

知识点：

**synchronized**  **关键字**  **:** 用来给对象和方法或者代码块加锁，当它锁定一个方法或者一个代码块的时候，同一时刻最多只有一个线程执行这个段代码。

**volatile：**用来确保将变量的跟新操作通知到其他线程，当把变量声明为volatile类型后，编译器与运行时都会注意到这个变量是共享的，因此不会将该变量上的操作与其他内存操作一起重排序。然而，在访问volatile变量时不会执行加锁操作，因此也就不会使执行线程阻塞，因此volatile变量是一种比 synchronized关键字更轻量级的同步机制。

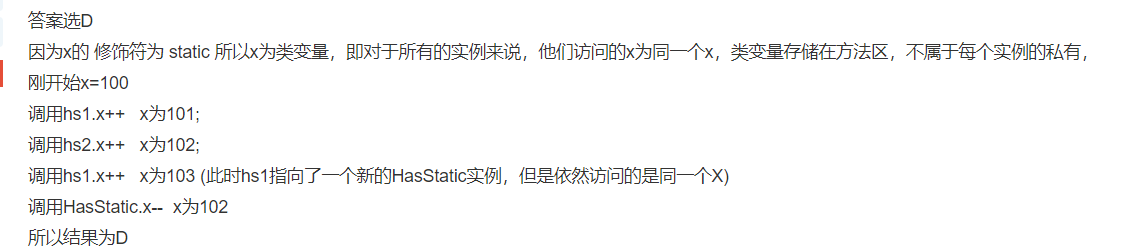
**serialize：**Java 对象序列化为二进制文件。

**static关键字：** static关键字可以修饰变量，方法，静态代码块。

注！

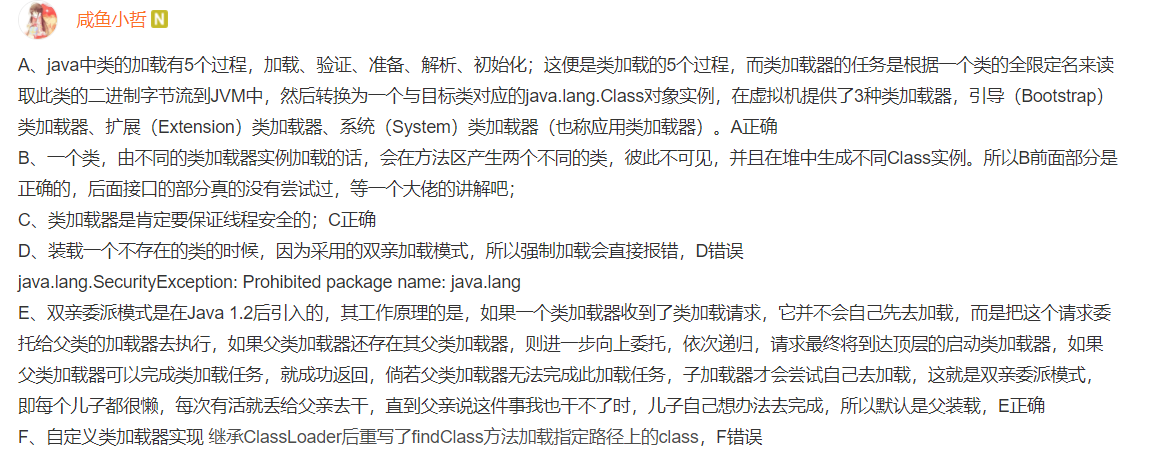


这里的main属于内部方法所以可以调用：



类加载器：





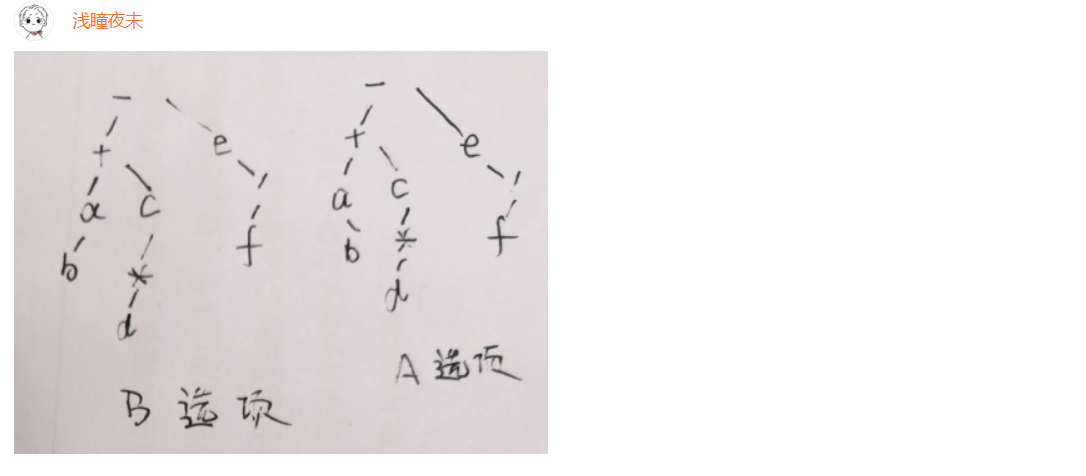
二叉树：



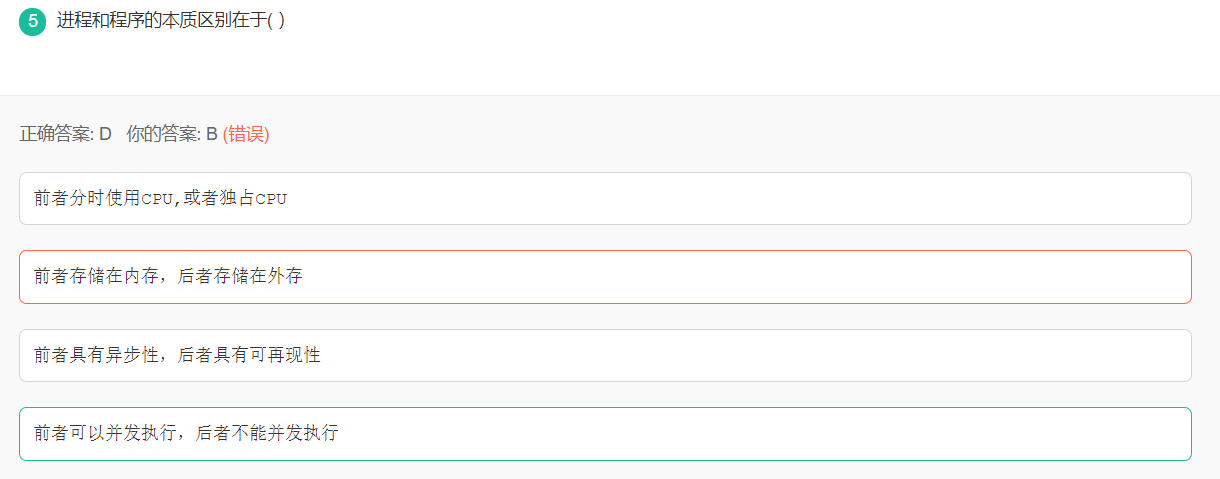


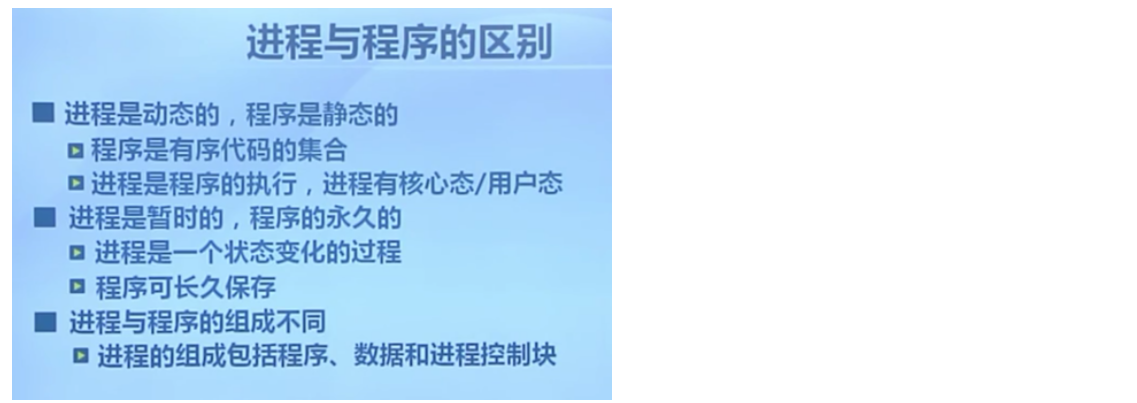
二叉树的先序后序遍历：

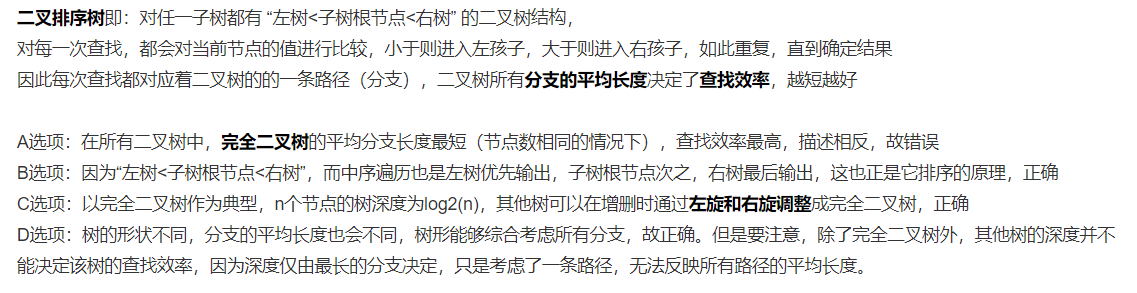




进程和程序的本质区别：



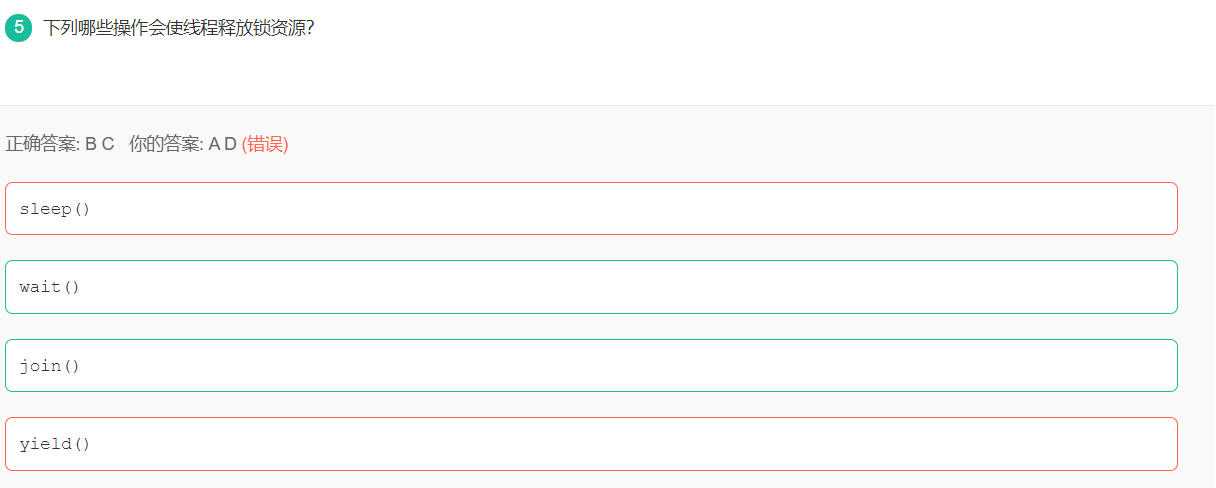


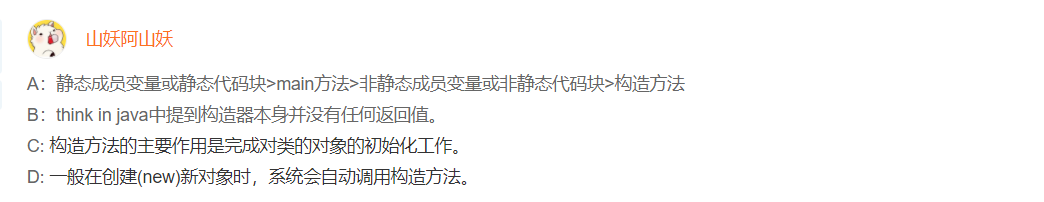
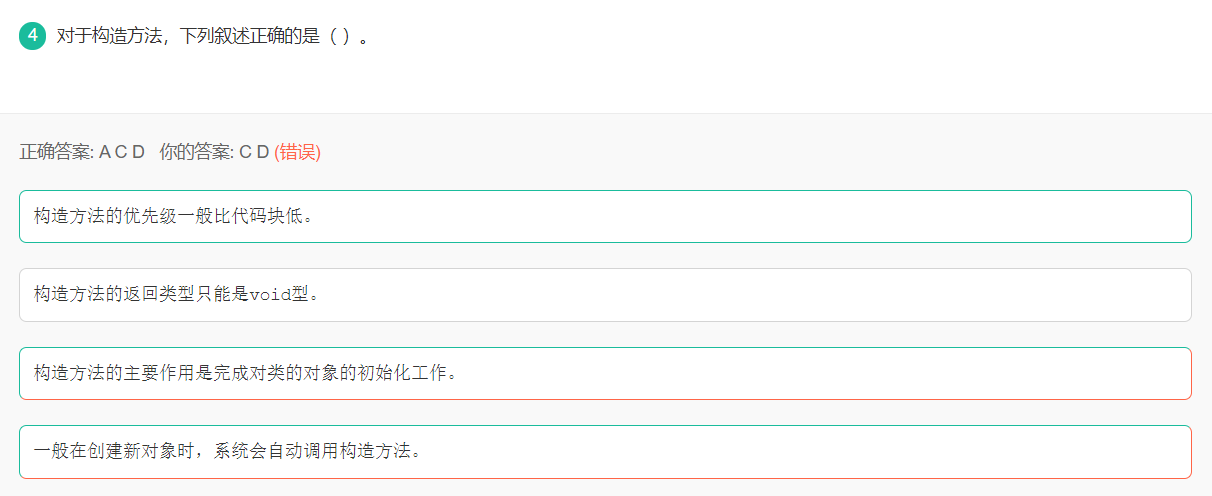
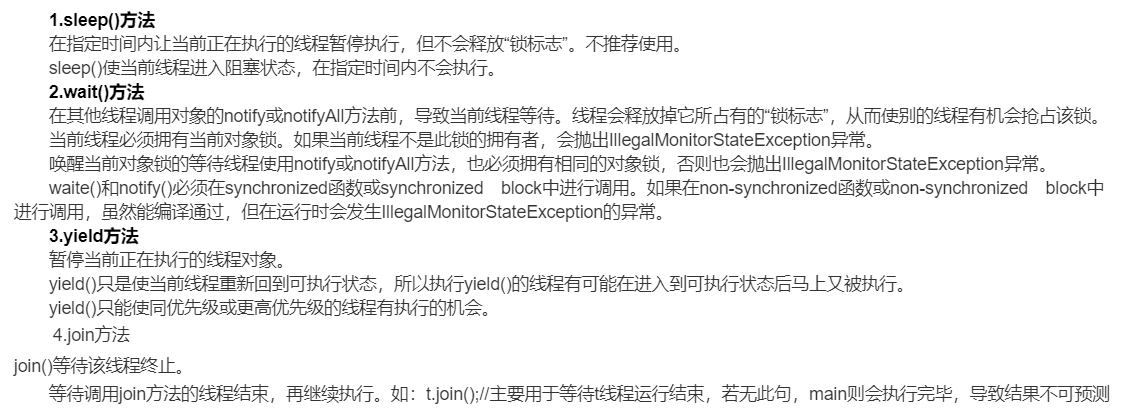


