# TODO

线程安全类

在集合框架中，有些类是线程安全的，这些都是jdk1.1中的出现的。在jdk1.2之后，就出现许许多多非线程安全的类。 下面是这些线程安全的同步的类：

**vector：就比arraylist多了个同步化机制（线程安全），因为效率较低，现在已经不太建议使用。在web应用中，特别是前台页面，往往效率（页面响应速度）是优先考虑的。**

**statck：堆栈类，先进后出**

**hashtable：就比hashmap多了个线程安全**

**enumeration：枚举，相当于迭代器**

除了这些之外，其他的都是非线程安全的类和接口。

线程安全的类其方法是同步的，每次只能一个访问。是重量级对象，效率较低。

其他：

1. hashtable跟hashmap的区别

hashtable是线程安全的,即hashtable的方法都提供了同步机制；hashmap不是线程安全的,即不提供同步机制 ；hashtable不允许插入空值,hashmap允许!

2. 多线程并发修改一 个 集合 怎么办

用老的Vector/Hashtable类

StringBuffer是线程安全，而StringBuilder是线程不安全的。对于安全与不安全没有深入的理解情况下，易造成这样的错觉，如果对于StringBuffer的操作均是线程安全的，然而，JAVA给你的保证的线程安全，是说它的方法是执行是排它的，而不是对这个对象本身的多次调用情况下，还是安全的。看看下边的例子，在StringBufferTest中有一个数据成员contents它是用来扩展的，它的每一次append是线程安全的，但众多次append的组合并不是线程安全的，这个输出结果不是太可控的，但如果对于log和getContest方法加关键字synchronized，那么结果就会变得非常条理，如果换成StringBuider甚至是append到一半，它也会让位于其它在此基础上操作的线程：

public class StringBufferTest {

   private StringBuffer contents = new StringBuffer();

   public void log(String message){

      contents.append(System.currentTimeMillis());

      contents.append("; ");

      contents.append(Thread.currentThread().getName());

      for(int i=0;i<10000;i++){

        contents.append(i);

          contents.append(message);    //append本身是线程安全的，修改contents时，其它线程无法访问。

          contents.append("\n");

      }

      contents.append("\n\n");

   }

   public void getContents(){

      System.out.println(contents);

   }

}

class RunThread extends Thread{

   String message;

   StringBufferTest buffer;

   public RunThread(StringBufferTest buffer, String message){

      this.buffer = buffer;

      this.message = message;

   }

   public void run(){

      while(true){

         buffer.log(message);

         buffer.getContents();

      }

   }

   public static void main(String[] args) {

      StringBufferTest ss = new StringBufferTest();

      new RunThread(ss, "you").start();

      new RunThread(ss, "me").start();

      new RunThread(ss, "she").start();

   }

}

        StringBuilder和StringBuffer的方法是一模一样，就是一个多线程和一个单线程的问题。线程调用同一StringBuffer 的append方法，这跟他是不是线程安全没有关系的，除非你的结果是append的一系列字符串变乱了，那才能说明他是线程不安全的。线程安全是指任何时刻都只有一个线程访问临界资源。线程安全 并不是说他的一系列操作是同步的 只是对于他执行某个方法的时候不允许别的线程去改变。针对一个类来说是不是线程安全就要看，多个线程在同时在运行，这些线程可能会同时执行某个方法。但是每次运行结果和单线程执行的结果一样，那么就可以说是线程安全的。因为log方法没有上锁，每个现在在append锁释放后，都可能得到cpu的执行片段。

但不要对多线程安全存在误解：

   public String toString(){

      StringBuffer buffer = new StringBuffer();

      buffer.append('<');

      buffer.append(this.name);

      buffer.append('>');

      return buffer.toString();

   }

       这个代码是完全线程安全的，在方法内部定义的变量，在每个线程线程进入的时候都会创建这个局部变量！不涉及线程安全问题。通常涉及系统安全的变量一般都是成员变量! stringBuffer本身的内部实现是现场安全的！线程安全那是类本身提供的功能是安全的。即你提供插入一个字符串，那么这个字符串插入是安全的，但是要插入两个字符串，两个的顺序你来定，这之间如果有别的插入出错就不管类的事情了，是你自己代码的问题。

# REF