Java 基础知识 30 问

1. 面向对象和面向过程的区别

2. Java 语言有哪些特点？

简单易学（2）平台无关（Java 虚拟机实现平台无关性）（3）面向对象（封装、继承、多态）（4）可靠性（5）安全性（6）支持多线程（7）编译与解释共存

3. 什么是 JDK？什么是 JRE？什么是 JVM？三者之间的联系与区别

JDK 是给开发者提供的工具箱，包括完整的 JRE、Java 运行环境还包含了其他供给开发者提供的工具包（2）JRE：普通用户只需要安装 JRE，程序开发者需要安装 JDK 来编译调试程序（3）JVM：运行程序时，JVM 负责将字节码转换成机器码，JVM 提供内存管理、垃圾回收和安全机制，这种独立于硬件和操作系统，是 java 程序可以一次编写多次执行的原因。JDK 用于开发，JRE 用于运行程序；JDK 和 JRE 都包含 JVM；JVM 是 Java 编程语言的核心并具有平台独立性

4. 什么是字节码？采用字节码的最大好处是什么？

Java 中引入了虚拟机的概念，就是在机器和编译程序之间加入了一层抽象的虚拟的机器，这台虚拟的机器在任何平台上都提供给编译程序一个共同的接口。Java 中，由虚拟机理解的代码称为字节码*（.class 文件），只面向虚拟机。Java 源代码经过编译器编译后编程字节码，字节码由虚拟机解释执行，虚拟机将每一条要执行的字节码送给解释器，解释器将其翻译成特定机器上的机器码，然后在特定的机器上执行，这也就解释了 Java 的编译与解释并存的特点

Java 源代码---->编译器---->jvm 可执行的 Java 字节码(即虚拟指令)---->jvm---->jvm 中解释器----->机器可执行的二进制机器码---->程序运行。

5. Java 和 C++的区别

6. 什么是 Java 程序的主类？应用程序和小程序的主类有何不同？

7. Java 应用程序与小程序之间有那些差别？

8. 字符型常量和字符串常量的区别

9. 构造器 Constructor 是否可被 override（不可以，但可以 overload）

10. 重载和重写的区别

（1） 重载：发生在同一个类中，方法名相同，参数类型，个数、顺序不同，方法返回值和访问修饰符可以不同，发生在编译时

（2） 重写：发生在父子类中，方法名、参数列表必须相同，返回值范围小于父类，抛出异常类型小于父类，访问修饰符大于等于父类；如果父类方法访问修饰是 private 则不能重写该方法

11. Java 面向对象编程三大特性:封装、继承、多态

12. String 和 StringBuffer、StringBuilder 的区别是什么？String 为什么是不可变的？

可变性：String 类使用字符数组保存字符串，private final char[] value，String 对象是不可变的；Stringbuffer 和 StringBuilder 对象是可变的

线程安全性：Stirng、StringBuffer 是线程安全的，StringBuilder 不是线程安全的

性能：每次对 String 类型改变的时候都会生成一个新的 String 对象，然后将指针指向新的 String 对象；StringBuffer 每次对会对 StringBuffer 对象本身进行操作，而不是生成新的对象；StringBuilder 相比 StringBuffer 仅能获得 10-15%左右的性能提升，但却要冒着多线程不安全的风险

对于三者使用的总结：如果要操作少量的数据用 = String 单线程操作字符串缓冲区 下操作大量数据 = StringBuilder 多线程操作字符串缓冲区 下操作大量数据 =StringBuffer

13. 自动装箱与拆箱

装箱：将基本类型用它们对应的引用类型包装起来；

拆箱：将包装类型转换为基本数据类型；

14. 在一个静态方法内调用一个非静态成员为什么是非法的？

15. 在 Java 中定义一个不做事且没有参数的构造方法的作用

16. import java 和 javax 有什么区别实际上 java 和 javax 没有区别。这都是一个名字

17. 接口和抽象类的区别是什么？

1.接口的方法默认是 public，所有方法在接口中不能有实现，抽象类可以有非抽象的方法

2.接口中的实例变量默认是 final 类型的，而抽象类中则不一定

3.一个类可以实现多个接口，但最多只能实现一个抽象类

4.一个类实现接口的话要实现接口的所有方法，而抽象类不一定

5.接口不能用 new 实例化，但可以声明，但是必须引用一个实现该接口的对象 从设计层面来说，抽象是对类的抽象，是一种模板设计，接口是行为的抽象，是一种行为的规范。

18. 成员变量与局部变量的区别有那些？

1.从语法形式上，看成员变量是属于类的，而局部变量是在方法中定义的变量或是方法的参数；成员变量可以被 public,private,static 等修饰符所修饰，而局部变量不能被访问控制修饰符及 static 所修饰；但是，成员变量和局部变量都能被 final 所修饰；