死锁和竞争的关系：

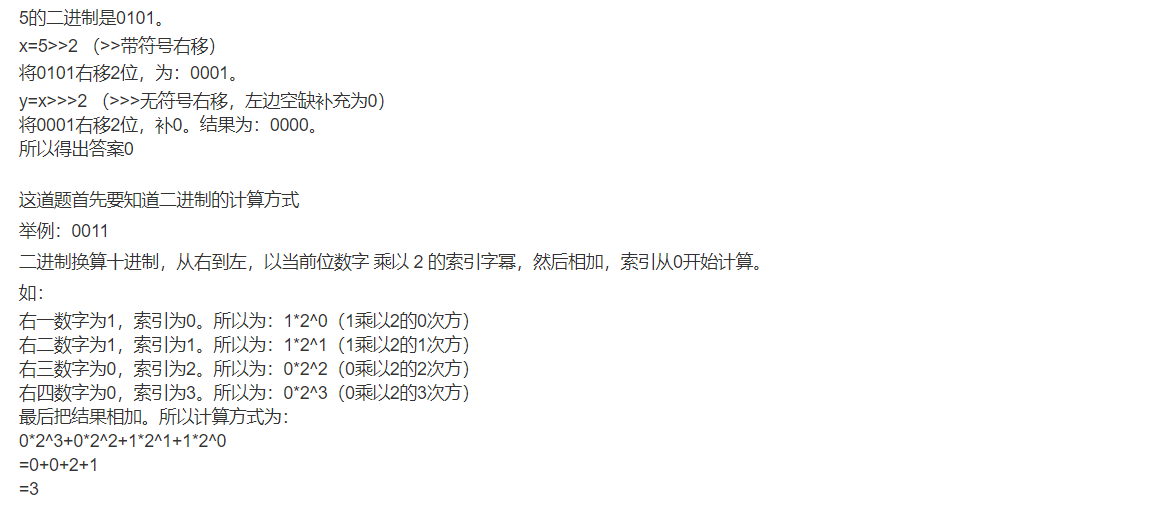
线程释放锁资源：





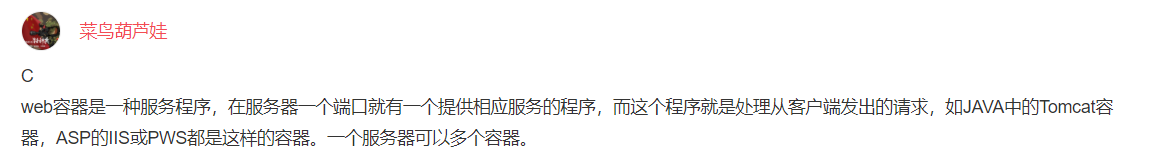
二进制：左移右移





将http转换为httpservletrequest对象：

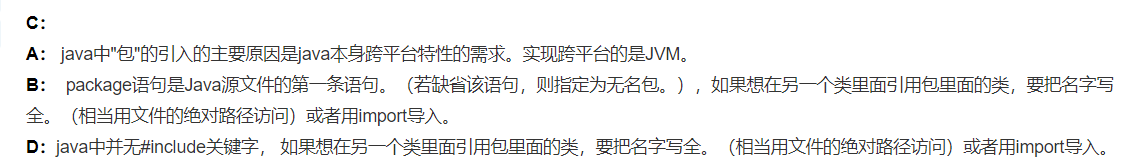


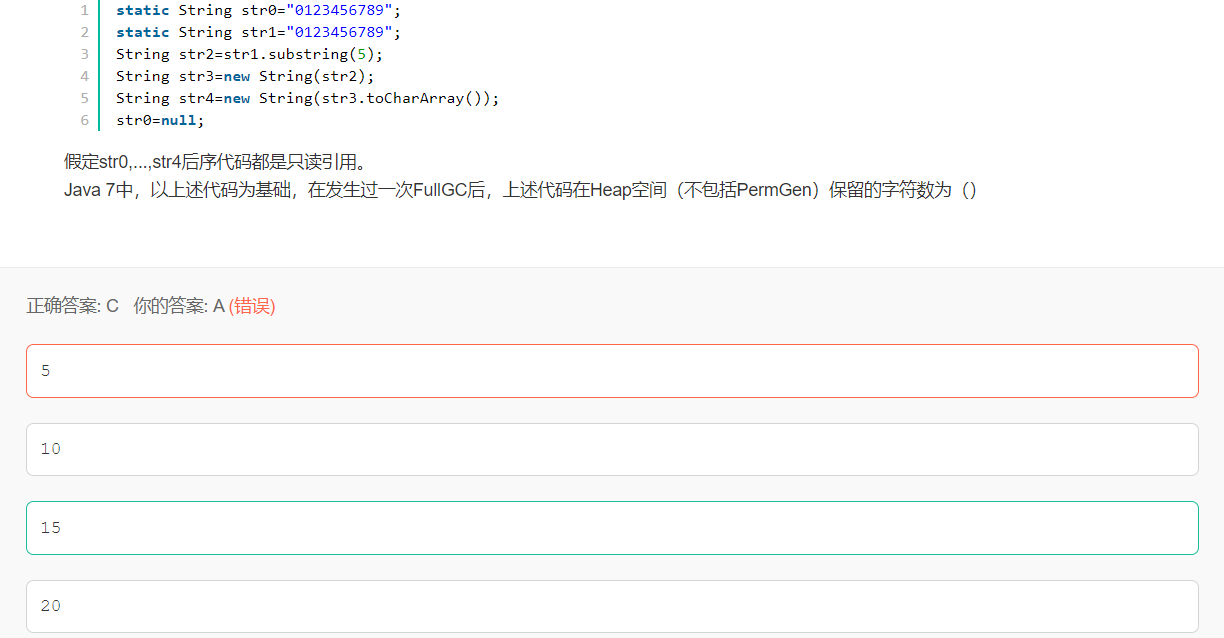


修饰符考题：

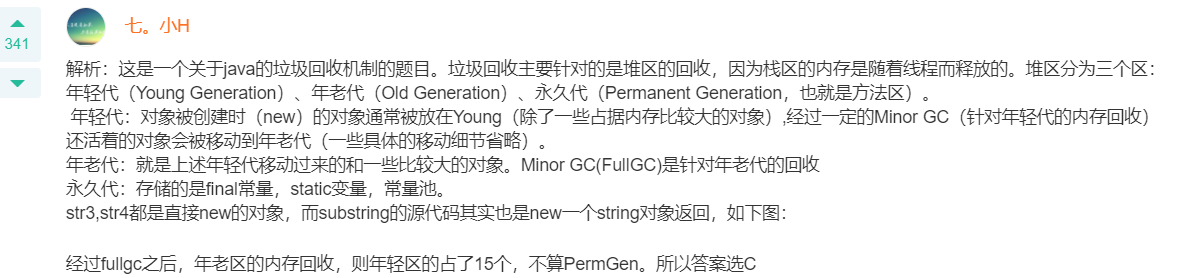




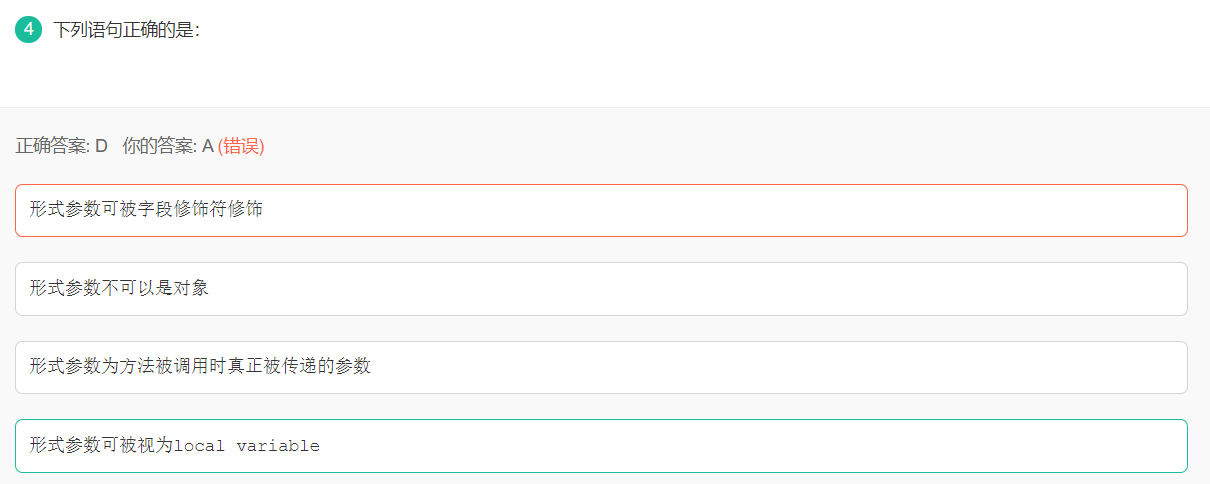








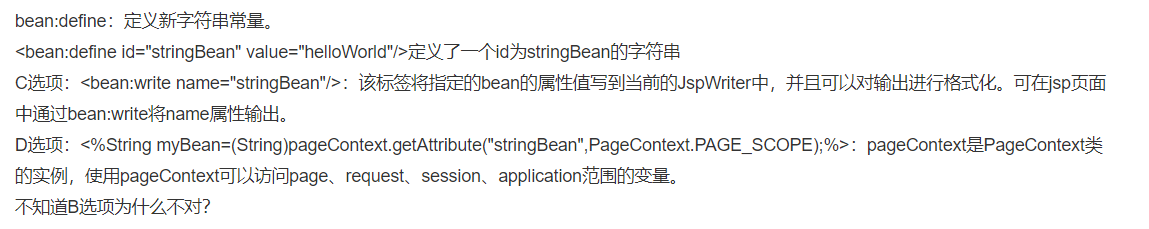
形式参数：



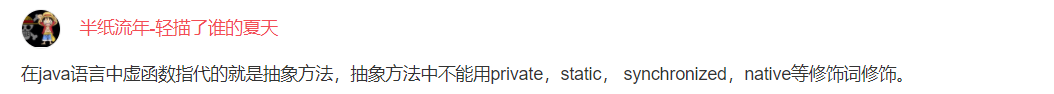


Jsp定义变量：



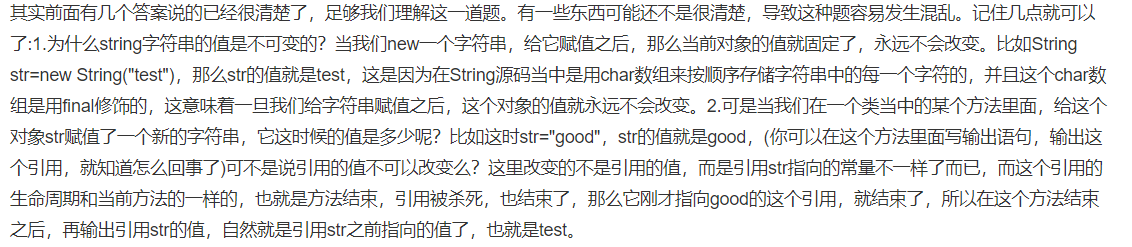




