# Java基础

1.String 是否线程安全

线程安全

2.String x="123" 创建了几个对象

若在常量池中没有则创建两个，一个在常量池，一个在堆中，有则直接从常量池中取，不创建

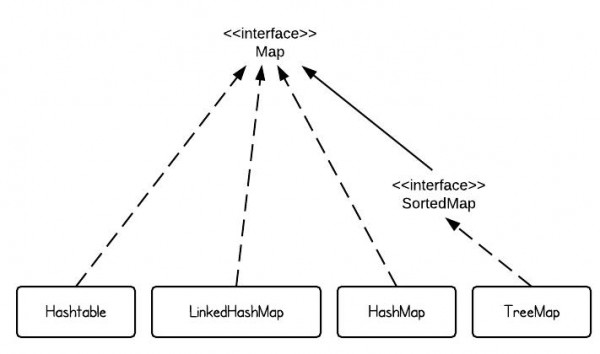
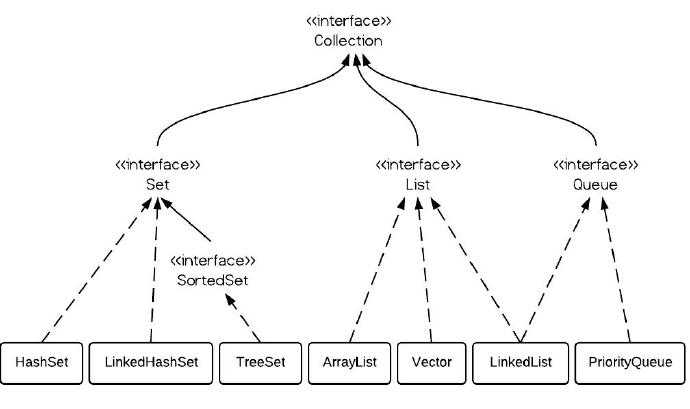
String m = new String("asad");

首先会先在堆中创建一个对象，其次若在常量池中已经存在则不再创建，若不存在，则在常量池中会再创建一个对象。

3. 数组有没有length()这个方法? String有没有length()这个方法？

数组没有length()这个方法，有length的属性。String有有length()这个方法

4.list,set 有什么区别，分别继承哪些接口



List底层是数组或者链表。Hashset里面引用的hashmap,虽然set的接口是collection,但是实现是用的hashmap.

5.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? 是用==还是equals()? 它们有何区别?

**Set底层是引用hashmap,判断逻辑和hashmap一样，hashmap添加一个元素的逻辑是，先利用key取hash找到对应的在数组中的位置，若该位置元素为null则new一个元素存放进去。**

**若已经存在元素，则判断key是否与该元素的key相等，若不相等则继续往下遍历，若最后都没有找到key相等的元素，则直接在链表尾部新插入一个节点（jdk8会判断链表元素个数大于8的话会树化），若找到了相等的key的节点，则将该节点的value直接替换为目标value。**

**关于判断key相等的逻辑如下：**

**if** (p.**hash** == hash &&  
 ((k = p.**key**) == key || (key != **null** && key.equals(k))))

==：判断内存地址， equals():要看具体的实现逻辑，不重写equals的话默认的是Object中的逻辑：

**public boolean** equals(Object obj) {  
 **return** (**this** == obj);  
}//还是判断内存地址

所以说如果是内存地址相同的话就判定为相等。 若内存地址不等，在判断equals方法是否为true，若为true则判定为相等。

关于hashmap 还有补充：（jdk7）当put一个元素的时候，如果是新插入的元素（不是替换已经存在的元素），若需要扩容，是先扩容，再插入，jdk8则是在插入元素后才判断是不是需要扩容，需要的话则扩容。关于扩容机制，jdk8中有很大的改进，不是每一个元素都重新计算hash值了 。

4.concurrenthashmap 的实现原理

答：

jdk7

分段+hashmap,分段加锁。

**static final class** Segment<K,V> **extends** ReentrantLock

每一个segment内部都有一个和hashmap类似的引用。concurrenthashmap

扩容不是针对所有的段，而是某一个段内部进行扩容。

当调用size()的方法的时候，是先统计两次各个段的modCount的和，若这两次统计值相同

则直接返回各个段内的元素数量的和，若统计的两次modCount的和不同，则对各个段进行加锁，计算各段内元素的数量的和并返回。

modCount: *The total number of mutative operations in this segment.*

此段中的可变操作总数（如新增，删除）

Jdk8:数组，链表，红黑树

5. 什么是Java虚拟机？为什么Java被称作是“平台无关的编程语言”？

Java虚拟机是一个可以执行Java字节码的虚拟机进程。Java源文件被编译成能被Java虚拟机执行的字节码文件。

Java被设计成允许应用程序可以运行在任意的平台，而不需要程序员为每一个平台单独重写或者是重新编译。Java虚拟机让这个变为可能，因为它知道底层硬件平台的指令长度和其他特性。

