1. Java包含基本数据类型和引用数据类型
2. 值类型变量不包含实例，实例是针对于对象的概念，当类实例化为对象的时候，这个时候可以称为是类的一个实例。
3. 优先级口诀：单目乘除位关系，逻辑三目后赋值
4. 只要给定（中序和前序）或者（中序和后序）就可以唯一确定一棵树
5. 字符串与整形做+运算，输出字符串
6. 字符型与做+运算，输出字符的ASCII值与整形之和
7. 普通类的方法可以和类名一致
8. ~逐位取反
9. &&短路于，&逻辑与，||短路或，|逻辑或，&与运算，|或运算
10. 当一个类所有的构造函数都是私有时，此类既不能被其他类实例化，也不能被其他类继承
11. 即使一个对象引用为null，也能调用它的静态成员
12. native是一个计算机函数，一个Native Method就是一个Java调用非Java代码的接口。方法的实现由非Java语言实现，比如C或C++
13. Java语言底层是由C语言实现的，在C语言中的字节用unsigned byte基本数据类型
14. Java中类的成员变量在类被实例化时会自动初始化，静态成员变量不实例化也会自动初始化
15. 递归法比递推法的执行效率低
16. instance是java的二元运算符，用来判断他左边的对象是否为右面类（接口，抽象类，父类）的实例
17. Math.cos中的参数的单位是弧度，计算余弦值
18. toRadians()是将角度转换为弧度 toDegrees()是将弧度转换为角度

|  |  |  |  |
| --- | --- | --- | --- |
|  | 基本数据类型 | 引用数据类型 | |
| 包装器类和String | 其他 |
| 函数中如何传递 | 值传递 | 引用传递 | |
| 是否可以改变内容 | 不可以 | 不可以 | 可以 |
| ==比较符 | 比较的是值 | 比较的是内存中的存放地址 | |
| equal | 无法应用 | String、Date等比较的是内容，未重写equal比较的是地址 | |

|  |  |  |
| --- | --- | --- |
|  | **栈（stack）** | **堆(heap)** |
| 存储内容 | 基本数据类型、局部变量，引用变量 | new创建的实例对象及数组 |
| 回收机制 | 用完就消失 | 靠垃圾回收机制不定期自动消除 |
| 是否有内存地址值 | 无 | 有 |
| 是否有默认初始化值 | 无 | 有 |
|  |  | 有一个专门存储字符串的字符串常量池 |

## 方法重写与重载

|  |  |  |
| --- | --- | --- |
|  | 重写 | 重载 |
| 发生的场景 | 子类继承父类 | 继承或同类 |
| 对形参的要求 | 顺序、类型、数量相同，名字无所谓 | 顺序、类型、数量不一样 |
| 对返回值的要求 | 和父类相同或为子类 | 无关联 |
| 访问修饰符 | 不能比父类严格 | 无关联 |
| 抛出异常 | 可以减少或删除，一定不能抛出新的或更广的异常 | 无关联 |
| 方法名称 | 必须相同 | 必须相同 |

## Object对象的方法

equal(…) toString() getClass() hashCode() notify() notifyAll() wait(…)

## 抽象类与接口

抽象类是什么：

抽象类不能创建实例，它只能作为父类被继承。**抽象类是从多个具体类中抽象出来的父类，它具有更高层次的抽象。**从多个具有相同特征的类中抽象出一个抽象类，以这个抽象类作为其子类的模板，**从而避免了子类的随意性。**

1.抽象类的的**抽象方法子类必须实现，除非子类也是抽象类**

2.抽象类的**构造函数、普通方法和普通成员变量和普通类要求相同**

3.不存在抽象构造函数

1.接口只能继承，而且可以多继承

2.方法只能被default（默认方法）、abstract、static三选一修饰，且只能是public

**默认方法和静态方法有方法体**

3.方法默认是被public、abstract修饰 ，**成员变量只能是public、static、final修饰**

4. 除非实现接口的类是抽象类，否则**该类要定义接口中的所有方法**。

5.接口的特性：接口、方法都是隐式抽象，不需要abstract，方法成员全是public

|  |  |  |
| --- | --- | --- |
|  | 抽象类 | 接口 |
| 可用什么修饰符修饰本身 | Public、abstract或默认（default包内可见） | Public、abstract或默认（default包内可见） |
| 抽象方法可以用什么修饰 | 只能是default、protected、public |  |
| 是否可以被实例化 | 不能 | 不能 |
| 有构造方法吗 | 可以有 | 没有 |
|  |  |  |
|  |  |  |

#### 设计层面上的区别

a. 抽象类是对一种事务的抽象，是对整个类进行抽象，包括属性，行为（方法）。接口是对行为（行为）的抽象。如果一个类继承或实现了某个抽象类，那么一定是抽象类的种类（拥有同一种属性或行为的类）。

b. 设计层面不同，抽象类作为很多子类的父类，是一种模板设计，而接口是一种规范，它是一种辐射式设计，也就是说对于抽象类，如果需要添加新的方法，可以直接在抽象方法中添加实现，子类可以不用变更，

