1. 解释一下为什么会出现4.0-3.6=0.4000001的现象

二进制小数无法精确表达十进制小数,计算机计算十进制小数时要先转化成二进制小数进行计算,这个过程中出现了误差

1. 请你讲讲一个十进制的数在内存中是怎么存的

以二进制补码形式存储,最高位是符号位,正数的补码就是原码,负数的补码是它的反码加1,再求反码时符号位不变,其他取反,1表示负数,0表示正数.

1. 接口和抽象类区别
2. 接口中所有的方法隐含的都是抽象方法,而抽象类可以包含抽象方法和非抽象方法
3. 类可以实现多个接口,但是只可以继承一个抽象类
4. 类如果不实现接口和抽象类中所有的抽象方法,那么这个类必须声明成抽象的
5. 接口中成员变量默认是public,抽象类中成员变量可以为private,protected,public
6. JDK8后接口可以包含default方法,抽象类中 不可以
7. HTTP 请求中的GET请求和POST区别
8. get在浏览器回退是无害的,post会再次提交请求
9. Get请求会被浏览器主动catch,二post不会,除非主动设置
10. get请求参数在url地址栏,可以看到,大小有限制,小于2kb,post请求在request body里面,没有大小显示.
11. 例出自己常用的jdk的包

Java.long 包装类,线程等都在这个包

Java.match 有BigDecimal精确数字类型

Java.util 并发,集合都在这个包下面

1. Equals和==的区别

Equals比较的是实体的值是否相等,在很多类中可以被覆盖,遵循几个约定

1. 自反性：对于任何非null的引用值x, x.equals(x)必须返回true
2. 对称性：对于任何非null的引用值x和y，当y.equals(x)返回true时，x.equlas(y)必须返回true
3. 传递性：对于任何非null的引用值x、y、z，如果x.equals(y)返回true，并且y.equals(x)也返回true，那么x.equals(z)也必须返回true
4. 一致性：对于任何非null的引用值x和y，只要比较对象中的所有信息没有被修改，多次调用equals一致返回true，或者false

= =比较的是地址值,不能覆盖

int a = 200;  
int b = 200;  
Integer c = 200;  
Integer d = 200;  
Integer e = Integer.*valueOf*(a);  
System.*out*.println(a == b);  
System.*out*.println(b == c);  
System.*out*.println(a == c);  
System.*out*.println(c == d);  
System.*out*.println(a == d);  
System.*out*.println(d == e );  
System.*out*.println(a == e );  
System.*out*.println(a == b && b== c && c.equals(d) );

true

true

true

false

true

false

true

True

1. Hashcode和equals方法的区别

对于覆盖了equals方法的类中同样也要覆盖hashcode,这个是jdk规定的结果

比较两个对象是否相等,hashcode比equals效率高,但是如果不重写hashcode,原本两个对象可以认为是相等的,但是由于hashcode默认返回表示对象地址的整数,必然不相等,所以需要重写hashcode.

Hashcode还有哈希碰撞的问题,两个不相等的对象的hashcode相等,这样还需要equals来比较,比如hashmap在计算key的索引值时用key.hashcode,在确定是都有相同元素时用equals比较

1. 什么是序列化与反序列化

序列化就是加将对象的内容转换为二进制流可以用来持久化和网络传输

反序列化是把二进制流还原为对象的过程

实现java系列化通过实现Serializale即可

1. Object中的常用方法,为什么wait和notify会放在object中

因为Java提供的锁是对象级别的,每个对象都有对象头,用来存储锁

1. 请列举你所知道的object方法并简要说明

Object() 默认构造方法

Clone() 创建并返回对象的一个副本

Equals(object obj) 比较当前对象是否和obj相同

Hashcode() 返回该对象的哈希值

Finallize() 垃圾回收器回收对象时调用的方法

Getclass() 返回对象的运行时类

Wait() 使当前线程等待

Notify() 唤醒在此对象监听器上等待的单个线程

NotifyAll() 唤醒在此对象监听器上等待的所有线程

toString() 返回该对象的字符串表示

1. 创建线程的方式
2. 继承Thread类
3. 实现runnable接口,实现run方法
4. 实现callable接口,实现call方法
5. 通过线程池创建线程
6. Arraylist和linkedlist区别

