前四周面试题

1、JAVA基本数据类型

整数：byte、short、int、long

浮点：float、double

布尔：boolean

字符：char

2、&和&&的区别

&是位运算符，表示按位与运算，&&是逻辑运算符，表示逻辑与（and）

| 是位运算符，表示按位或运算，|| 是逻辑运算符，表示逻辑或（o）

3、String s = new String("xyz");创建了几个String对象

两个，一个字符对象，一个字符对象引用对象

4、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错

short s1 = 1; s1 = s1 + 1; （s1+1运算结果是int型，需要强制转换类型）

short s1 = 1; s1 += 1;（可以正确编译）

5、Java有没有goto关键字

Java中的保留字，现在没有在java中使用

6、数组有没有length()这个方法?

数组没有length()方法，但是有length属性

7、方法重载是什么？构造方法是否能重载?

1）方法名相同

2）参数列表不同（参数个数或参数类型不同）

构造方法可以重载

8、用最有效率的方法算出2乘以8等於几

2 << 3

9、什么是值传递，什么是引用传递

基本数据类型都是值传递

引用类型都是引用传递（地址传递）

10、switch()中匹配的值可以是什么类型

byte,short,int,char

jdk1.7版本新增String

11、char型变量中能不能存贮一个中文汉字?为什么?

是能够定义一个中文的，因为java中以Unicode编码(0~65535)，一个char占2个字节，可以放一个中文是没问题的

12、float型float f=3.4是否正确?

不正确，3.4是double类型常量,可以使用如下两种方式 ：

```
float f=(float)3.4;
float f = 3.4f;
```

13、排序都有哪几种方法？请写出冒泡和选择排序（代码略）

插入排序、冒泡排序、选择排序、归并排序，分配排序等

14、一个".java"源文件中是否可以包括多个类？有什么限制？

可以。只有一个类名与文件名相同，也只有这个类才能声明为public。

15、静态变量和实例变量的区别？

static 声明的变量是静态的，静态的属于类的，每个对象共享的，就一份。由类名.调用

没有static声明的变量是非静态的即实例变量，由对象.调用

16、是否可以从一个static方法内部发出对非static方法的调用？

不可以直接使用,必须在静态方法内创建对象调用非静态方法

17、跳转语句有几个，分别怎么使用？

continue; 不跳出循环体，结束本轮运行，进入到下一圈运行

break; 跳出当前循环体，也可以结束switch语句段

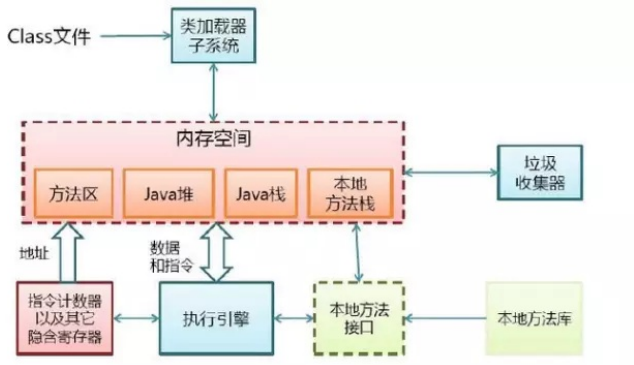
return; 结束方法，直接返回到方法调用的位置

18、JDK和JRE的区别是什么？

jdk 是JAVA开发环境，包括API

jre 是JAVA运行环境，包括JVM

19、JAVA虚拟机的内存分配（理解）



此图看出jvm内存结构

JVM内存结构主要包括两个子系统和两个组件。两个子系统分别是Classloader子系统和Executionengine(执行引擎)子系统；两个组件分别是Runtimedataarea(运行时数据区域)组件和Nativeinterface(本地接口)组件。

Classloader子系统的作用：

根据给定的全限定名类名(如java.lang.Object)来装载class文件的内容到Runtimedataarea中的methodarea(方法区域)。Java程序员可以extendsjava.lang.ClassLoader类来写自己的Classloader。

Executionengine子系统的作用：

执行classes中的指令。任何JVMspecification实现(JDK)的核心都是Executionengine，不同的JDK例如Sun的JDK和IBM的JDK好坏主要就取决于他们各自实现的Executionengine的好坏。

Nativeinterface组件：

与nativelibraries交互，是其它编程语言交互的接口。当调用native方法的时候，就进入了一个全新的并且不再受虚拟机限制的世界，所以也很容易出现JVM无法控制的nativeheapOutOfMemory。