而对于接口不行，如果接口进行了变更，那么实现它的类都需要做变更。

## 一个Java文件中能声明几个类的问题（非内部类

1.可以声明多个类

2.被public修饰的类一定和文件名相同

3.和文件名相同的类一定要有main方法（程序会从和文件名相同的类中找到入口，也就是main方法

## 环境变量

## JAVA\_HOME

JAVA\_HOME里面的内容是JDK安装目录。其实不设置JAVA\_HOME也是可以的，只要有Path就行了。但是设置JAVA\_HOME可以方便在Path中使用、更新JDK目录。比如，jdk的安装路径是C:\jdk1.5.0\_22，Path设置为C:\jdk1.5.0\_22\bin，而当我们把JAVA\_HOME设为C:\jdk1.5.0\_22时，设置Path就可以写成%JAVA\_HOME%\bin，以后当我们使用其他版本的jdk，就可以只修改JAVA\_HOME的值，此外，当我们要使用Redis等组件时，也可以灵活使用JAVA\_HOME做些改动。

它指向jdk的安装目录，Eclipse/tomcat等就是通过其找到并使用安装好的jdk。

## Path

当我们安装完jdk之后，打开cmd（在非安装目录的路径下）输入javac、java，会提示找不到命令。我们需要将命令所在的路径添加到Path系统变量中，这时，系统就可以找到可执行文件了。

Windows在查找可执行文件是这样的：在终端输入java时，系统就会先在当前目录查找java程序，如果有就会执行java，否则就会在Path中指定的路径中找，因为我们在PATH配置了...\jdk\bin，系统会在这个路径下找到Java程序并执行。否则就提示找不到命令。Path的作用其实就是方便我们使用一些命令。

## Java内存划分

JVM的内存分为程序计数器、堆、虚拟机栈、本地方法栈、方法区

程序计数器：又称PC寄存器，全程：Program Counter Register ，此处是指JVM内部的虚拟寄存器，存放着吓一跳将要执行的指定的地址

堆：由所有线程共享；运行时动态申请的内存都在堆上分配，包括new 的对象和数组，JDK8后，静态成员变量、常量池等也在堆中

虚拟机栈：每个线程都有独里的栈，存放局部变量、对象引用、操作数栈、方法出口；后进先出，被调方法结束后，对应栈区变量等立即被销毁

本地方法栈：主要与虚拟机用到native方法有关

方法区：

## Java 成员变量和局部变量

根据定义位置的不同，可以将变量分为成员变量和局部变量

成员变量：在类范围内定义的变量

局部变量：在一个方法内定义的变量

成员变量可以分为

实例属性（不用static修饰）

随着实例的存在而存在

类属性（用static修饰）

随着类的存在而存在

成员变量无需显示初始化，系统会自动对其进行默认初始化（即使没有初始化的成员变量也会被系统赋一个默认值）

局部变量可以分为

形参

在整个方法内有效

方法局部变量（在方法内定义）

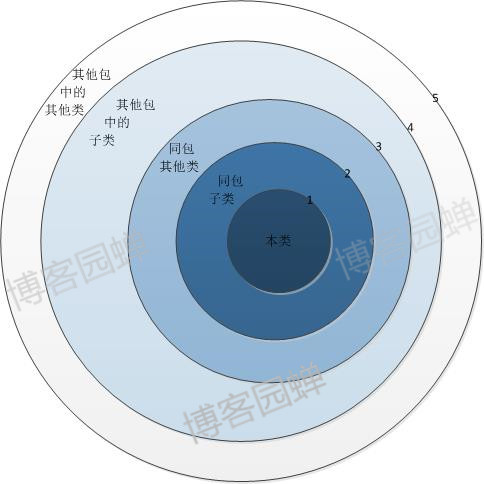
从定义这个变量开始到方法结束这一段时间内有效

代码块局部变量（代码块内定义）

从定义这个变量开始到代码块结束这一段时间内有效

局部变量除了形参外，都必须显示初始化

## 访问修饰符private、default、protected、public



Default和public可以修饰类，四种都可以修饰成员变量和方法

private只可以覆盖圆1，即只有本类可以访问；

default可以覆盖到圆3，即本类、同包子类、同包其他类都可以访问，**简单说就是与本类同包的所有类都可以访问；**

protected可以覆盖到圆4，即本类、同包子类、同包其他类、其他包中的子类都可以访问，**简单说就是与本类同包的类及其他包中的子类都可以访问；**

public可以覆盖到圆5，即本类、同包子类、同包其他类、其他包子类、其他包其他类都可以访问，**简单说就是所有类都可以访问；**