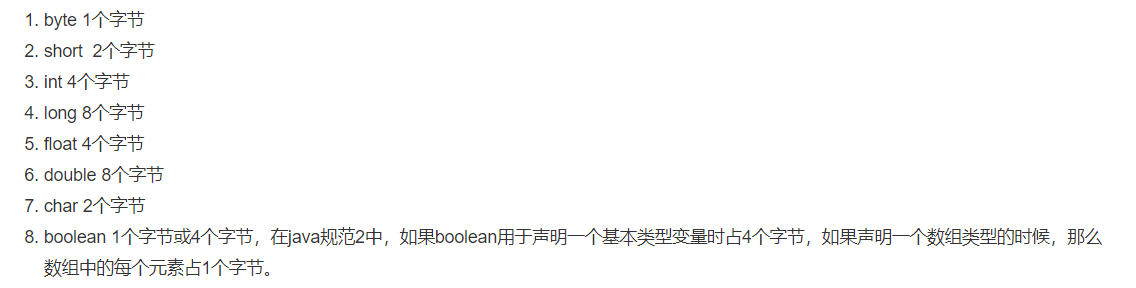


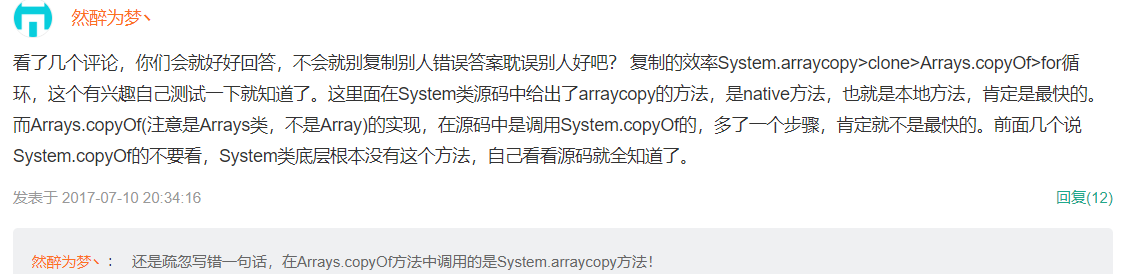
形式参数：





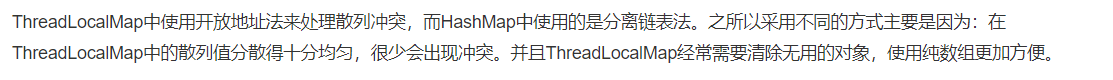
对应数据类型占的字符数：

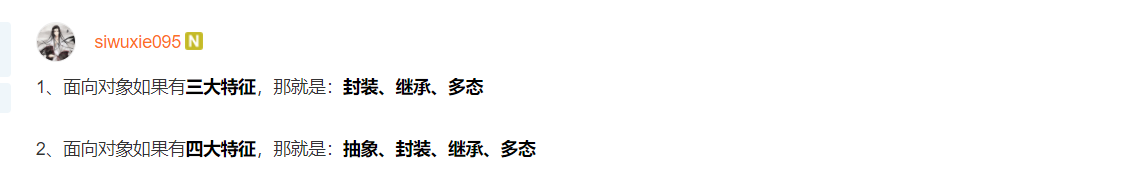


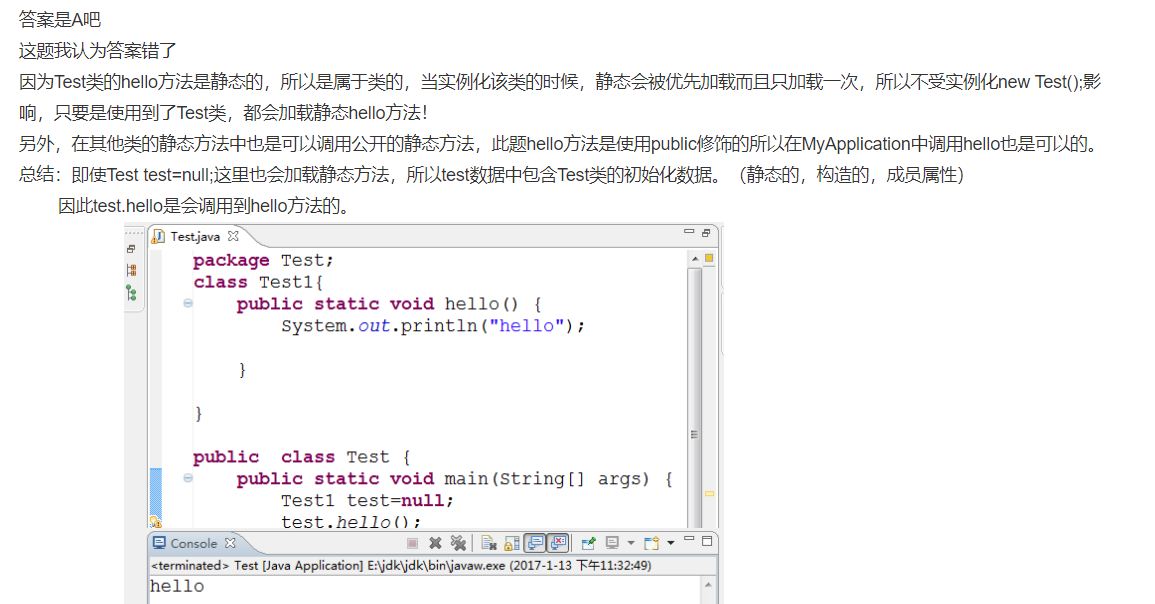


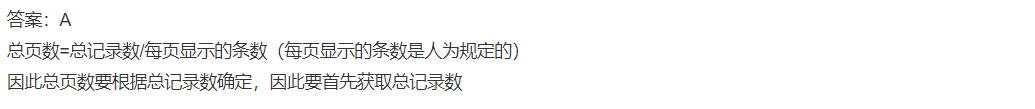
哈希冲突的开放定址法：



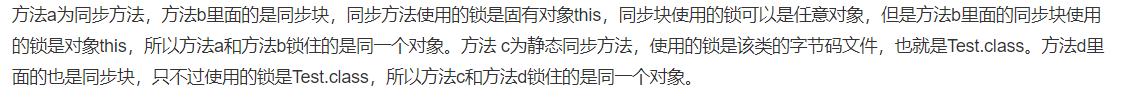
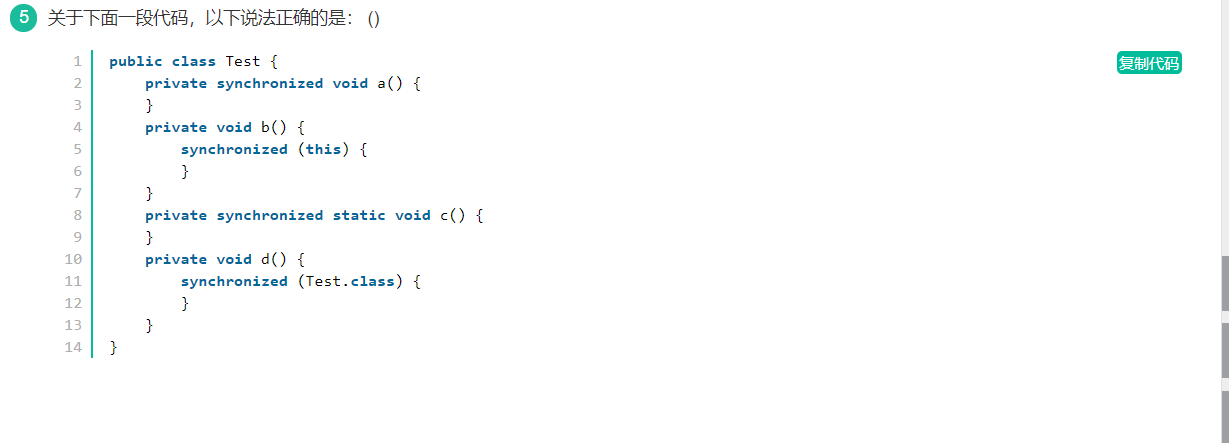


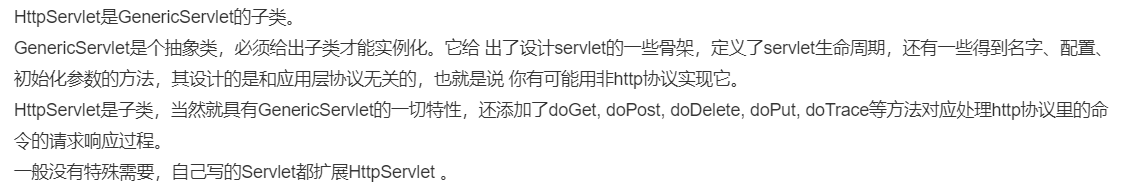
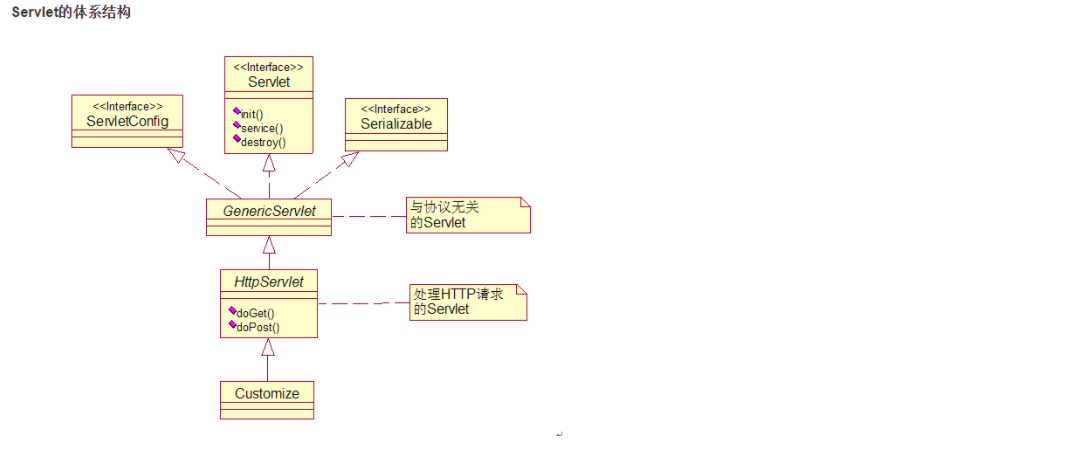