6.Iterator和ListIterator的区别是什么？

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。

7.HashMap和Hashtable有什么区别？

HashMap和Hashtable都实现了Map接口，因此很多特性非常相似。但是，他们有以下不同点：

HashMap允许键和值是null，而Hashtable不允许键或者值是null。

Hashtable是同步的，而HashMap不是。因此，HashMap更适合于单线程环境，而Hashtable适合于多线程环境。

HashMap提供了可供应用迭代的键的集合，因此，HashMap是快速失败的。另一方面，Hashtable提供了对键的列举(Enumeration)。

8.Enumeration接口和Iterator接口的区别有哪些？

Enumeration速度是Iterator的2倍，同时占用更少的内存。但是，Iterator远远比Enumeration安全，因为其他线程不能够修改正在被iterator遍历的集合里面的对象。同时，Iterator允许调用者删除底层集合里面的元素，这对Enumeration来说是不可能的。

9. JVM的永久代中会发生垃圾回收么？

垃圾回收不会发生在永久代，如果永久代满了或者是超过了临界值，会触发完全垃圾回收(Full GC)。如果你仔细查看垃圾收集器的输出信息，就会发现永久代也是被回收的。这就是为什么正确的永久代大小对避免Full GC是非常重要的原因。请参考下Java8：从永久代到元数据区

10．PreparedStatement比Statement有什么优势？

PreparedStatements是预编译的，因此，性能会更好。同时，不同的查询参数值，PreparedStatement可以重用。

11.解释下Servlet的生命周期。

Servlet生命周期可以分成四个阶段：加载和实例化、初始化、服务、销毁。

　　当客户第一次请求时，首先判断是否存在Servlet对象，若不存在，则由Web容器创建对象，而后调用init()方法对其初始化，此初始化方法在整个Servlet生命周期中只调用一次。

　　完成Servlet对象的创建和实例化之后，Web容器会调用Servlet对象的service()方法来处理请求。

　　当Web容器关闭或者Servlet对象要从容器中被删除时，会自动调用destory()方法。

12.doGet()方法和doPost()方法有什么区别？

doGet：GET方法会把名值对追加在请求的URL后面。因为URL对字符数目有限制，进而限制了用在客户端请求的参数值的数目。并且请求中的参数值是可见的，因此，敏感信息不能用这种方式传递。

doPOST：POST方法通过把请求参数值放在请求体中来克服GET方法的限制，因此，可以发送的参数的数目是没有限制的。最后，通过POST请求传递的敏感信息对外部客户端是不可见的。

13.什么是cookie？session和cookie有什么区别？

cookie是Web服务器发送给浏览器的一块信息。浏览器会在本地文件中给每一个Web服务器存储cookie。以后浏览器在给特定的Web服务器发请求的时候，同时会发送所有为该服务器存储的cookie。下面列出了session和cookie的区别：

无论客户端浏览器做怎么样的设置，session都应该能正常工作。客户端可以选择禁用cookie，但是，session仍然是能够工作的，因为客户端无法禁用服务端的session。

在存储的数据量方面session和cookies也是不一样的。session能够存储任意的Java对象，cookie只能存储String类型的对象。

14．代理

<https://www.jianshu.com/p/9a61af393e41?from=timeline&isappinstalled=0>

静态代理：代理类与委托类实现同一接口，并且在代理类中需要硬编码接口

JDK动态代理：只能够代理实现了接口的委托类，底层使用反射机制进行方法的调用

Cglib动态代理：可以在运行时对类或者是接口进行增强操作，且委托类无需实现接口，不能对final类以及final方法进行代理，底层将方法全部存入一个数组中，通过数组索引直接进行方法调用。

15.ArrayList和Vector的区别（是否有序、是否重复、数据结构、底层实现）

ArrayList和Vector都实现了List接口，他们都是有序集合，并且存放的元素是允许重复的。它们的底层都是通过数组来实现的，因此列表这种数据结构检索数据速度快，但增删改速度慢。