Arraylist底层是数组,查询快增删慢

Linkedlist底层是双线链表,查询慢,增删快

1. Arraylist的扩容机制

初始容量为10,当容量达到最大并且add时,会发生扩容,

首先会创建一个扩容0.5倍的新数组,如果新数组容量大于需要的容量,则把原数组元素赋给新数组,如果新数组容量还是小于需要的容量,则扩容到需要的最小容量,然后把原数组元素赋给新数组

1. Hashmap1.7和1.8的区别
2. 1.7在put元素时采用头插法,1.8采用了尾插法
3. 1.8引用了红黑树,即链表长度达到7,继续put元素并且hashtable[]数组长度也就是HashMap的capacity(容量)大于等于64,链表就会转化为红黑树,如果没有达到64,则会做一个数组扩容
4. Hashmap的key值可以是任意对象或者数据类型吗

Key可以是null,但是不可以是可变对象,因为key会通过算出key的哈希值来确定他在数组的位置,如果对象可变,那么哈希值也会发生改变,会发生错误.

1. Hashmap的初始容量

Hashmap的初始容量为16,负载因子为0.75,也就是当容量达到12开始扩容

当第一次put元素时开始第一次扩容,容量为16,当put元素的容量达到12时开始第二次扩容,新数组容量为原来的2倍

为什么默认负载因子为0.75----空间利用率和时间效率达到平衡

1. Hashmap,concurrentHashmap,初始化阈值为什么为8,才转为红黑树
   1. 当阈值为8时,链表长度为8的概率很小,,如果在小减少的并不明显
   2. 树结构查找的时间复杂度为O(log(n)),而链表为O(n),当阈值为8,查询速度快,但是树结构占用空间多,是一种时间和空间的平衡

18 .HashMap在并发下会产生什么问题？有什么替代方案?(HashTable, ConcurrentHashMap)。它们两者的实现原理

HashMap并发下产生问题：由于在发生hash冲突，插入链表的时候，多线程会造成环链，再get的时候变成死循环，Map.size()不准确，数据丢失  
 [https://www.iteye.com/blog/hwl-sz-1897468](https://www.iteye.com/blog/hwl-sz-1897468" \t "_blank)

HashTable: 通过synchronized来修饰，效率低，多线程put的时候， 只能有一个线程成功，其他线程都处于阻塞状态

ConcurrentHashMap：  
 1.7 采用锁分段技术提高并发访问率  
 1.8 数据依旧是分段存储，但锁采用了synchronized，内部采用Node数组+链表+红黑树的结构存储，当单个链表存储数量达到红黑树阈值8时（此时链表已有元素7），并且数组长度大于64时，存储结构转换为红黑树来存储，否则只进行数组的扩容

1. Hashmap为什么不采用平衡树而采用红黑树
   1. 红黑树左右高度不能超过2,平衡树左右高度不能超过1
   2. 红黑树在插入,删除时使链表旋转的概率比平衡树低很多,所以效率高
2. 常见异常分为哪两种（Exception,Error），区别是什么，了解受检异常和非受检异常吗

Exception,Error有共同的父类Throwable

Error: 表示程序发生错误，是程序无法处理的，不可恢复的，如OutOfMemoryError

Exception: 表示程序可处理的异常，又分为CheckedException（受检异常）、UncheckedException(非受检异常)，

受检异常发生在编译期，必须要使用try...catch 或者 throws捕获或者抛出异常，否则编译不通过；

非受检异常发生在运行期，具有不确定性，主要由程序的逻辑问题引起的，在程序设计的时候要认真考虑，尽量处理异常