一、语法基础

1. 编程规范
2. 接口和类：一个单词：单词的首字母必须大写，举例：Student,Dog 多个单词：每个单词的首字母必须大写举例：HelloWorld，StudentName
3. 方法和变量：一个单词：单词的首字母小写，举例：main,age 多个单词：从第二个单词开始，每个单词的首字母大写，举例：studentAge,showAllNames()
4. 常量：一个单词：全部大写，举例：PI 多个单词：每个字母都大写，用\_隔开，举例：STUDENT\_MAX\_AGE
5. 注释：没写一个功能，先写注释
6. 常量
7. 字面值常量：

字符串常量：用双引号括起来的内容，举例：“hello”，“world”

整数常量：所有整数，举例：12，23

小数常量：所有小数，举例：12.34，56.78

字符常量：用单引号括起来的内容，举例：‘a’,’A’,’0’

布尔常量，较为特有，举例：true，false

空常量：举例：null

1. 自定义常量
2. 变量

从本质上讲，变量其实是内存中的一小块区域，使用变量名来访问这块区域，因此，每一个变量使用前必须要先申请（声明），然后必须进行赋值（填充内容），才能使用。通过画图说明一个变量的组成部分数据类型，变量名，变量值。

1. 数据类型



1. 类型转换
2. 显示转换：数据类型都是数值型时：由小到大的转换（int转long）
3. 强制转换：数据类型不同时的转换
4. break的使用之：跳出多层循环

wc:for(int x=0; x<3; x++) {

nc:for(int y=0; y<4; y++) {

if(y == 2) {

//break nc;

break wc;

}

System.out.print("\*");

}

System.out.println();

}

7、Java内存分配

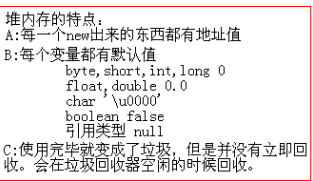
栈 存储局部变量

堆 存储new出来的东西

方法区 (面向对象部分讲)

本地方法区 (和系统相关)

寄存器 (给CPU使用)



8、基本数据类型包装类

为了方便操作基本数据类型值，将其封装成了对象，在对象中定义了属性和行为丰富了该数据的操作。用于描述该对象的类就称为基本数据类型对象包装类。

将基本数据类型封装成对象的好处在于可以在对象中定义更多的功能方法操作该数据。

常用的操作之一：用于基本数据类型与字符串之间的转换。

基本类型和包装类的对应：Byte，Short，Integer，Long，Float，Double，Character，Boolean

二、堆栈

1、栈是为执行线程留出的内存空间。当函数被调用的时候，栈顶为局部变量和一些 bookkeeping 数据预留块。当函数执行完毕，块就没有用了，可能在下次的函数调用的时候再被使用。栈通常用后进先出（LIFO）的方式预留空间；因此最近的保留块（reserved block）通常最先被释放。这么做可以使跟踪堆栈变的简单；从栈中释放块（free block）只不过是指针的偏移而已。

2、堆（heap）是为动态分配预留的内存空间。和栈不一样，从堆上分配和重新分配块没有固定模式；你可以在任何时候分配和释放它。这样使得跟踪哪部分堆已经被分配和被释放变的异常复杂；有许多定制的堆分配策略用来为不同的使用模式下调整堆的性能。

每一个线程都有一个栈，但是每一个应用程序通常都只有一个堆（尽管为不同类型分配内存使用多个堆的情况也是有的）。 直接回答你的问题：

1. 当线程创建的时候，操作系统（OS）为每一个系统级（system-level）的线程分配栈。通常情况下，操作系统通过调用语言的运行时（runtime）去为应用程序分配堆。
2. 栈附属于线程，因此当线程结束时栈被回收。堆通常通过运行时在应用程序启动时被分配，当应用程序（进程）退出时被回收。
3. 当线程被创建的时候，设置栈的大小。在应用程序启动的时候，设置堆的大小，但是可以在需要的时候扩展（分配器向操作系统申请更多的内存）。
4. 栈比堆要快，因为它存取模式使它可以轻松的分配和重新分配内存（指针/整型只是进行简单的递增或者递减运算），然而堆在分配和释放的时候有更多的复杂的 bookkeeping 参与。另外，在栈上的每个字节频繁的被复用也就意味着它可能映射到处理器缓存中，所以很快（译者注：局部性原理）。