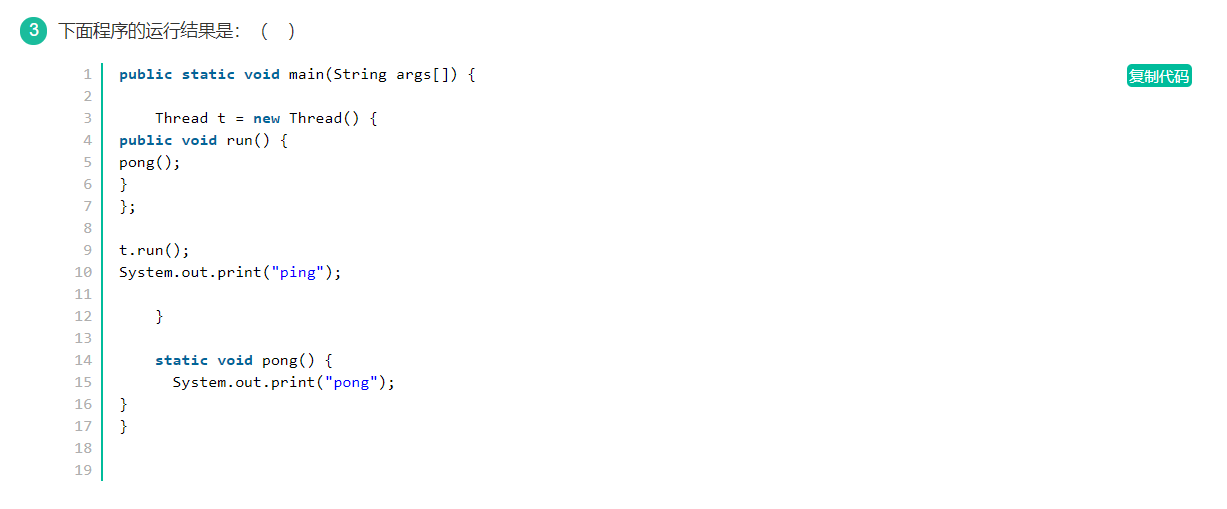
线程锁：

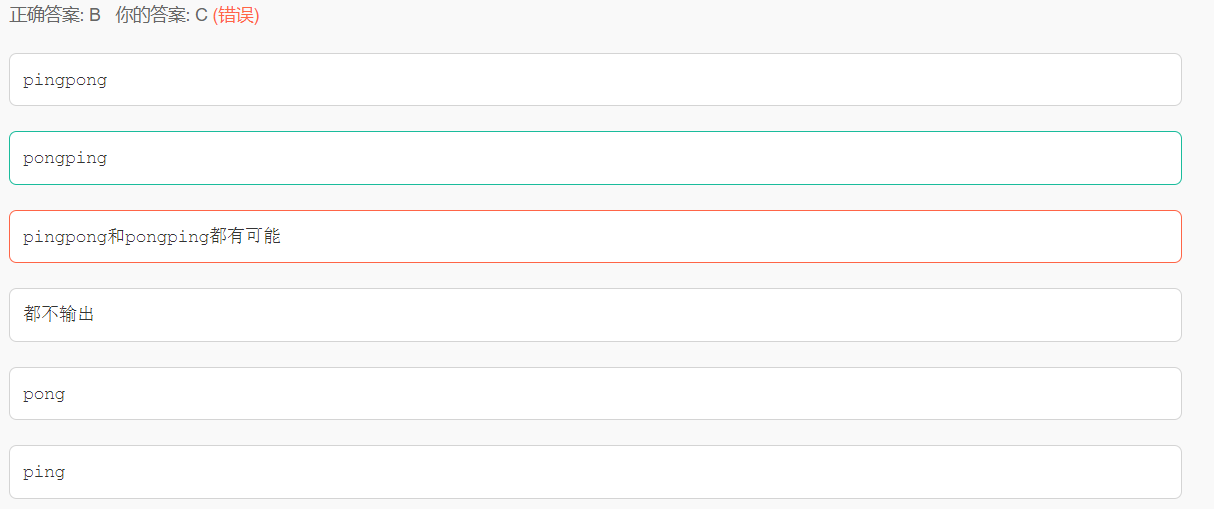


Replace：



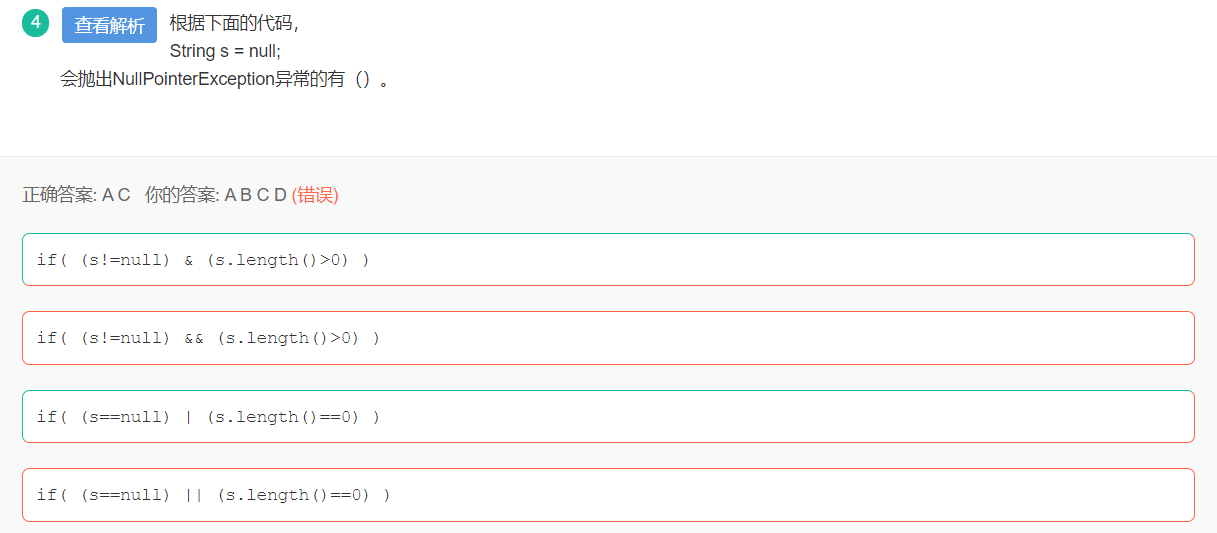
线程问题：

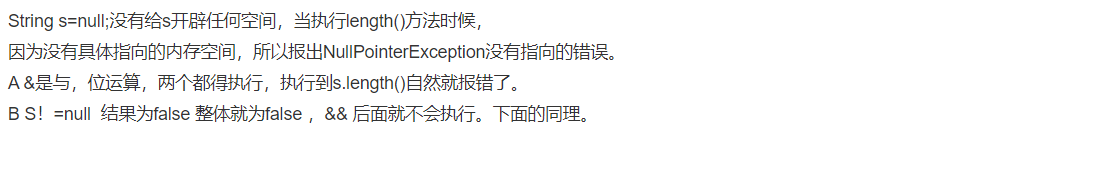




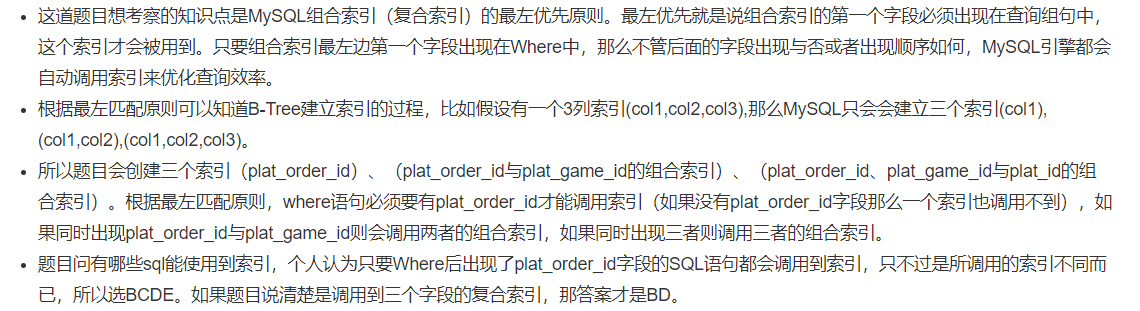
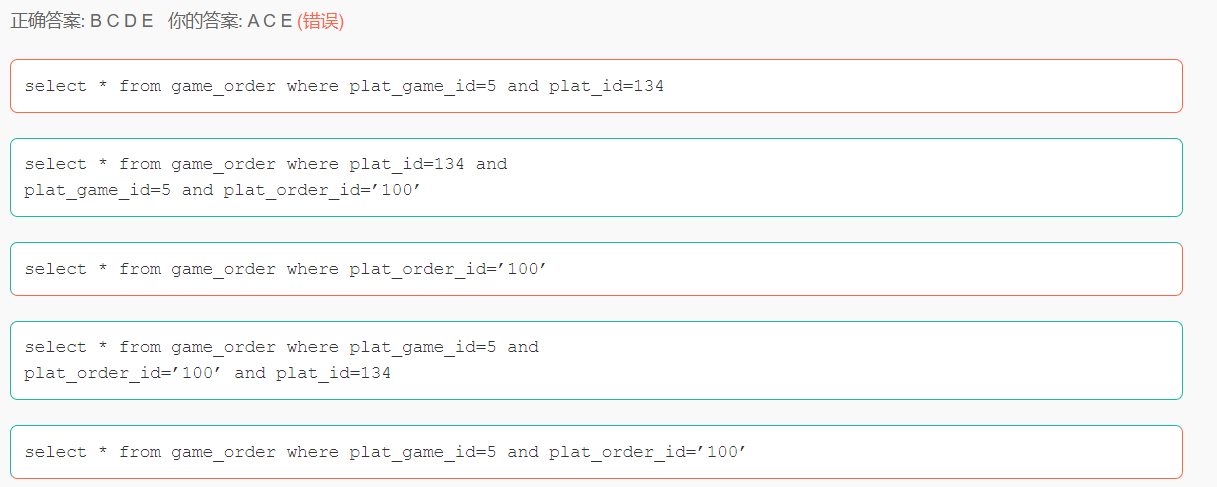
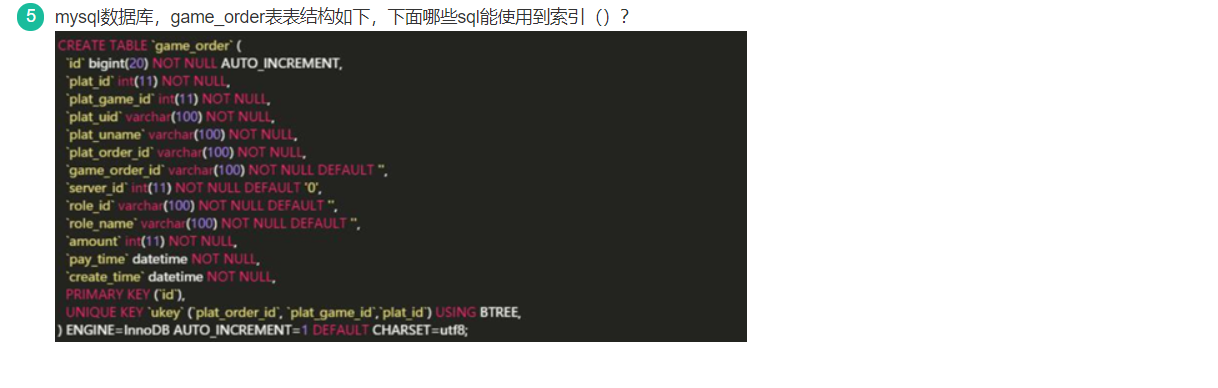


||&&表达式问题：两个符号的类似于短路不会再执行





数据库索引采用的是最左优先原则：



线程的性质：

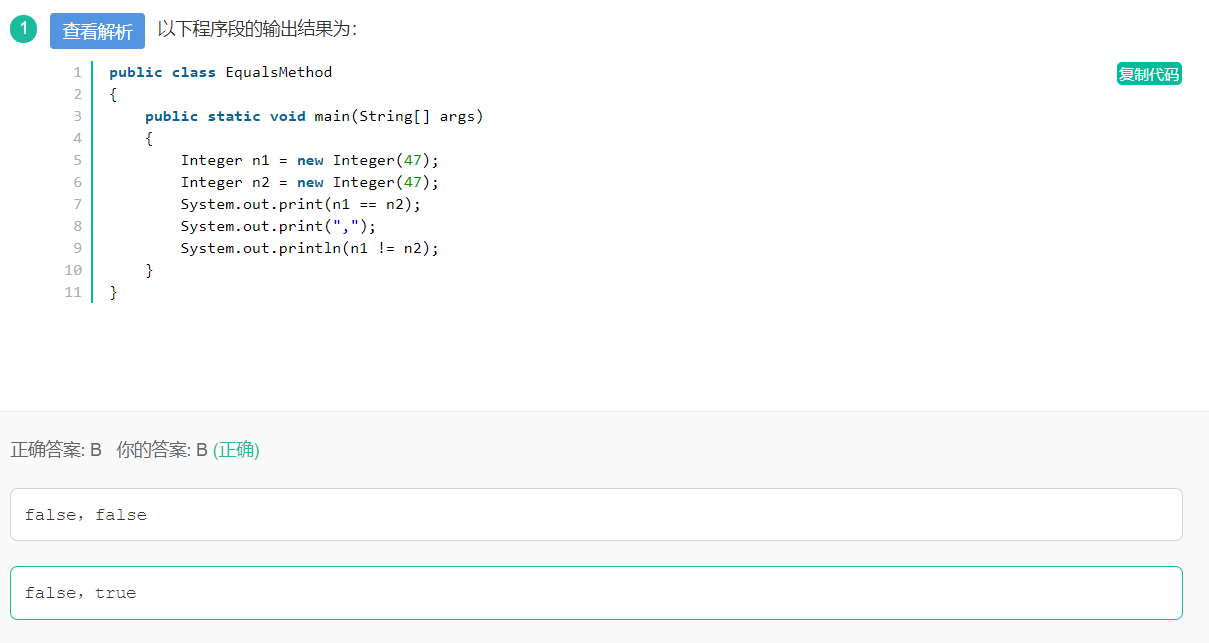


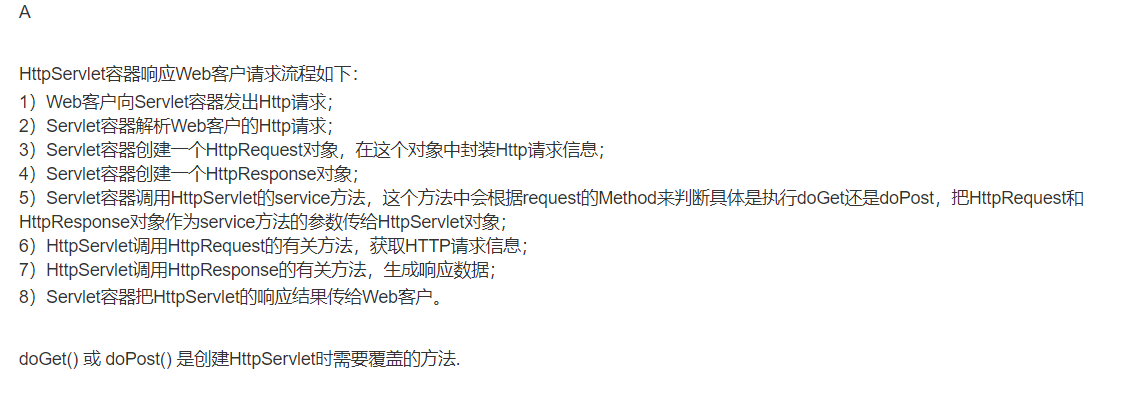
关键字volatile是线程同步的轻量级实现，所以volatile性能肯定比synchronized要好，并且只能修改变量，而synchronized可以修饰方法，以及代码块。

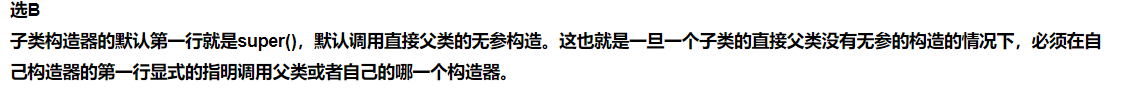
        多线程访问volatile不会发生阻塞，而synchronized会出现阻塞

