# 初步认识

1. Java程序是由类构建而成的。从类定义开始，你可以创建任意多的对象，这些对象通常称为那个类的实例。
2. 类包含成员，最主要的成员有余和方法。域是属于类本身或类对象的数据变量，他们构成对象或类的状态。方法是在域上进行运算从而操纵状态的语句集合。语句定义类的行为：把值赋给域或者其他变量，计算算术表达式，调用方法和控制执行流程

Java基本数据类型：

|  |  |
| --- | --- |
| Boolean | True/false |
| Char | 16位 |
| byte | 8 |
| short | 16 |
| int | 32 |
| long | 64 |
| float | 32 |
| double | 64 |

类和对象：

1. 域是与类和它的对象相关联的数据变量。域保存通过类运算的结果。
2. 方法包含类的可执行代码，方法由语句构建。方法的调用方式及方法包含的语句最终决定程序的执行。
3. 类和接口可以是其他类或者接口的成员
4. 对象通过含有new关键字的表达式创建。从类定义的创建对象也叫作实例化（instantiation）,所以对象经常被叫做实例（instance）。
5. 在JAVA 中，新创建的对象分配在堆（heap）的系统内存去中。
6. Java中所有对象都通过对象引用（Object reference）进行访问。这种变量类型叫做引用类型（Object reference），它与基本类型不同，基本类型的变量中存的是对应类型的值，当对象引用不再引用任何对象时，就为空。
7. 同类的所有对象共享一个域。这种共享的变量叫做类变量（class varible）,它特定于类而不是类的对象。
8. 未被引用的对象由垃圾回收器（garbage collector）自动回收。垃圾收集器在后台运行，并跟踪对象引用，当对象不再引用时，垃圾收集器就把它从内存分配堆里除去，虽然真正的删除可能要等到合适的时候。
9. 面向对象的好处：隐藏了内部数据的具体实现。类的操作通过方法（既操作对象数据以获得结果的指令）来声明，方法可以访问对其他对象隐藏了的实现细节，把数据隐藏在方法后，使别的对象不能访问，这就是数据包装（data encapsulation）的基础。
10. 方法可能只返回单一的值。如果让方法返回多个值，必须创建另一个对象，他唯一的目的就是保存返回值，并返回这个对象。
11. 当调用一个方法时，执行流就离开当前的方法，从被调用的方法的方法体开始执行。当调用方法执行结束后，当前方法就从调用方法的那条语句之后开始执行。当我们开始执行方法体时，从这个方法的角度来看，方法调用的目标对象就是当前（current）或接收方法（receiver）对象。传递给方法的自变量通过方法声明的参数进行访问。
12. 类方法：特定于类的方法。类方法常用于完成特定于类自身的操作，这些操作通常作用于类的实例。类方法通过static关键字声明，所以叫做静态方法。
13. 静态方法不能直接访问非静态成员。调用静态方法时，并没有特定的对象引用让他操作。但是一般情况下，静态方法完成类方面的饿操作，非静态类完成对象方面的操作。作用于对象域的静态方法。
14. 数组是相同类型变量的集合。
15. 数组长度在创建后保持固定不变
16. 数组对象具有一个length域，表明数组中的元素个数。数组的边界为0-length-1之间，包含上下界。访问数组边界以外的元素是一种常见的错误，特别是在数组元素上执行循环时。为了捕获这种错误，所有的数组都要进行越界检查，确保下标在边界内。如果试图使用超出边界的下标，运行时系统将向程序报告该错误，方法会抛出IndexOutOfBoundsException异常。
17. 长度为0的数组是空数组。以数组为自变量的方法一般要求接收到的数组非空，并将检查数组的长度。但是，在检查数组长度之前，首先必须保证数组非空。如果其中任何一个检查失败了，方法就会抛出IllegalArgumentException异常。
18. String对象具有一个length方法，他会返回String中的字符个数。其中，字符下标为0-length-1，你可以通过charAt方法访问。CharAt方法以整数下标作为自变量，返回指定下标的字符。在这种情况下，字符串类似于字符数组，但String对象不是字符数组，用户也不能把字符数组作为字符串引用。不过，通过将字符数组作为自变量传给String构造函数。ToCharArray方法也可以从字符串中生成字符数组。
19. String对象是只读的，或者说不变的。String对象的内容永远不变。
20. 面向对象的最大好处就是可以继承（子类化）现有类的行为，同事以子类实例的身份重用原有类的代码。原有的类称为超类。当利用继承生成新的类时，新派生的类继承了超类所有的域和方法。
21. 如果子类没有特别覆盖超类的行为，子类就继承了超类的所有行为。因为子类继承了超类的域和方法。此外，子类可以添加新的域和方法，从而添加新的行为。
22. 类型强制转换（类型转换）：可以通过在表达式前面的括号中添加表示类型的前缀来进行类型强制转换。当你这样做的时候，编译器不会自动相信你，因此他要对此进行检查。好的编译器能在编译器能在编译时就告诉你是否有错，否则就添加一个运行时检查，以验证强制转换是否允许。Java语言是强类型的，类型之间的复制有很严格的规定。
23. 接口：只有方法的声明。接口的设计者所声明的方法必须被实现接口的类支持，并且，还声明了那些方法应该做什么。
24. 接口也可以声明static或者final的命名常量。此外，接口还可以声明其他嵌套接口设置类。
25. 接口也可以通过关键字exends继承。一个接口可以继承一个或者多个其他的接口，增加新的常量和方法，这些新常量和新方法必须由实现被继承接口的类实现。一个类的超类型是他继承的类和实现的接口，包括那些类和接口的所有超类型。因而，对象不仅仅是他的特定类的实例，还是包括接口在内的任何超类型的实例。一个对象能够被多态的用于他的超类和超接口，其中包括他们任何的超类型。