RuntimeDataArea组件：

这就是我们常说的JVM的内存了。它主要分为五个部分——

1、Heap(堆)：一个Java虚拟实例中只存在一个堆空间

2、MethodArea(方法区域)：被装载的class的信息存储在Methodarea的内存中。当虚拟机装载某个类型时，它使用类装载器定位相应的class文件，然后读入这个class文件内容并把它传输到虚拟机中。

3、JavaStack(java的栈)：虚拟机只会直接对Javastack执行两种操作：以帧为单位的压栈或出栈

4、ProgramCounter(程序计数器)：每一个线程都有它自己的PC寄存器，也是该线程启动时创建的。PC寄存器的内容总是指向下一条将被执行指令的饿地址，这里的地址可以是一个本地指针，也可以是在方法区中相对应于该方法起始指令的偏移量。

5、Nativemethodstack(本地方法栈)：保存native方法进入区域的地址

20、构造方法的功能及特点

构造方法为了初始化对象的属性。

1）和类同名

2）没有返回值，void也不能写

3）当不声明构造方法的时候，系统送一个空构造，但是如果声明了构造方法，系统就不送了

4）构造方法是当对象被创建的时候自动被调用的，不能手动调用

21、多态4要素

a) 子类继承父类

b) 子类重写父类方法

c) 父类引用子类对象

d) 父类引用调用子类重写父类的方法，实现多态

22、抽象类和接口的区别

1)抽象类是abstract class ,接口是interface

2)抽象类只能被单继承，接口可以被多实现

3)抽象类中可以普通的属性和方法，接口中只能有抽象方法和静态常量

23、内部类有几种

1）非静态内部类

2）静态内部类

3）方法内部类

4）匿名内部类

24、单例是什么？分别写下饿汉模式和懒汉模式

单例是只能创建唯一对象

饿汉模式

```
class T{
      private  static  T  t  ;
      private  T(){}
      public  static  T   getInstance(){
            if(t == null){
                 t = new T();
             }
            return t;
      }
}
```

懒汉模式

```
class T{
```

```
	private static  T t = new T();
	private T(){}
	public static T  getInstance(){
		return t;
	}
}
```

25、final,finally,finalize的区别?

final　　修饰变量　　常量　（大写）

　　修饰方法　　不能被子类重写

　　修饰类　　不能被继承（终类）

finally　使用try,catch的语句段

finalize　　是垃圾回收机制，System.gc()调用每个对象的finalize方法

26、==和equals比较的区别

	基本数据类型8	引用类型
==	比较值	地址
equals	不能比较基本类型	地址，除了4种类型: 字符串、文件、日期、封装类 因为重写了equals方法，比较的是值

27、什么时候用assert

assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，assertion就是在程序中的一条语句，它对一个boolean表达式进行检查，一个正确程序必须保证这个boolean表达式的值为true；如果该值为false，说明程序已经处于不正确的状态下，系统将给出警告或退出。一般来说，assertion用于保证程序最基本、关键的正确性。

28、给我一个你最常见到的Runtime Exception

IndexOutOfBoundsException，NullPointerException，NumberFormatException,InputMismatchException,ClassNotFoundException等

29、error和exception有什么区别

error 表示恢复不是不可能，但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况

exception 表示一种设计或实现问题。也就是说，它表示程序代码的逻辑错误等。

30、构造器Constructor是否可被override

构造器Constructor不能被继承，因此不能重写Overriding，但可以被重载Overloading

31、try {}里有一个return语句，那么紧跟在这个try后的finally {}里的code会不会被执行，什么时候被执行，在return前还是后

会执行，在return前执行finally

32、两个对象值相同(x.equals(y) == true)，但却可有不同的hashCode，这句话对不对?

不对，equals相同，hashCode一定相同

33、GC是什么? 为什么要有GC

GC是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目

的，Java语言没有提供释放已分配内存的显示操作方法。

34、int 和 Integer 有什么区别

int是基本数据类型，不能使用任何方法。

Integer是int的封装类，可以对基本类型int自动装箱和拆箱。并且Integer是引用类型，有一堆方法。

35、throw和throws区别

throw是要抛出一个异常对象

throws 是方法上声明一个异常类，多个异常类之间用逗号分隔。

throw 一个异常对象，要么使用try-catch给异常解决了。要么声明方法throws 异常类，给异常抛给调用的位置。

36、Collection 和 Collections的区别

Collection是集合类的上级接口，继承与他的接口主要有Set 和List和Queue

Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作

37、Math.round(11.5)等於多少? Math.round(-11.5)等於多少

Math.round(11.5)==12;

Math.round(-11.5)==-11; 四舍五入

38、Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型

方法的重写Override和重载Overload是Java多态性的不同表现。

Overload重载，方法名相同，参数个数或参数类型不同。

Override是必须有继承或实现关系，子类重写（重写：方法名，返回值，参数列表都要相同）父类的方法，实现类实现接口。

39、Set里的元素是不能重复的，那么用什么方法来区分重复与否呢?