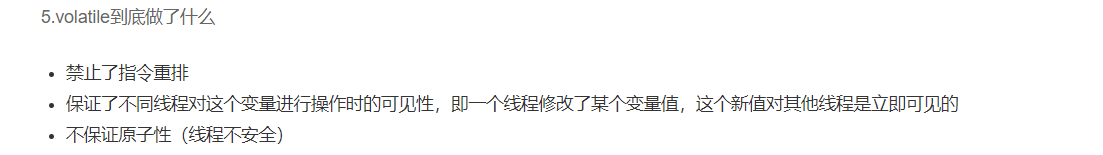
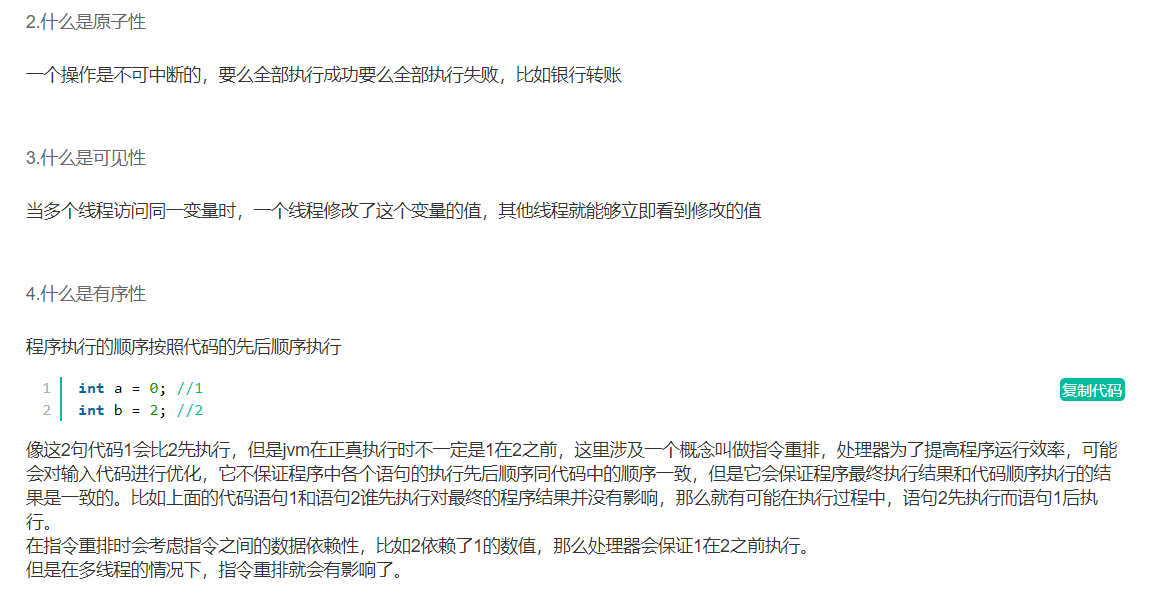
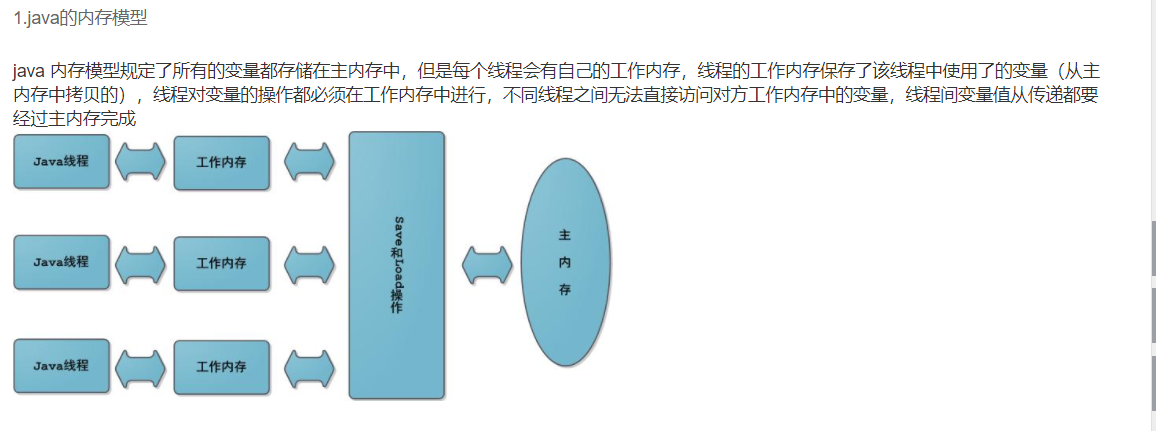
        volatile能保证数据的可见性，但不能保证原子性；而synchronized可以保证原子性，也可以间接保证可见性，因为它会将私有内存和公共内存中的数据做同步

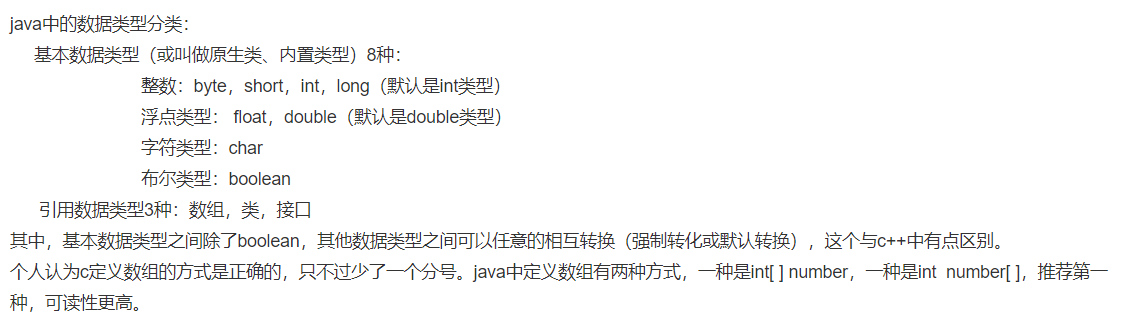
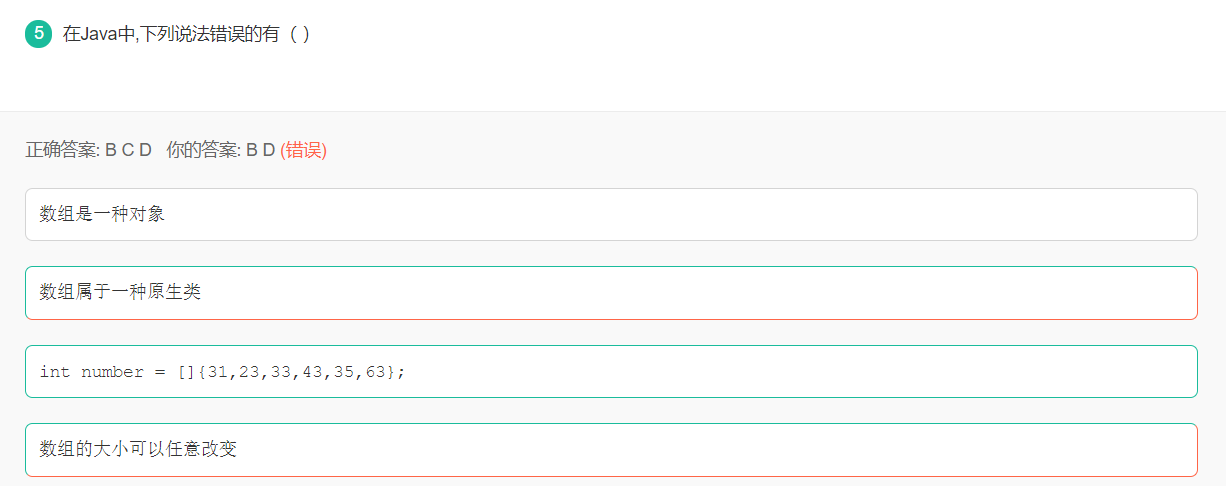
        关键字volatile解决的下变量在多线程之间的可见性；而synchronized解决的是多线程之间资源同步问题











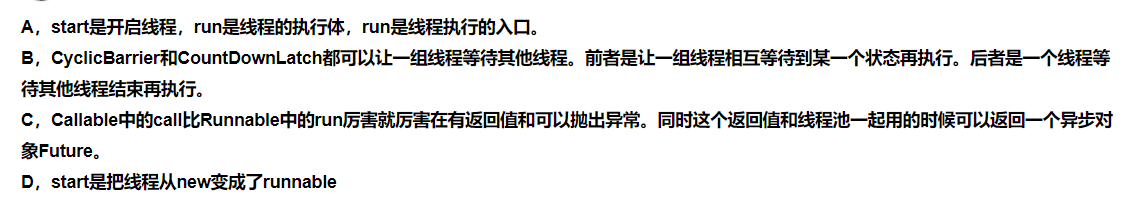
Hashmap详解：



在这里帮大家总结一下hashMap和hashtable方面的知识点吧： 1.  关于HashMap的一些说法： a)  HashMap实际上是一个“链表散列”的数据结构，即数组和链表的结合体。HashMap的底层结构是一个数组，数组中的每一项是一条链表。 b)  HashMap的实例有俩个参数影响其性能： “初始容量” 和 装填因子。 c)  HashMap实现不同步，线程不安全。  HashTable线程安全 d)  HashMap中的key-value都是存储在Entry中的。 e)  HashMap可以存null键和null值，不保证元素的顺序恒久不变，它的底层使用的是数组和链表，通过hashCode()方法和equals方法保证键的唯一性 f)  解决冲突主要有三种方法：定址法，拉链法，再散列法。HashMap是采用拉链法解决哈希冲突的。 注： 链表法是将相同hash值的对象组成一个链表放在hash值对应的槽位；    用开放定址法解决冲突的做法是：当冲突发生时，使用某种探查(亦称探测)技术在散列表中形成一个探查(测)序列。 沿此序列逐个单元地查找，直到找到给定 的关键字，或者碰到一个开放的地址(即该地址单元为空)为止（若要插入，在探查到开放的地址，则可将待插入的新结点存人该地址单元）。   拉链法解决冲突的做法是： 将所有关键字为同义词的结点链接在同一个单链表中 。若选定的散列表长度为m，则可将散列表定义为一个由m个头指针组成的指针数 组T[0..m-1]。凡是散列地址为i的结点，均插入到以T[i]为头指针的单链表中。T中各分量的初值均应为空指针。在拉链法中，装填因子α可以大于1，但一般均取α≤1。拉链法适合未规定元素的大小。     2.  Hashtable和HashMap的区别： a)   继承不同。  public class Hashtable extends Dictionary implements Map public class HashMap extends  AbstractMap implements Map b)  Hashtable中的方法是同步的，而HashMap中的方法在缺省情况下是非同步的。在多线程并发的环境下，可以直接使用Hashtable，但是要使用HashMap的话就要自己增加同步处理了。 c)  Hashtable 中， key 和 value 都不允许出现 null 值。 在 HashMap 中， null 可以作为键，这样的键只有一个；可以有一个或多个键所对应的值为 null 。当 get() 方法返回 null 值时，即可以表示 HashMap 中没有该键，也可以表示该键所对应的值为 null 。因此，在 HashMap 中不能由 get() 方法来判断 HashMap 中是否存在某个键， 而应该用 containsKey() 方法来判断。 d)  两个遍历方式的内部实现上不同。Hashtable、HashMap都使用了Iterator。而由于历史原因，Hashtable还使用了Enumeration的方式 。 e)  哈希值的使用不同，HashTable直接使用对象的hashCode。而HashMap重新计算hash值。 f)  Hashtable和HashMap它们两个内部实现方式的数组的初始大小和扩容的方式。HashTable中hash数组默认大小是11，增加的方式是old\*2+1。HashMap中hash数组的默认大小是16，而且一定是2的指数。   注：  HashSet子类依靠hashCode()和equal()方法来区分重复元素。      HashSet内部使用Map保存数据，即将HashSet的数据作为Map的key值保存，这也是HashSet中元素不能重复的原因。而Map中保存key值的,会去判断当前Map中是否含有该Key对象，内部是先通过key的hashCode,确定有相同的hashCode之后，再通过equals方法判断是否相同。