# 类和对象

1. Java程序设计的基本单位是类。类中含有方法，即作为运算核心的可以只利用基本类型（整形。浮点型）进行运算。
2. 类的三种成员：

1）域：与类及有关对象相关联的数据变量，其中保存着类或对象的状态。

2）方法：包含类的可执行代码并定义了对象的行为。

3）嵌套类和嵌套接口：嵌套类和嵌套接口是那些嵌套在其他类或接口中的类和接口。

1. 类修饰符：

1）public；

2）abstract：一个抽象类被认为是不完整的，不能创建实例。

3）final：final类不能有子类

4）strict folating point:定义为strctfp的类要求其中所定义的浮点运算法都进行精确运算。

一个类不能既是final，又是abstract的。

1. 域
2. 类中的变量称为域。域的声明包括类型名、域名字和可选的初始化子句，该子句用于给域赋一个初始值。
3. 当声明域时，可以指定一个对应类型的值对域进行初始化。初始化表达式或者说初始化值，不一定是常量，可以是另一个域、方法调用或包含所有这些的表达式。唯一的是初始化值的类型必须正确，并且，如果是调用方法的话，不会抛出检查性异常，因为这里没有捕获异常的代码。
4. 静态域：一个域的实例被类的所有对象共享。这种域可以声明为static，称为静态域或类变量。当你在类中声明一个static域后，就只存在这个域的一份拷贝，不管从这个类创建了多少实例。
5. Final域：final变量在其值初始化之后不能改变--任何试图对这种赋值的操作将导致编译时错误。我们已经见过用final来定义命名常量，因为常量的值是不变的。一般来说，final域用于定义类或者对象的不可变属性-也就是在类或者对象的整个生命周期内都不变的属性。
6. 如果一个final域没有初始化值，就称为空final。其用处只是简单的对域进行初始化还不够的情况下。这种域必须在类初始化之后（静态final域的情形），或者在类的对象完全构造完成之后立即初始化。
7. Private 和protected 访问修饰符只能用于成员，而不能用于类和接口本身（除非嵌套）。如果要从某个类的一段代码中访问一个成员，首先要能从这个类中访问该成员的类。有一点很重要，访问控制是在每个类（或者接口）的层次上进行的，而不是在每个对象上。这就意味着一个类的所有代码都能访问类的成员，不管该代码作用于哪个实例。
8. Abstract方法不能是static、final、synchronized、native或者strict的。
9. toString是个特殊的方法。如果某对象出现在用+运算符连接字符串的语句中，就调用toString以获得String。
10. 方法调用时，控制流就从调用方法转移到被调用方法，被调用方法的语句就按照这些语句的语义顺序执行。被调用的方法在以下三种情况下停止执行，并返回调用者：执行了return语句；已经到达方法的结尾；抛出未捕获异常。
11. 如果方法有结果要返回，他只能返回一个结果--或者是一个基本值，或者是对象引用。如果方法中有多个结果要返回，可以以多种方式进行；返回对某对象的引用，结果就被保存在该对象的域中，带一个或多个实际上是对象引用的参数，结果就保存在该对象中；返回包含结果的数组。
12. 方法的签名：方法名及参数的个数和类型
13. 编写clone方法的三个策略：

1）必须实现cloneable接口，提供一个clone方法来克隆对象。

2）通过Object类实现的clone方法，他通过把原对象的所有域拷贝到新对象中以完成简单的克隆。

3）CloneNotSupportedException异常可以用于告知类的clone方法不应该被调用。

# 接口

1.