**Java中各种集合（字符串类）的线程安全性**

**一、概念：**

* 线程安全：就是当多线程访问时，采用了加锁的机制；即当一个线程访问该类的某个数据时，会对这个数据进行保护，其他线程不能对其访问，直到该线程读取完之后，其他线程才可以使用。防止出现数据不一致或者数据被污染的情况。
* 线程不安全：就是不提供数据访问时的数据保护，多个线程能够同时操作某个数据，从而出现数据不一致或者数据污染的情况。
* 对于线程不安全的问题，一般会使用synchronized关键字加锁同步控制。
* 线程安全 工作原理： jvm中有一个main memory对象，每一个线程也有自己的working memory，一个线程对于一个变量variable进行操作的时候， 都需要在自己的working memory里创建一个copy,操作完之后再写入main memory。   
  当多个线程操作同一个变量variable，就可能出现不可预知的结果。   
  而用synchronized的关键是建立一个监控monitor，这个monitor可以是要修改的变量，也可以是其他自己认为合适的对象(方法)，然后通过给这个monitor加锁来实现线程安全，每个线程在获得这个锁之后，要执行完加载load到working memory 到 use && 指派assign 到 存储store 再到 main memory的过程。才会释放它得到的锁。这样就实现了所谓的线程安全。

**二、线程安全(Thread-safe)的集合对象：**

* Vector
* HashTable
* StringBuffer

**三、非线程安全的集合对象：**

* ArrayList ：
* LinkedList：
* HashMap：
* HashSet：
* TreeMap：
* TreeSet：
* StringBulider：

**四、相关集合对象比较：**

* **Vector、ArrayList、LinkedList：**  
  **1、Vector：**   
  Vector与ArrayList一样，也是通过数组实现的，不同的是它支持线程的同步，即某一时刻只有一个线程能够写Vector，避免多线程同时写而引起的不一致性，但实现同步需要很高的花费，因此，访问它比访问ArrayList慢。   
  **2、ArrayList：**  
  a. 当操作是在一列数据的后面添加数据而不是在前面或者中间，并需要随机地访问其中的元素时，使用ArrayList性能比较好。   
  b. ArrayList是最常用的List实现类，内部是通过数组实现的，它允许对元素进行快速随机访问。数组的缺点是每个元素之间不能有间隔，当数组大小不满足时需要增加存储能力，就要讲已经有数组的数据复制到新的存储空间中。当从ArrayList的中间位置插入或者删除元素时，需要对数组进行复制、移动、代价比较高。因此，它适合随机查找和遍历，不适合插入和删除。   
  **3、LinkedList：**  
  a. 当对一列数据的前面或者中间执行添加或者删除操作时，并且按照顺序访问其中的元素时，要使用LinkedList。   
  b. LinkedList是用链表结构存储数据的，很适合数据的动态插入和删除，随机访问和遍历速度比较慢。另外，他还提供了List接口中没有定义的方法，专门用于操作表头和表尾元素，可以当作堆栈、队列和双向队列使用。

　　  Vector和ArrayList在使用上非常相似，都可以用来表示一组数量可变的对象应用的集合，并且可以随机的访问其中的元素。

* **HashTable、HashMap、HashSet：**  
  HashTable和HashMap采用的存储机制是一样的，不同的是：   
  **1、HashMap：**  
  a. 采用数组方式存储key-value构成的Entry对象，无容量限制；   
  b. 基于key hash查找Entry对象存放到数组的位置，对于hash冲突采用链表的方式去解决；   
  c. 在插入元素时，可能会扩大数组的容量，在扩大容量时须要重新计算hash，并复制对象到新的数组中；   
  d. 是非线程安全的；   
  e. 遍历使用的是Iterator迭代器；

**2、HashTable：**  
a. 是线程安全的；   
b. 无论是key还是value都不允许有null值的存在；在HashTable中调用Put方法时，如果key为null，直接抛出NullPointerException异常；   
c. 遍历使用的是Enumeration列举；

**3、HashSet：**  
a. 基于HashMap实现，无容量限制；   
b. 是非线程安全的；   
c. 不保证数据的有序；

* **TreeSet、TreeMap：**  
  TreeSet和TreeMap都是完全基于Map来实现的，并且都不支持get(index)来获取指定位置的元素，需要遍历来获取。另外，TreeSet还提供了一些排序方面的支持，例如传入Comparator实现、descendingSet以及descendingIterator等。   
  **1、TreeSet：**  
  a. 基于TreeMap实现的，支持排序；   
  b. 是非线程安全的；

**2、TreeMap：**  
a. 典型的基于红黑树的Map实现，因此它要求一定要有key比较的方法，要么传入Comparator比较器实现，要么key对象实现Comparator接口；   
b. 是非线程安全的；

* **StringBuffer和StringBulider：**  
  StringBuilder与StringBuffer都继承自AbstractStringBuilder类，在AbstractStringBuilder中也是使用字符数组保存字符串。

　　  1、在执行速度方面的比较：StringBuilder > StringBuffer ；   
　　  2、他们都是字符串变量，是可改变的对象，每当我们用它们对字符串做操作时，实际上是在一个对象上操作的，不像String一样创建一些对象进行操作，所以速度快；   
　  　3、 StringBuilder：线程非安全的；   
　　  4、StringBuffer：线程安全的；

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