JAVA常见面试题及解答

Java的垃圾回收总结  浅谈Java中的内部类  
       JAVA常见面试题及解答（精华）  2009-05-20 14:47:25|  分类： 编程语言 |  标签： |字号大  
       1）transient和volatile是java关键字吗？（瞬联）  
       如果用transient声明一个实例变量，当对象存储时，它的值不需要维持。例如：  
class T  
{   
transient int a;  //不需要维持  
int b;  //需要维持  
}   
       这里，如果T类的一个对象写入一个持久的存储区域，a的内容不被保存，但b的将被保存。  
       volatile修饰符告诉编译器被volatile修饰的变量可以被程序的其他部分改变。在多线程程序中，有时两个或更多的线程共享一个相同的实例变 量。考虑效率问题，每个线程可以自己保存该共享变量的私有拷贝。实际的变量副本在不同的时候更新，如当进入synchronized方法时。 用strictfp修饰类或方法，可以确保浮点运算（以及所有切断）正如早期的Java版本那样准确。切断只影响某些操作的指数。当一个类被 strictfp修饰，所有的方法自动被strictfp修饰。  
       strictfp的意思是FP-strict，也就是说精确浮点的意思。在Java虚拟机进行浮点运算时，如果没有指定strictfp关键字 时，Java的编译器以及运行环境在对浮点运算的表达式是采取一种近似于我行我素的行为来完成这些操作，以致于得到的结果往往无法令你满意。而一旦使用了 strictfp来声明一个类、接口或者方法时，那么所声明的范围内Java的编译器以及运行环境会完全依照浮点规范IEEE-754来执行。因此如果你 想让你的浮点运算更加精确，而且不会因为不同的硬件平台所执行的结果不一致的话，那就请用关键字strictfp。  
你可以将一个类、接口以及方法声明为strictfp，但是不允许对接口中的方法以及构造函数声明strictfp关键字，例如下面的代码：  
strictfp interface A {}  
public strictfp class FpDemo1 {    
      strictfp void f() {}    
}   
2. 错误的使用方法  
interface A {    
    strictfp void f();    
}    
public class FpDemo2 {    
    strictfp FpDemo2() {}    
}  
一旦使用了关键字strictfp来声明某个类、接口或者方法时，那么在这个关键字所声明的范围内所有浮点运算都是精确的，符合IEEE-754规范  
的。例如一个类被声明为strictfp，那么该类中所有的方法都是strictfp的。  
2）抽象类和接口有什么区别？（瞬联）  
1.abstract class 在 Java 语言中表示的是一种继承关系，一个类只能使用一次继承关系。但是，一个类却可以实现多个interface。  
2.在abstract class 中可以有自己的数据成员，也可以有非abstarct的成员方法，而在interface中，只能够有静态的不能被修改的数据成员（也就是必须是 static final的，不过在 interface中一般不定义数据成员），所有的成员方法都是abstract的。  
3.abstract class和interface所反映出的设计理念不同。其实abstract class表示的是"is-a"关系，interface表示的是"like-a"关系。  
4.实现抽象类和接口的类必须实现其中的所有方法。抽象类中可以有非抽象方法。接口中则不能有实现方法。  
5.接口中定义的变量默认是public static final 型，且必须给其初值，所以实现类中不能重新定义，也不能改变其值。  
6.抽象类中的变量默认是 friendly 型，其值可以在子类中重新定义，也可以重新赋值。  
7.接口中的方法默认都是 public,abstract 类型的。  
3）能说一下java的反射(reflection)机制吗？（瞬联）  
开放性和原因连接(causally-connected)是反射系统的两大基本要素  
4）在java中怎样实现多线程？（瞬联）  
extends Thread  
implement Runnable  
方法一：继承 Thread 类，覆盖方法 run()，我们在创建的 Thread 类的子类中重写 run() ,加入线程所要执行的代码即可。下面是一个例子：  
public class MyThread extends Thread  
{  
int count= 1, number;  