使用equals和hashCode两个方法进行比较

40、16、List, Set, Map是否继承自Collection接口

答： List，Set是，Map不是

41、abstract class和interface有什么区别

abstract class是抽象类，一个类中如果有至少一个抽象方法（即没有方法体的方法）则此类必须被声明为抽象类。抽象类是被子类继承extends

interface是接口，比抽象类更抽象，接口中定义的默认是public abstract的方法和public final static的静态常量。接口是被实现类实现implements

42、接口是否可继承接口? 抽象类是否可实现接口? 抽象类是否可继承实体类

接口可以继承接口。抽象类可以实现(implements)接口，抽象类可以继承实体类，但前提是实体类必须有明确的构造函数

43、是否可以继承String类

String类是final类故不可以继承

44、ArrayList、LinkedList、Vector的区别

LinkedList是基于链表的（即每个元素地址不一定连续），所以对它进行索引比较慢，增删效率高

ArrayList和Vector是基于数组的（地址连续），所以对它进行索引很快，增删效率低

ArrayList和Vector有如下不同：

一.同步性:Vector是线程安全的，也就是说是同步的，而ArrayList是线程序不安全的，不是同步的

二.数据增长:当需要增长时,Vector默认增长为原来一培，而ArrayList却是原来的一半

45、HashMap和Hashtable区别

区别：

1、HashMap是线程不安全的，效率高

　　HashTable是线程安全的，效率低，JDK1.5后ConcurrentHashMap替代。

2、HashTable是过时的JDK1.0,HashMap是JDK1.2的

3、HashMap允许键值对为null,HashTable不允许

4、HashMap的迭代器是fail-fast,HashTable使用枚举器进行迭代

46、JAVA中的Collection数据结构

Collection

├List（列表：有序、可重复）

│ ├LinkedList（链表列表）

│ ├ArrayList（数组列表，不安全）

│ └Vector（向量列表，线程安全）

│ 　└Stack（栈）

├Set（集合：无序、不可重复）

│ ├HashSet （基于hash集合，索引快）

│ └TreeSet （排序的集合，调用compareTo方法比较大小）

Map（键值对：无序、key不可重复）

├Hashtable（基于hash，线程安全）

├HashMap（基于hash，不安全）

└TreeMap（对key排序）

47、String、StringBuffer、StringBuilder区别

String不可变长，StringBuffer和StringBuilder可变长

1、效率高低:通常情况

StringBuilder > StringBuffer > String

2、StringBuffer是线程安全的，StringBuilder不安全

3、StringBuilder是StringBuffer的简单实现版，都带缓冲区

48、int 和 Integer 有什么区别

Java 提供两种不同的类型：引用类型和原始类型（或内置类型）。

int是基本数据类型、Integer是引用类型

引用类型和原始类型的行为完全不同，并且它们具有不同的语义。引用类型和原始类型具有不同的特征和用法，它们包括：大小和速度问题，这种类型以哪种类型的数据结构存储，当引用类型和原始类型用作某个类的实例数据时所指定的缺省值。对象引用实例变量的缺省值为 null，而原始类型实例变量的缺省值与它们的类型有关

49、描述一下JVM加载class文件的原理机制

JVM中类的装载是由ClassLoader和它的子类来实现的,Java ClassLoader 是一个重要的Java运行时系统组件。它负责在运行时查找和装入类文件的类。

50、什么是java序列化，如何实现java序列化

序列化是把对象从程序，以规定的序列一次保存到外部介质的过程。

序列化的实现：将需要被序列化的类实现Serializable接口，该接口没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的

51、在JAVA中，如何跳出当前的多重嵌套循环？

用break和标签配合跳出某段代码、或者return 方法。

52、进程和线程的区别是什么

进程是执行着的应用程序，而线程是进程内部的一个执行序列。一个进程可以有多个线程。线程又叫做轻量级进程。

53、创建线程有几种不同的方式？你喜欢哪一种？为什么？

有两种方式可以用来创建线程：

- 继承Thread类
- 实现Runnable接口

实现Runnable接口这种方式更受欢迎，因为这不需要继承Thread类。在应用设计中已经继承了别的对象的情况下，这需要多继承（而Java不支持多继承），只能实现接口。同时，线程池也是非常高效的，很容易实现和使用。