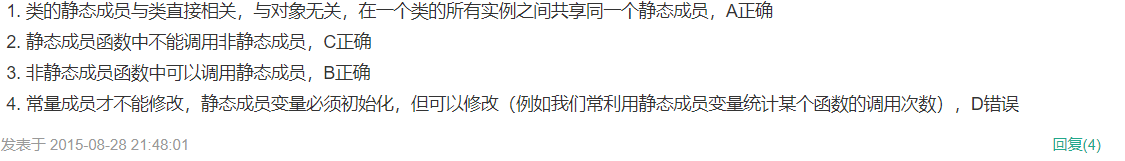
多线程：



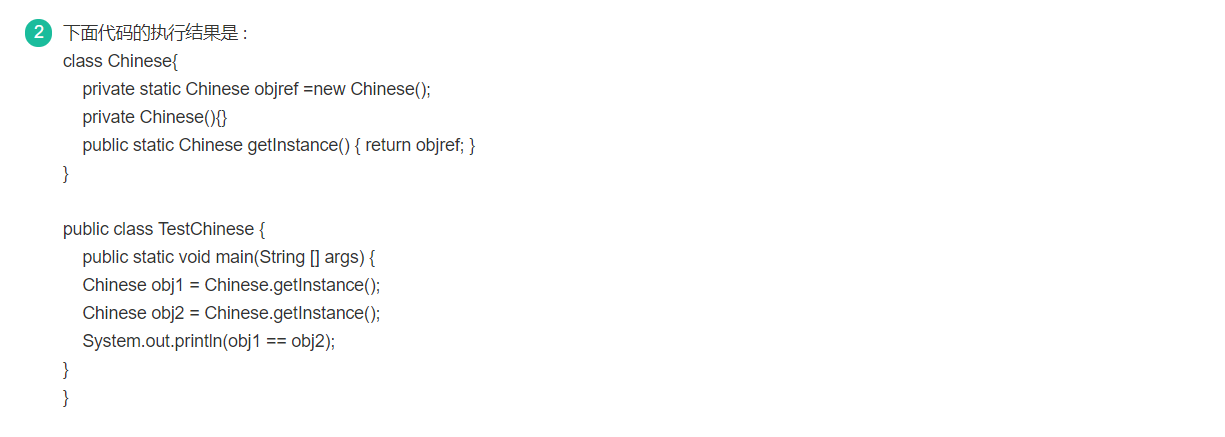


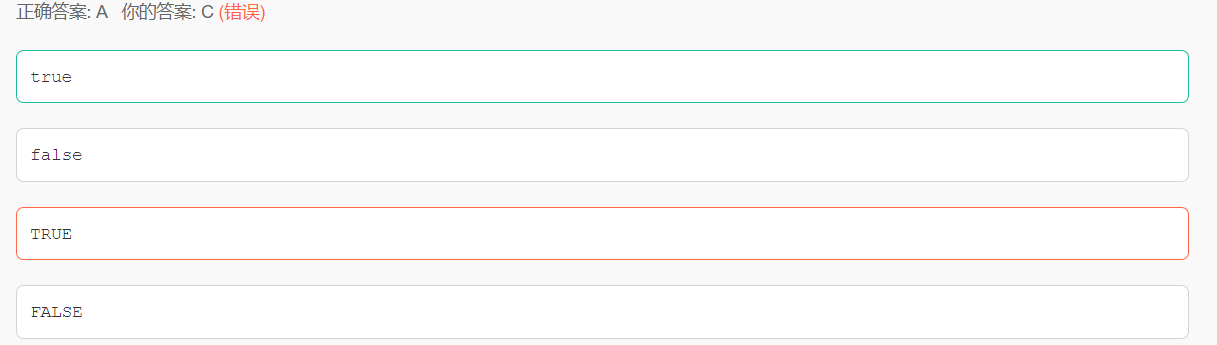






系统里==返回的true应该是大写的TRUE





构造器创建是public 类名（）{}

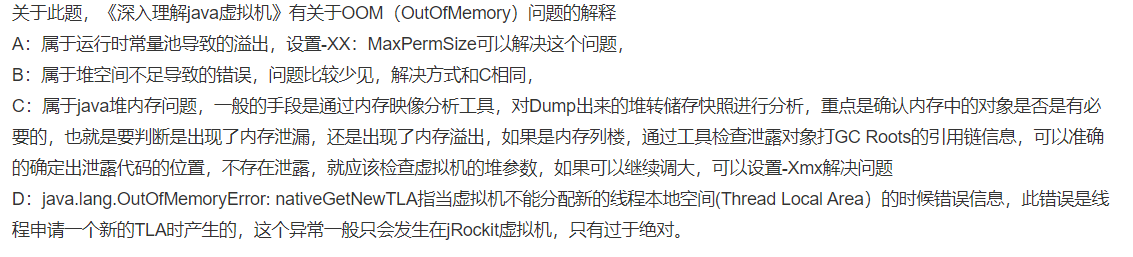




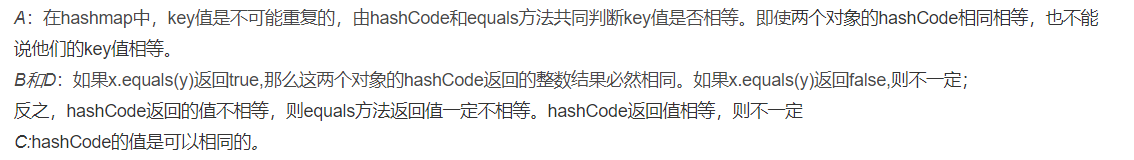
抽象类可以有构造函数：1.8以后default修饰可以有方法体





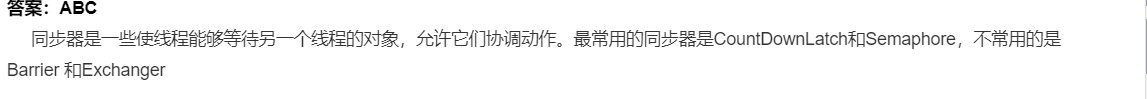


Hashcode()和equals的区别：equals返回true时hashcode一定相等，但hashcode相同的时候equals不一定相等



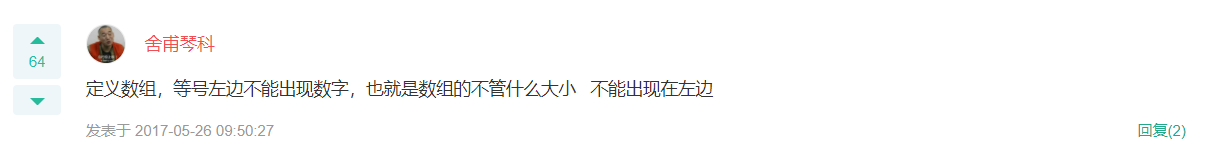
JDK用于并发的同步器：





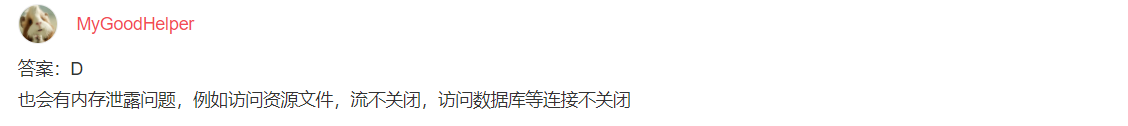
二维数组合法的是：



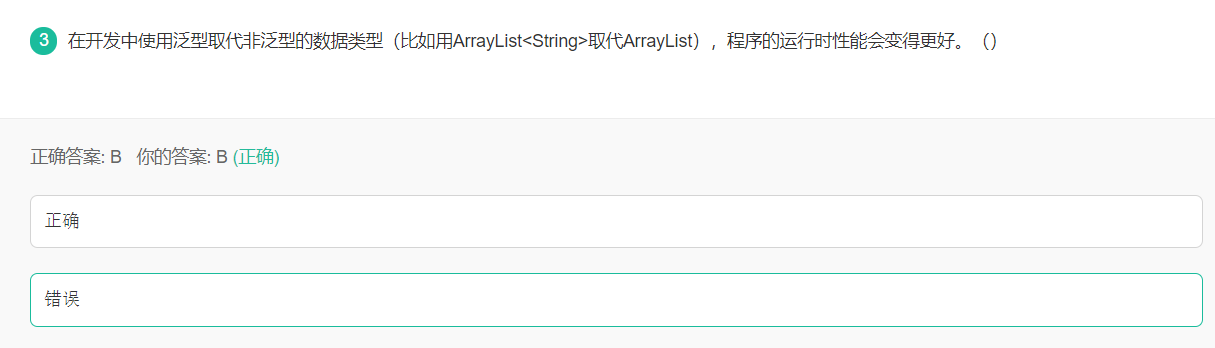


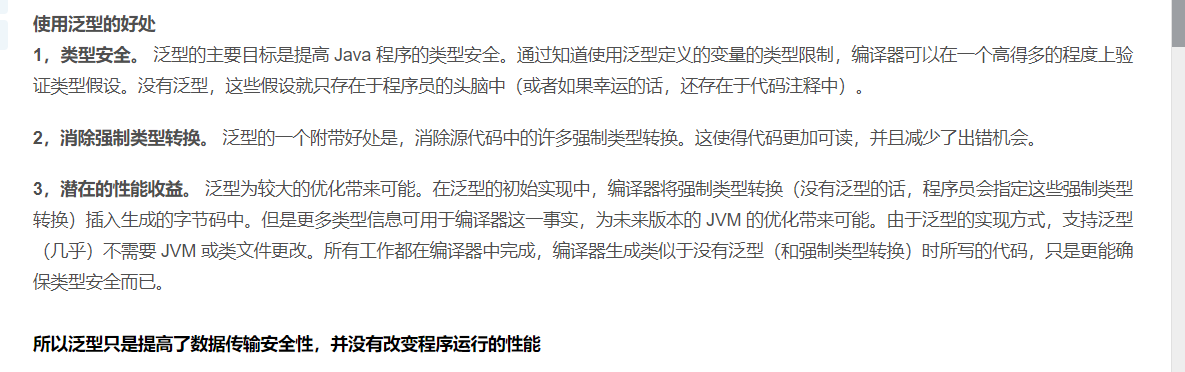
垃圾收集的语言也存在泄露问题：





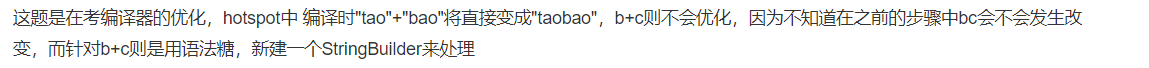
泛型并不能提高性能：







在判断过程中的想加会新建一个stringbuffer来处理：



String a = "tao"+"bao";

       String b = new String("taobao");

      System.out.println(a==MESSAGE); //true

      System.out.println(b==MESSAGE);  //false

      b = b.intern();

      System.out.println(b==MESSAGE); //true

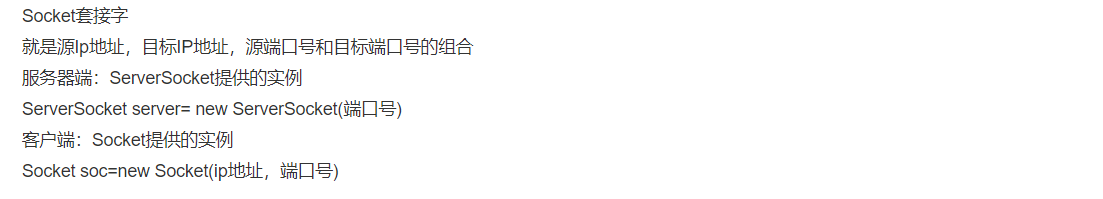
System. out .println(a==a.intern());  //true

什么操作会使得当前线程停止：



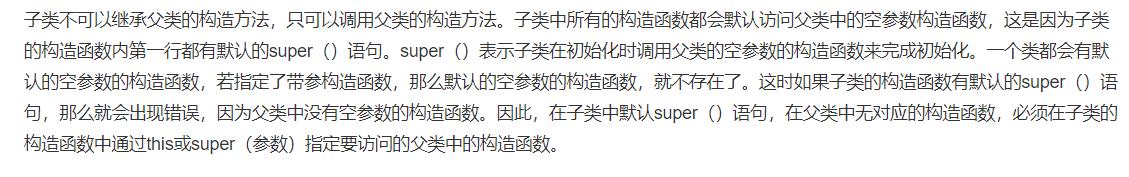
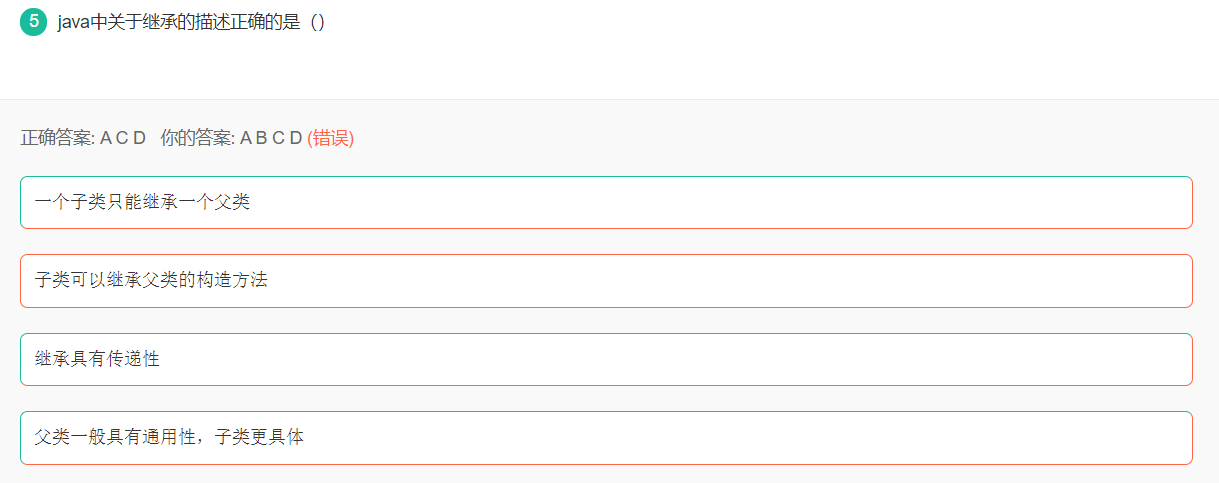