public MyThread(int num)  
{  
umber = num;  
System.out.println  
("创建线程 " + number);  
}  
public void run() {  
while(true) {  
System.out.println  
("线程 " + number + ":计数 " + count);  
if(++count== 6) return;  
}  
}  
public static void main(String args[])  
{  
for(int i = 0;i 〈 5; i++) new MyThread(i+1).start();  
}  
}  
这种方法简单明了，符合大家的习惯，但是，它也有一个很大的缺点，那就是如果我们的类已经从一个类继承（如小程序必须继承自 Applet 类），则无法再继承 Thread 类，这时如果我们又不想建立一个新的类，应该怎么办呢？  
我们不妨来探索一种新的方法：我们不创建Thread类的子类，而是直接使用它，那么我们只能将我们的方法作为参数传递给 Thread 类的实例，有点类似回调函数。但是 Java 没有指针，我们只能传递一个包含这个方法的类的实例。  
那么如何限制这个类必须包含这一方法呢？当然是使用接口！（虽然抽象类也可满足，但是需要继承，而我们之所以要采用这种新方法，不就是为了避免继承带来的限制吗？）  
Java 提供了接口 java.lang.Runnable 来支持这种方法。  
方法二：实现 Runnable 接口  
Runnable接口只有一个方法run()，我们声明自己的类实现Runnable接口并提供这一方法，将我们的线程代码写入其中，就完成了这一部 分的任务。但是Runnable接口并没有任何对线程的支持，我们还必须创建Thread类的实例，这一点通过Thread类的构造函数 public Thread(Runnable target);来实现。下面是一个例子：  
public class MyThread implements Runnable  
{  
int count= 1, number;  
public MyThread(int num)  
{  
umber = num;  
System.out.println("创建线程 " + number);  
}  
public void run()  
{  
while(true)  
{  
System.out.println  
("线程 " + number + ":计数 " + count);  
if(++count== 6) return;  
}  
}  
public static void main(String args[])  
{  
for(int i = 0; i 〈 5;i++) new Thread(new MyThread(i+1)).start();  
}  
}  
严格地说，创建Thread子类的实例也是可行的，但是必须注意的是，该子类必须没有覆盖 Thread 类的 run 方法，否则该线程执行的将是子类的 run 方法，而不是我们用以实现Runnable 接口的类的 run 方法，对此大家不妨试验一下。  
使用 Runnable 接口来实现多线程使得我们能够在一个类中包容所有的代码，有利于封装，它的缺点在于，我们只能使用一套代码，若想创建多个线程并使各个线程执行不同的代 码，则仍必须额外创建类，如果这样的话，在大多数情况下也许还不如直接用多个类分别继承 Thread 来得紧凑。  
综上所述，两种方法各有千秋，大家可以灵活运用。  
下面让我们一起来研究一下多线程使用中的一些问题。  
三、线程的四种状态  
1. 新状态：线程已被创建但尚未执行（start() 尚未被调用）。  
2. 可执行状态：线程可以执行，虽然不一定正在执行。CPU 时间随时可能被分配给该线程，从而使得它执行。  
3. 死亡状态：正常情况下 run() 返回使得线程死亡。调用 stop()或 destroy() 亦有同样效果，但是不被推荐，前者会产生异常，后者是强制终止，不会释放锁。  
4. 阻塞状态：线程不会被分配 CPU 时间，无法执行。  
四、线程的优先级  
线程的优先级代表该线程的重要程度，当有多个线程同时处于可执行状态并等待获得 CPU 时间时，线程调度系统根据各个线程的优先级来决定给谁分配 CPU 时间，优先级高的线程有更大的机会获得 CPU 时间，优先级低的线程也不是没有机会，只是机会要小一些罢了。  
你可以调用 Thread 类的方法 getPriority() 和 setPriority()来存取线程的优先级，线程的优先级界于1(MIN\_PRIORITY)和10(MAX\_PRIORITY)之间，缺省是5(NORM\_PRIORITY)。