2.从变量在内存中的存储方式来看，成员变量是对象的一部分，而对象存在于堆内存，局部变量存在于栈内存

3.从变量在内存中的生存时间上看，成员变量是对象的一部分，它随着对象的创建而存在，而局部变量随着方法的调用而自动消失。

4.成员变量如果没有被赋初值，则会自动以类型的默认值而赋值（一种情况例外被 final 修饰但没有被 static 修饰的成员变量必须显示地赋值）；而局部变量则不会自动赋值。

19. 创建一个对象用什么运算符？对象实体与对象引用有何不同？

new 运算符，new 创建对象实例（对象实例在堆内存中），对象引用指向对象实例（对象引用存放在栈内存中）。一个对象引用可以指向 0 个或 1 个对象（一根绳子可以不系气球，也可以系一个气球）;一个对象可以有 n 个引用指向它（可以用 n 条绳子系住一个气球）。

20. 什么是方法的返回值？返回值在类的方法里的作用是什么？

21. 一个类的构造方法的作用是什么？若一个类没有声明构造方法，改程序能正确执行吗？为什么？

主要作用是完成对类对象的初始化工作。可以执行。因为一个类即使没有声明构造方法也会有默认的不带参数的构造方法。

22. 构造方法有哪些特性？

名字与类名相同；没有返回值，但不能用 void 声明构造函数；生成类的对象时自动执行，无需调用。

23. 静态方法和实例方法有何不同？

在外部调用静态方法时，可以使用"类名.方法名"的方式，也可以使用"对象名.方法名"的方式。而实例方法只有后面这种方式。也就是说，调用静态方法可以无需创建对象。

静态方法在访问本类的成员时，只允许访问静态成员（即静态成员变量和静态方法），而不允许访问实例成员变量和实例方法；实例方法则无此限制.

24. 对象的相等与指向他们的引用相等，两者有什么不同？

对象的相等 比的是内存中存放的内容是否相等而引用相等 比较的是他们指向的内存地址是否相等。

26. == 与 equals(重要)

==：它的作用是判断两个对象的地址是不是相等。即，判断两个对象是不是同一个对象。(基本数据类型==比较的是值，引用数据类型==比较的是内存地址)

equals()：它的作用也是判断两个对象是否相等。但它一般有两种使用情况：

- 情况 1：类没有覆盖 equals() 方法。则通过 equals() 比较该类的两个对象时，等价于通过“==”比较这两个对象。
- 情况 2：类覆盖了 equals() 方法。一般，我们都覆盖 equals() 方法来两个对象的内容相等；若它们的内容相等，则返回 true (即，认为这两个对象相等)。

27. hashCode 与 equals（重要）

面试官可能会问你：“你重写过 hashcode 和 equals 么，为什么重写 equals 时必须重写 hashCode 方法？”

hashCode（）介绍

hashCode() 的作用是获取哈希码，也称为散列码；它实际上是返回一个 int 整数。这个哈希码的作用是确定该对象在哈希表中的索引位置。hashCode() 定义在 JDK 的 Object.java 中，这就意味着 Java 中的任何类都包含有 hashCode() 函数。

散列表存储的是键值对(key-value)，它的特点是：能根据“键”快速的检索出对应的“值”。这其中就利用到了散列码！（可以快速找到所需要的对象）

为什么要有 hashCode

我们以“HashSet 如何检查重复”为例子来说明为什么要有 hashCode：

当你把对象加入 HashSet 时，HashSet 会先计算对象的 hashcode 值来判断对象加入的位置，同时也会与其他已经加入的对象的 hashcode 值作比较，如果没有相符的 hashcode，HashSet 会假设对象没有重复出现。但是如果发现有相同 hashcode 值的对象，这时会调用 equals（）方法来检查 hashcode 相等的对象是否真的相同。如果两者相同，HashSet 就不会让其加入操作成功。如果不同的话，就会重新散列到其他位置。（摘自我的 Java 启蒙书《Head fist java》第二版）。这样我们就大大减少了 equals 的次数，相应就大大提高了执行速度。

hashCode（）与 equals（）的相关规定

1. 如果两个对象相等，则 hashcode 一定也是相同的
2. 两个对象相等,对两个对象分别调用 equals 方法都返回 true
3. 两个对象有相同的 hashcode 值，它们也不一定是相等的
4. **因此，equals 方法被覆盖过，则 hashCode 方法也必须被覆盖**
5. hashCode() 的默认行为是对堆上的对象产生独特值。如果没有重写 hashCode()，则该 class 的两个对象无论如何都不会相等（即使这两个对象指向相同的数据）

28. Java 中的值传递和引用传递

值传递是指对象被值传递，意味着传递了对象的一个副本，即使副本被改变，也不会影响源对象。（因为值传递的时候，实际上是将实参的值复制一份给形参。）

引用传递是指对象被引用传递，意味着传递的并不是实际的对象，而是对象的引用。因此，外部对引用对象的改变会反映到所有的对象上。（因为引用传递的时候，实际上是将实参的地址值复制一份给形参。）