继承关系：

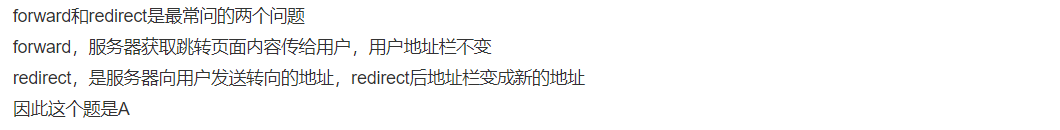


转发和重定向的区别：

forward和redirect是最常问的两个问题

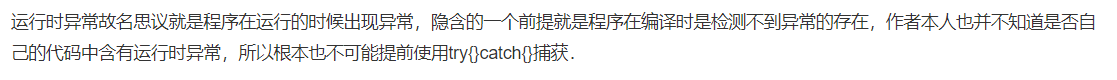
forward，服务器获取跳转页面内容传给用户，用户地址栏不变

redirect，是服务器向用户发送转向的地址，redirect后地址栏变成新的地址

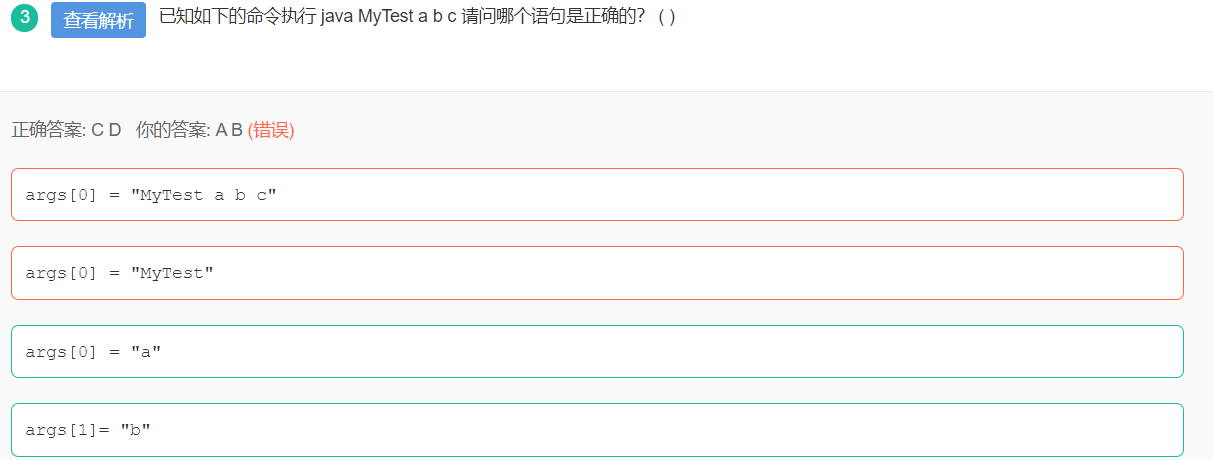


Java异常类：



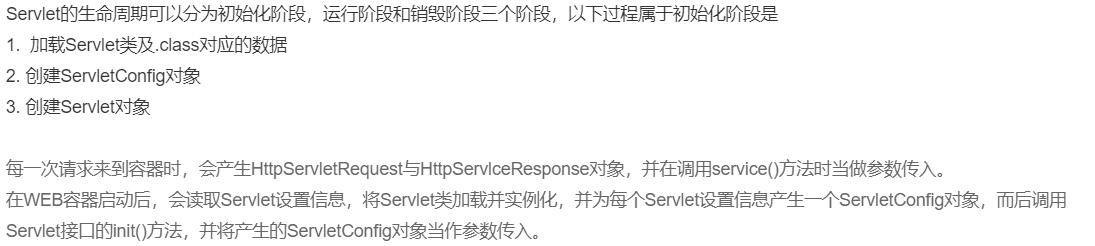


Java mytest a b c 就是往java的[]args里面存入abc三个元素：

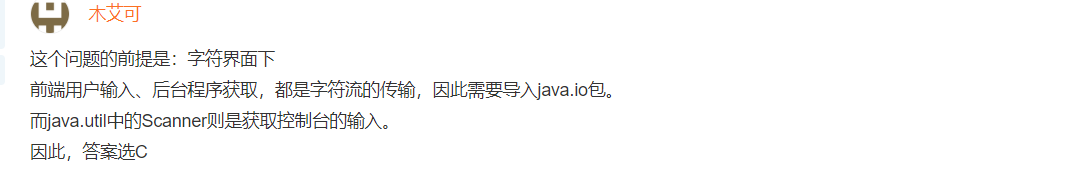
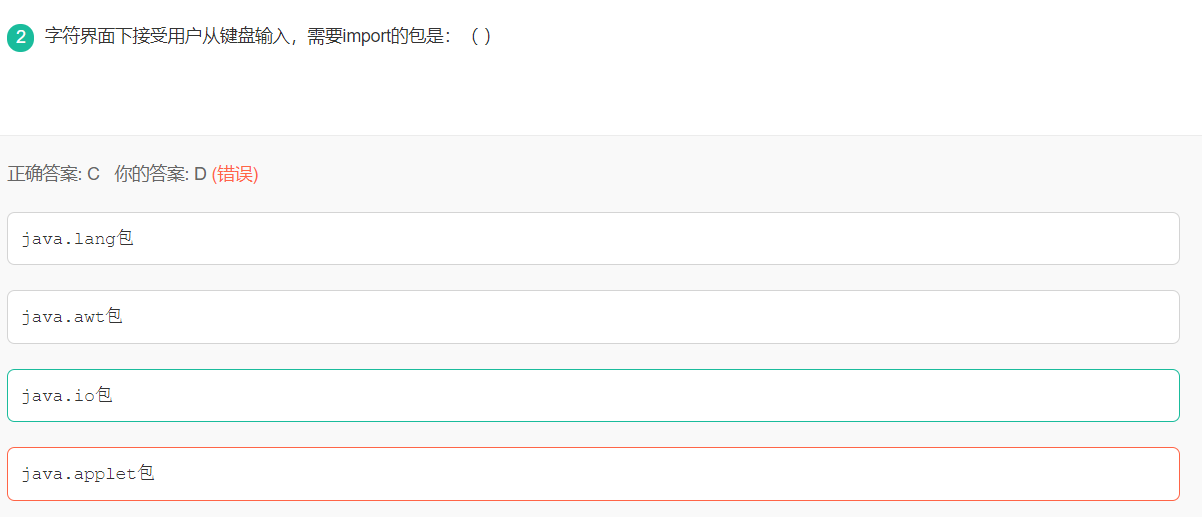


Sevrlet生命周期：

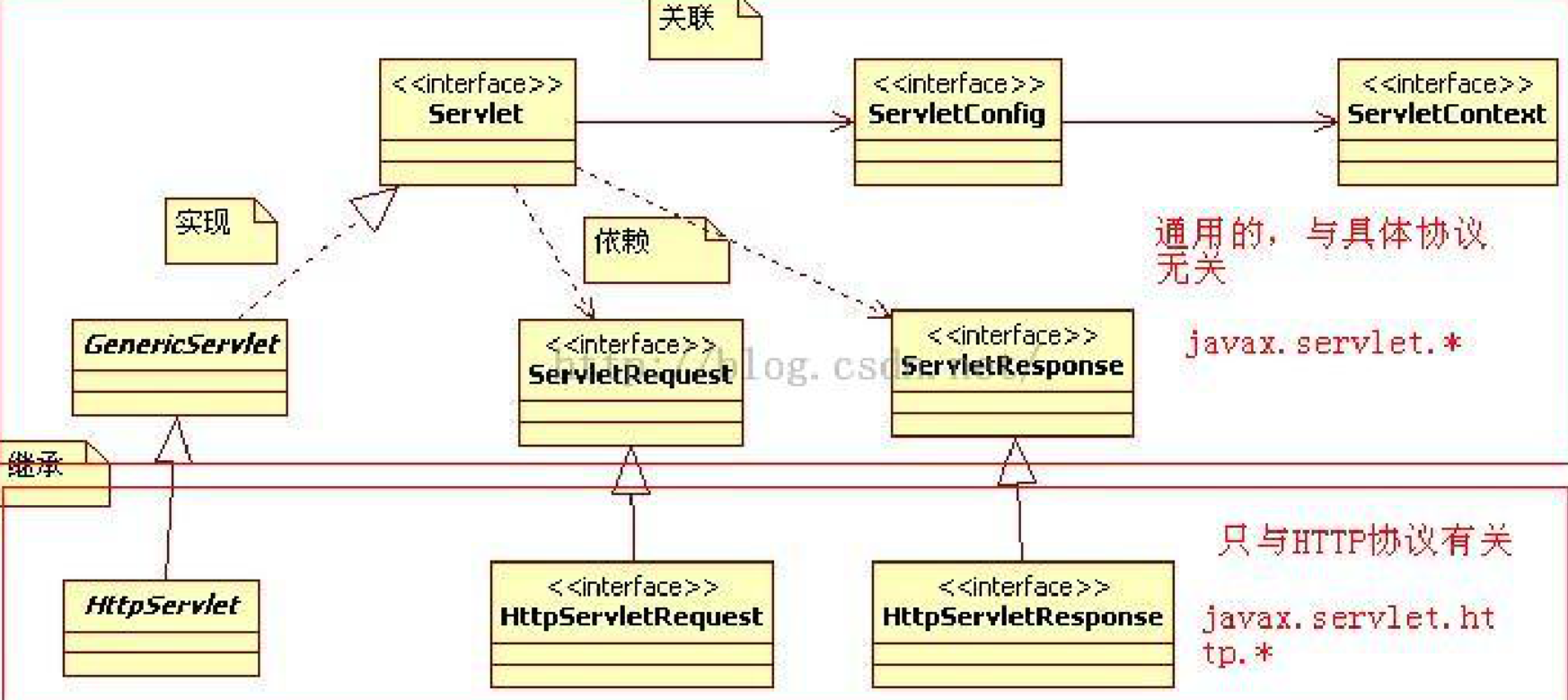








Http service

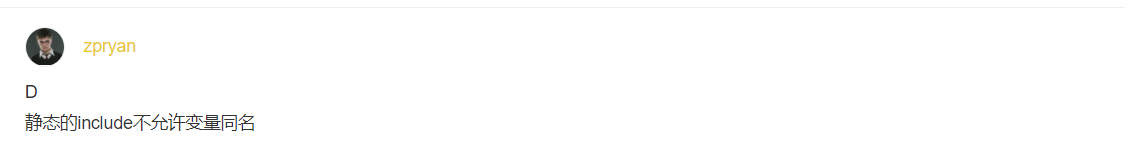


记住只有（0\d）表示从0到9没有（9\d）



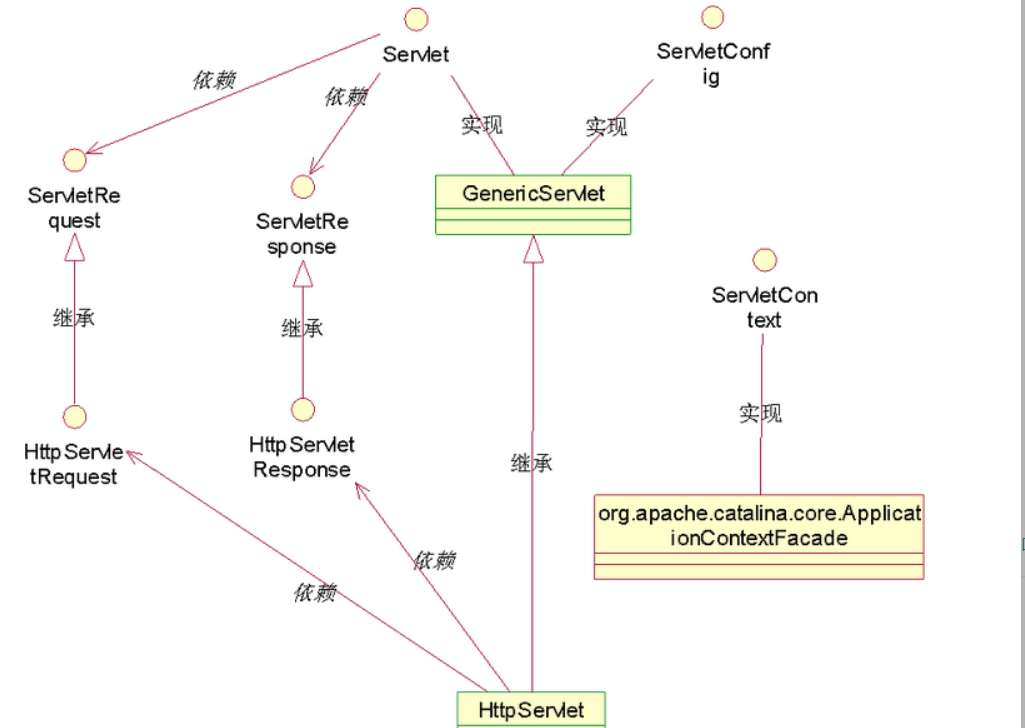
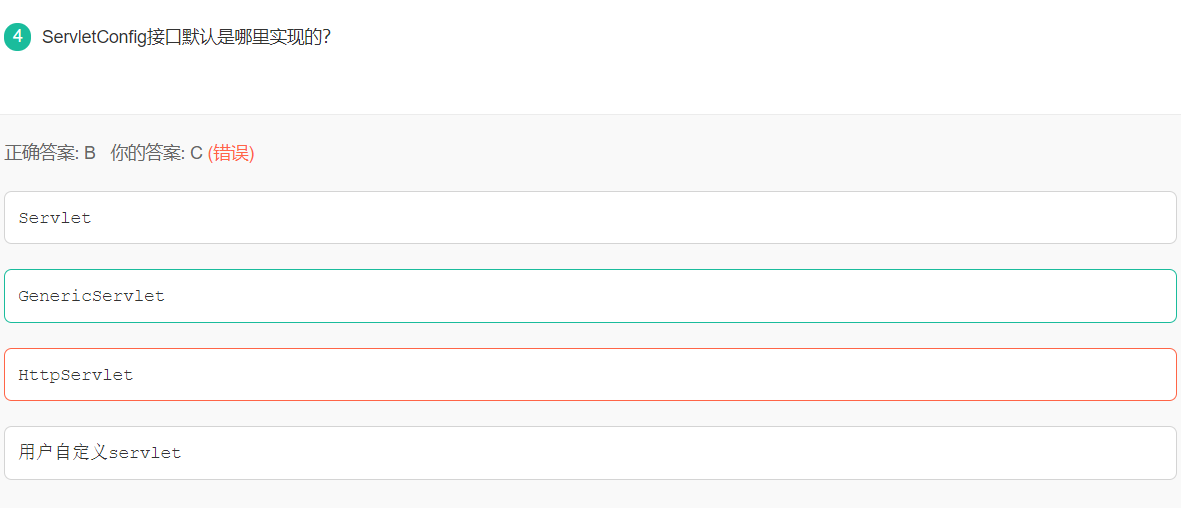


