和传递的对象是一样的。**

**对象数组**

**数组既可以存储基本数据类型值，也可以存储对象。例如**

**Circle[] circleArray=new Circle[10];**

**为了初始化数组circleArray，可以使用如下的for循环：**

**for(int i=0;i<circleArray.length;i++){**

**circleArray[i]=new Circle();**

**}**

**对象的数组实际上是引用变量的数组。因此，调用circleArray[1].getArea()实际上调用了两个层次的引用。circleArray引用了整个数组，circleArray[1]引用了一个Circle对象。**

**当使用new操作符创建对象数组后，这个数组中的每个元素都是默认值为null的引用变量。**

**不可变对象和类**

**可以定义不可变类来产生不可变对象。不可变对象的内容不能被改变。**

**通常，创建一个对象后，它的内容是允许之后改变的。有时候也需要创建一个一旦创建其内容就不能再改变的对象。我们称这种对象为一个不可变对象（immutable object），而它的类就称为不可变类（immutable class）。**

**例如，String类就是不可变的。**

**如果一个类是不可变的，那么它的所有数据域必须都是私有的，而且没有对任何一个数据域提供公共的set方法。**

**一个类的所有数据域都是私有的切没有修改器并不意味着它一定是不可变类。**

**要使一个类称为不可变的，它必须满足：**

**1）所有数据域都是私有的。**

**2）没有修改器方法。**

**3）没有一个返回指向可变数据域的引用的访问器方法。**

**变量的作用域**

**实例变量和静态变量的作用域是整个类，无论变量是在哪里声明的。**

**一个类的实例变量和静态变量称为类变量（class's variable）或数据域（data field）。在方法内部定义的变量称为局部变量。无论在何处声明，类变量的作用域都是整个类。**

**类的变量和方法可以在类中以任意顺序出现。但是，当一个数据域是基于对另一个数据域的引用来进行初始化时则不是这样。在这种情况下，必须先声明另一个数据域。**

**如果一个局部变量和一个类变量具有相同的名字，那么局部变量优先，而同名的类变量将被隐藏。**

**为了避免混淆和错误，除了方法中的参数，不要将实例变量或静态变量的名字作为局部变量名。**

**this引用**

**关键字this引用对象自身。它也可以在构造方法内部用于调用同一个类的其他构造方法。**

**关键字this是指向调用对象本身的引用名。可以用this关键字引用对象的实例成员。**

**this关键字可以用于引用类的隐藏数据域。例如，在数据域的set方法中，经常将数据域名用作参数名。在这种情况下，这个数据域在set方法中被隐藏。为了给它设置新值，需要在方法中引用隐藏的数据域名。**

**隐藏的静态变量可以简单地通过“类名.静态变量”。隐藏的实例变量就需要使用关键字this来引用。**

**this可以用于调用同一个类的另一个构造方法。**

**java要求在构造方法中，语句this（参数列表）应在任何其他可执行语句之前出现。**

**如果一个类中有多个构造方法，最好尽可能使用this（参数列表）实现它们。通常，无参数或参数少的构造方法可以用this（参数列表）调用参数多的构造方法。这样可以简化代码，使类易于维护和阅读。**