29. 简述线程，程序、进程的基本概念。以及他们之间关系是什么？

线程与进程相似，但线程是一个比进程更小的执行单位。一个进程在其执行的过程中可以产生多个线程。与进程不同的是同类的多个线程共享同一块内存空间和一组系统资源，所以系统在产生一个线程，或是在各个线程之间作切换工作时，负担要比进程小得多，也正因为如此，线程也被称为轻量级进程。

程序是含有指令和数据的文件，被存储在磁盘或其他的数据存储设备中，也就是说程序是静态的代码。

进程是程序的一次执行过程，是系统运行程序的基本单位，因此进程是动态的。系统运行一个程序即是一个进程从创建，运行到消亡的过程。简单来说，一个进程就是一个执行中的程序，它在计算机中一个指令接着一个指令地执行着，同时，每个进程还占有某些系统资源如 CPU 时间，内存空间，文件，文件，输入输出设备的使用权等等。换句话说，当程序在执行时，将会被操作系统载入内存中。 线程是进程划分成的更小的运行单位。线程和进程最大的不同在于基本上各进程是独立的，而各线程则不一定，因为同一进程中的线程极有可能会相互影响。从另一角度来说，进程属于操作系统的范畴，主要是同一段时间内，可以同时执行一个以上的程序，而线程则是在同一程序内几乎同时执行一个以上的程序段。

30. 线程有哪些基本状态？这些状态是如何定义的?

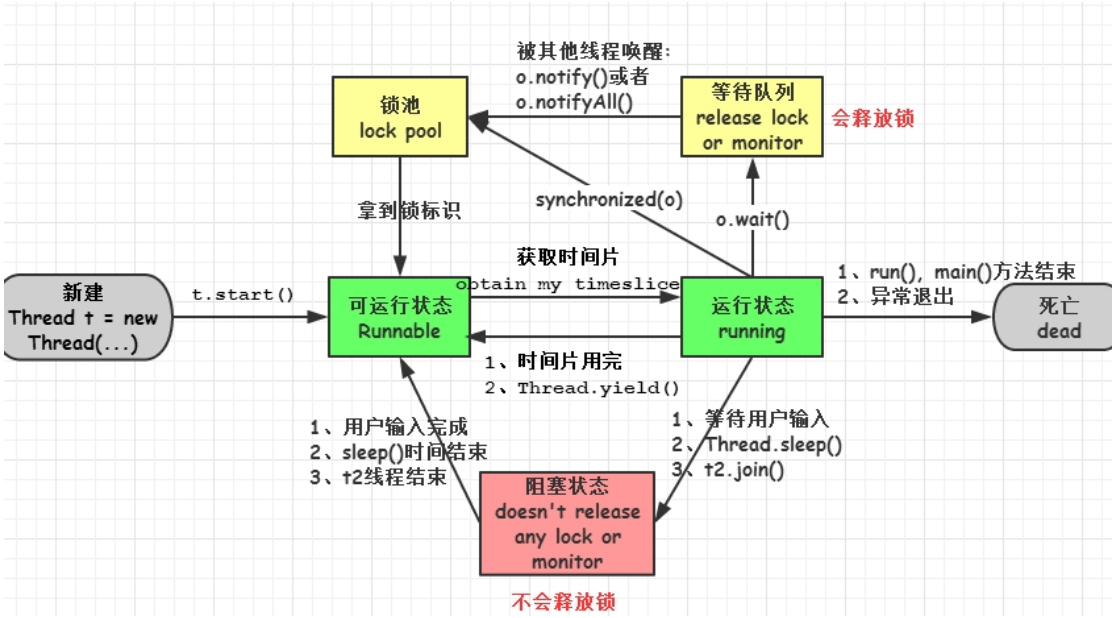
1.新建(new)：新创建了一个线程对象。

2.可运行(runnable)：线程对象创建后，其他线程(比如 main 线程）调用了该对象的 start()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获 取 cpu 的使用权。

3.运行(running)：可运行状态(runnable)的线程获得了 cpu 时间片（ timeslice），执行程序代码。

4.阻塞(block)：阻塞状态是指线程因为某种原因放弃了 cpu 使用权，也即让出了 cpu timeslice，暂时停止运行。直到线程进入可运行(runnable)状态，才有 机会再次获得 cpu timeslice 转到运行(running)状态。阻塞的情况分三种： (一). 等待阻塞：运行(running)的线程执行 o.wait()方法，JVM 会把该线程放 入等待队列(waitting queue)中。 (二). 同步阻塞：运行(running)的线程在获取对象的同步锁时，若该同步锁 被别的线程占用，则 JVM 会把该线程放入锁池(lock pool)中。 (三). 其他阻塞: 运行(running)的线程执行 Thread.sleep(long ms)或 t.join()方法，或者发出了 I/O 请求时，JVM 会把该线程置为阻塞状态。当 sleep()状态超时 join()等待线程终止或者超时、或者 I/O 处理完毕时，线程重新转入可运行(runnable)状态。

5.死亡(dead)：线程 run()、main()方法执行结束，或者因异常退出了 run()方法，则该线程结束生命周期。死亡的线程不可再次复生。



[Java 基础学习书籍推荐]