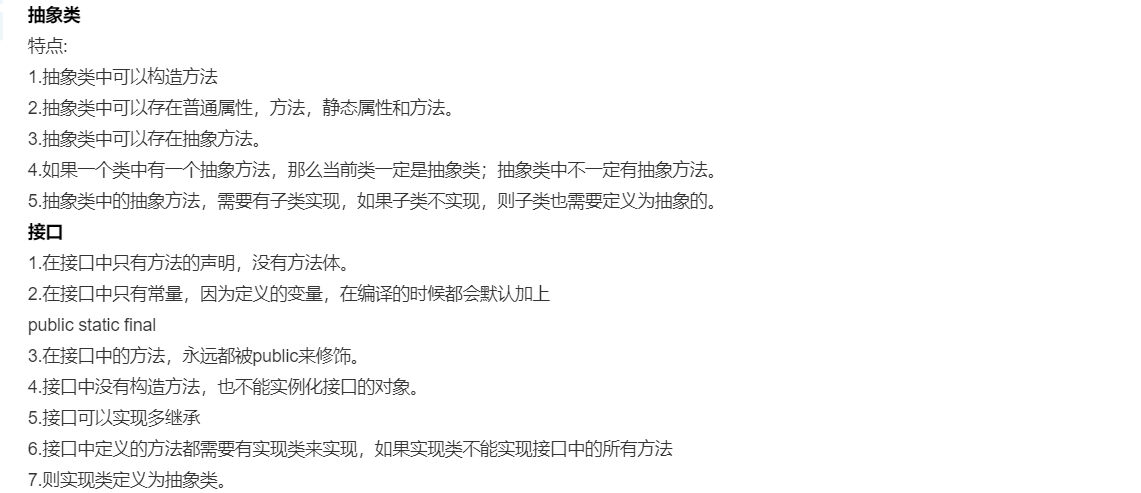


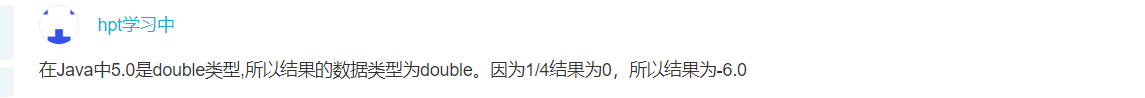


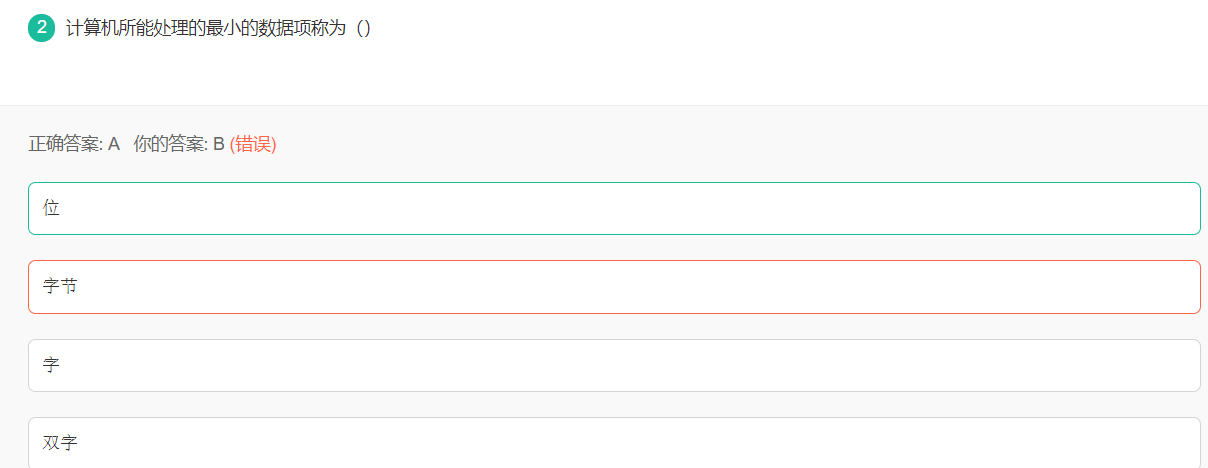
genericservlet实现了servlet和servletconfig继承了httpservlet

Servletrequest和servletresponse分别继承了httpservletrequest和httpresponse依赖servlet





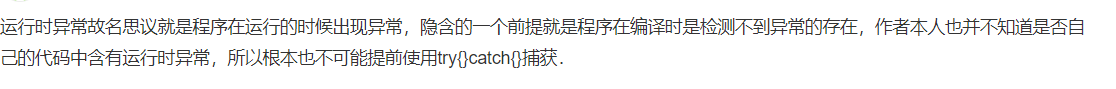






第二次错！！！！记住error是不需要捕获的。

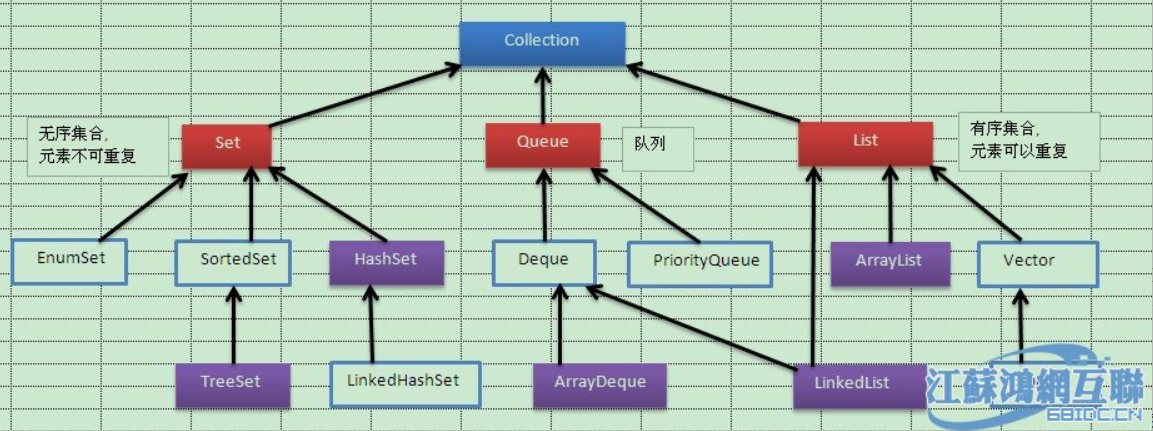
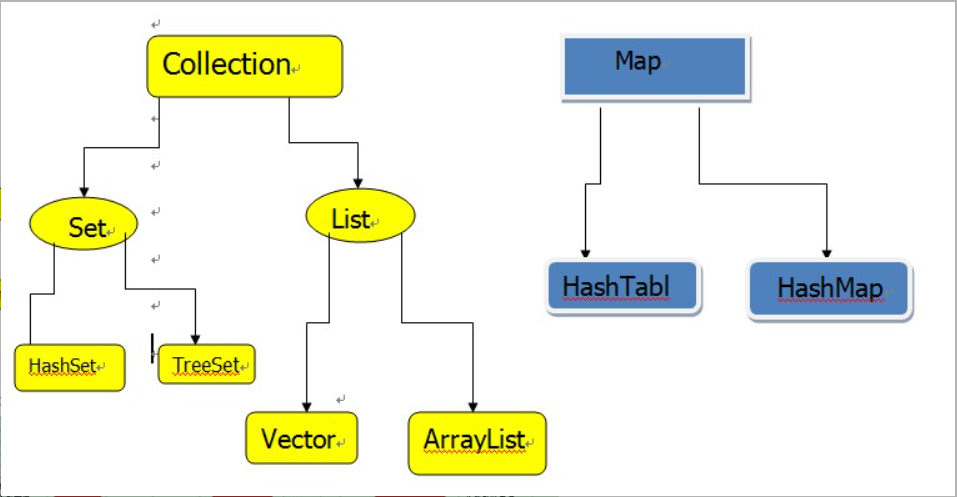






集合的关系：





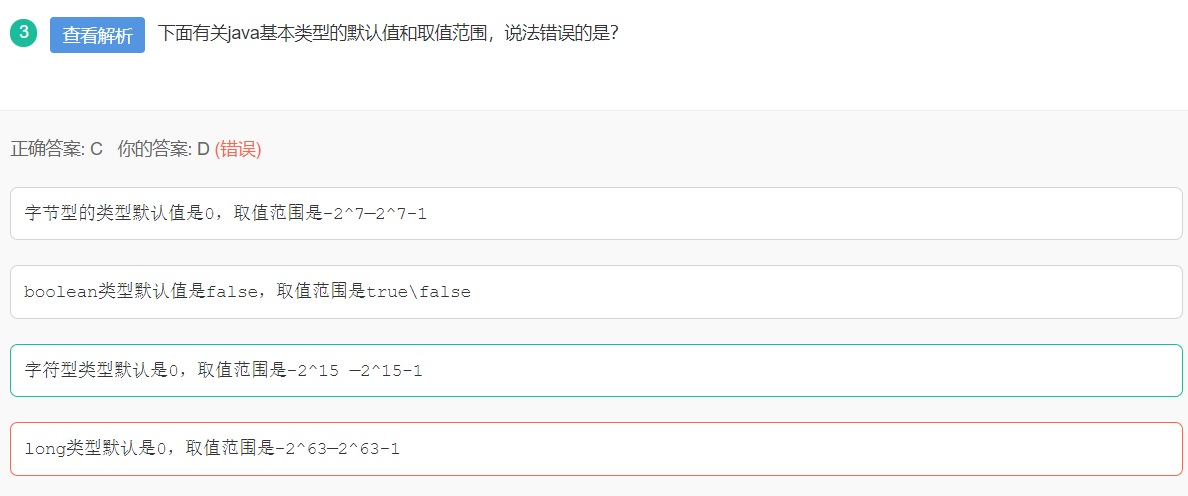
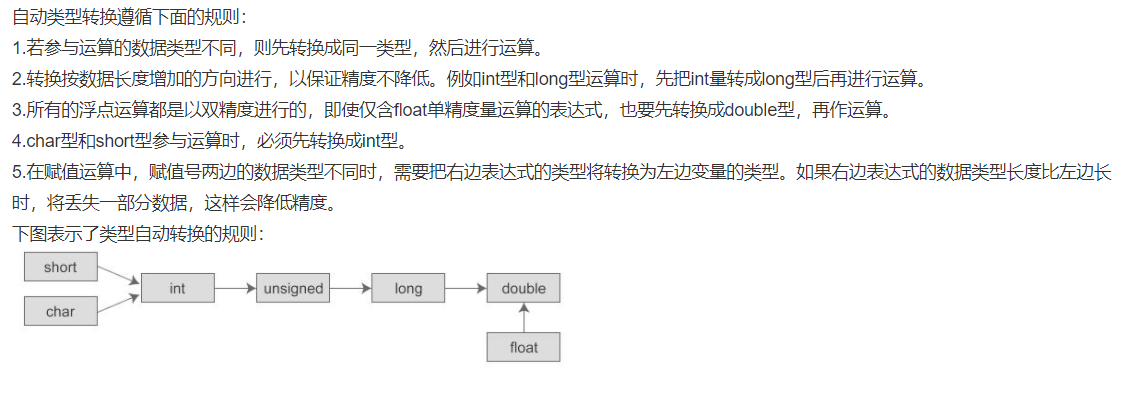
守护线程Daemon：

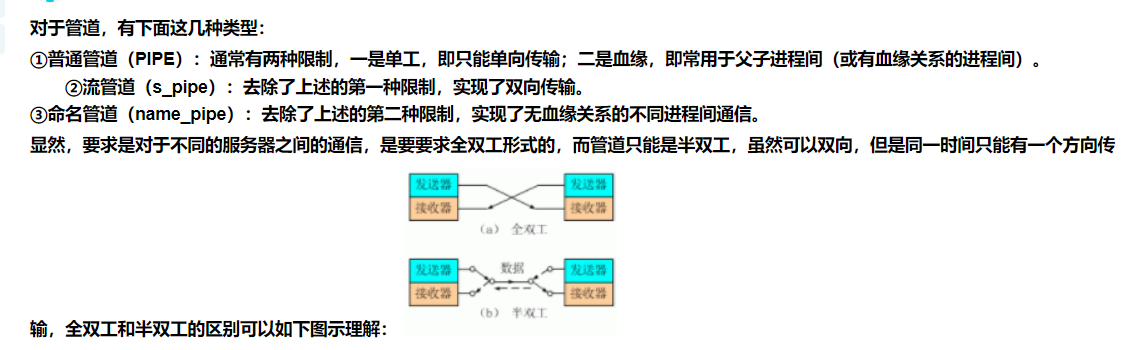


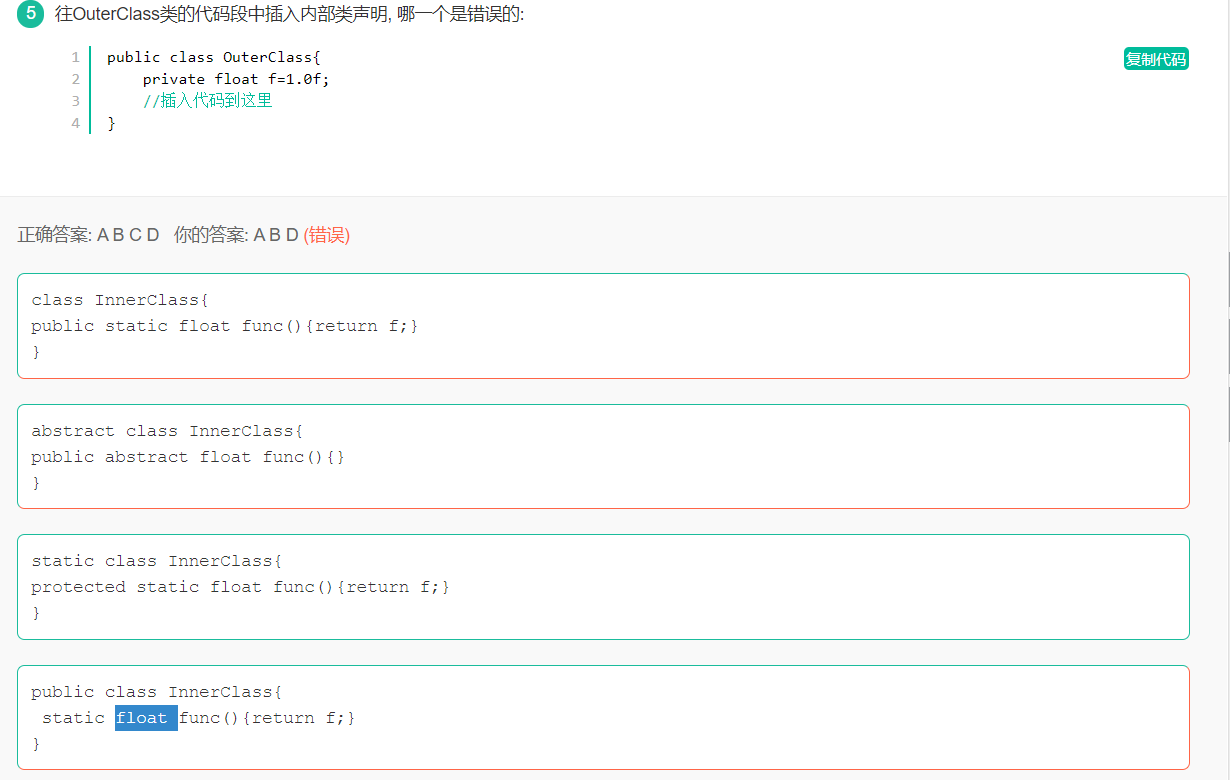


类型转换：









s