54、概括的解释下线程的几种可用状态

线程在执行过程中，可以处于下面几种状态：

- 就绪(Runnable):线程准备运行，不一定立马就能开始执行。
- 运行中(Running)：进程正在执行线程的代码。
- 等待中(Waiting):线程处于阻塞的状态，等待外部的处理结束。
- 睡眠中(Sleeping)：线程被强制睡眠。
- I/O阻塞(Blocked on I/O)：等待I/O操作完成。
- 同步阻塞(Blocked on Synchronization)：等待获取锁。
- 死亡(Dead)：线程完成了执行。

55、同步方法和同步代码块的区别是什么

在Java语言中，每一个对象有一把锁。线程可以使用synchronized关键字来获取对象上的锁。synchronized关键字可应用在方法级别(粗粒度锁)或者是代码块级别(细粒度锁)。

56、什么是死锁(deadlock)

两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程都陷入了无限的等待中。

57、什么是迭代器(Iterator)？

Iterator接口提供了很多对集合元素进行迭代的方法。每一个集合类都包含了可以返回迭代器实例的迭代方法。迭代器可以在迭代的过程中删除底层集合的元素。

克隆(cloning)或者是序列化(serialization)的语义和含义是跟具体的实现相关的。因此，应该由集合类的具体实现来决定如何被克隆或者是序列化。

58、Iterator和ListIterator的区别是什么？

下面列出了他们的区别：

- Iterator可用来遍历Set和List集合，但是ListIterator只能用来遍历List。
- Iterator对集合只能是前向遍历，ListIterator既可以前向也可以后向。
- ListIterator实现了Iterator接口，并包含其他的功能，比如：增加元素，替换元素，获取前一个和后一个元素的索引，等等。

59、快速失败(fail-fast)和安全失败(fail-safe)的区别是什么？

Iterator的安全失败是基于对底层集合做拷贝，因此，它不受源集合上修改的影响。java.util包下面的所有的集合类都是快速失败的，而java.util.concurrent包下面的所有的类都是安全失败的。快速失败的迭代器会抛出ConcurrentModificationException异常，而安全失败的迭代器永远不会抛出这样的异常。

60、数组和列表(ArrayList)有什么区别？什么时候应该使用Array而不是ArrayList？

- Array可以包含基本类型和对象类型，ArrayList只能包含对象类型。
- Array大小是固定的，ArrayList的大小是动态变化的。
- ArrayList提供了更多的方法和特性，比如：addAll()，removeAll()，iterator()等等。
- 对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。

61、如果对象的引用被置为null，垃圾收集器是否会立即释放对象占用的内存？

不会，在下一个垃圾回收周期中，这个对象将是可被回收的。

62、Java中垃圾回收有什么目的？什么时候进行垃圾回收？

垃圾回收的目的是识别并且丢弃应用不再使用的对象来释放和重用资源。

63、在Java中，对象什么时候可以被垃圾回收？

当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

64、MVC是啥？MVC的好处?

Model-View-Controller(模型视图控制器），用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

好处：

1：耦合性低

2：重用性高

3：部署快

4：可维护性高

5：有利软件工程化管理

65、在java中wait和sleep方法的不同

1.这两个方法来自不同的类分别是Thread和Object

2.最主要是sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可以使用同步控制块或者方法。

3.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而sleep可以在任何地方使用

66、用Java编程一个会导致死锁的程序

```
public class DeadLock {
    public static String obj1 = "obj1";
    public static String obj2 = "obj2";
    public static void main(String[] args){
        Thread a = new Thread(new Lock1());
        Thread b = new Thread(new Lock2());
        a.start();
        b.start();
    }
}
class Lock1 implements Runnable{
    @Override
    public void run(){
        try{
            System.out.println(Lock"1 running");
```

```
            while(true){
                synchronized(DeadLock.obj1){
                    System.out.println("Lock1 lock obj1");
                    Thread.sleep(3000);//获取obj1后先等一会儿，让Lock2有足够的时间锁住obj2
                    synchronized(DeadLock.obj2){
                        System.out.println("Lock1 lock obj2");
                    }
                }
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
class Lock2 implements Runnable{
    @Override
    public void run(){
        try{
            System.out.println("Lock2 running");
            while(true){
                synchronized(DeadLock.obj2){
                    System.out.println("Lock2 lock obj2");
                    Thread.sleep(3000);
                    synchronized(DeadLock.obj1){
                        System.out.println("Lock2 lock obj1");
                    }
                }
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

67、为什么我们调用start()方法时会执行run()方法，为什么我们不能直接调用run()方法？

这个问题的回答应该是这样的，当你调用start()方法时你将创建新的线程，并且执行在run()方法里的代码。但是如果你直接调用run()方法，它不会创建新的线程也不会执行调用线程的代码。