而ArrayList和Vector的区别主要在两个方面：

vector 是在方法上添加synchronize来实现线程安全

第一，线程安全。Vector是线程安全的，而ArrayList是线程不安全的。因此在如果集合数据只有单线程访问，那么使用ArrayList可以提高效率。而如果有多线程访问你的集合数据，那么就必须要用Vector，因为要保证数据安全。

第二，数据增长。ArrayList和Vector都有一个初始的容量大小，当存储进它们里面的元素超过了容量时，就需要增加它们的存储容量。ArrayList每次增长原来的0.5倍，而Vector增长原来的一倍。ArrayList和Vector都可以设置初始空间的大小，Vector还可以设置增长的空间大小，而ArrayList没有提供设置增长空间的方法。

16．List 和 Map 区别？（数据结构，存储特点）

这个要从两个方面来回答，一方面是List和Map的数据结构，另一方面是存储数据的特点。在数据结构方面，List存储的是单列数据的集合，而Map存储的是key、value类型的数据集合。在数据存储方面，List存储的数据是有序且可以重复的，而Map中存储的数据是无序且key值不能重复（value值可以重复）。

17. List、Map、Set三个接口，存取元素时，各有什么特点？

List与Set具有相似性，它们都是单列元素的集合，所以，它们有一个共同的父接口，叫Collection。Set里面不允许有重复的元素，所谓重复，即不能有两个相等（注意，不是仅仅是相同，==或者equals为true）的对象 ，即假设Set集合中有了一个A对象，现在我要向Set集合再存入一个B对象，但B对象与A对象equals相等，则B对象存储不进去。所以，Set集合的add方法有一个boolean的返回值，当集合中没有某个元素，此时add方法可成功加入该元素时，则返回true，当集合含有与某个元素equals相等的元素时，此时add方法无法加入该元素，返回结果为false。Set取元素时，没法说取第几个，只能以Iterator接口取得所有的元素，再逐一遍历各个元素。

List表示有先后顺序的集合， 注意，不是那种按年龄、按大小、按价格之类的排序。当我们多次调用add(Obj e)方法时，每次加入的对象就像火车站买票有排队顺序一样，按先来后到的顺序排序。有时候，也可以插队，即调用add(int index,Obj e)方法，就可以指定当前对象在集合中的存放位置。一个对象可以被反复存储进List中，每调用一次add方法，这个对象就被插入进集合中一次，其实，并不是把这个对象本身存储进了集合中，而是在集合中用一个索引变量指向这个对象，当这个对象被add多次时，即相当于集合中有多个索引指向了这个对象，如图x所示。List除了可以以Iterator接口取得所有的元素，再逐一遍历各个元素之外，还可以调用get(index i)来明确说明取第几个。

Map与List和Set不同，它是双列的集合，其中有put方法，定义如下：put(obj key,obj value)，每次存储时，要存储一对key/value，不能存储重复的key，这个重复的规则也是按equals比较相等。取则可以根据key获得相应的value，即get(Object key)返回值为key 所对应的value。另外，也可以获得所有的key的结合（map.keySet()），还可以获得所有的value的结合（map.values()），还可以获得key和value组合成的Map.Entry对象的集合（map.entrySet()）。

List 以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序

18. Collection框架中实现比较要实现什么接口

要实现比较有两种方式：第一种，实体类实现Comparable<T>接口，并实现 compareTo(T t) 方法，我们称为内部比较器。第二种，创建一个外部比较器，这个外部比较器要实现Comparator接口的 compare(T t1, T t2)。

19.fail-fast 和fail-safe

一：快速失败（fail—fast）

在用迭代器遍历一个集合对象时，如果遍历过程中对集合对象的内容进行了修改（增加、删除、修改），则会抛出Concurrent Modification Exception。

原理：迭代器在遍历时直接访问集合中的内容，并且在遍历过程中使用一个 modCount 变量。集合在被遍历期间如果内容发生变化，就会改变modCount的值。每当迭代器使用hashNext()/next()遍历下一个元素之前，都会检测modCount变量是否为expectedmodCount值，是的话就返回遍历；否则抛出异常，终止遍历。

注意：这里异常的抛出条件是检测到 modCount！=expectedmodCount 这个条件。如果集合发生变化时修改modCount值刚好又设置为了expectedmodCount值，则异常不会抛出。因此，不能依赖于这个异常是否抛出而进行并发操作的编程，这个异常只建议用于检测并发修改的bug。

场景：java.util包下的集合类都是快速失败的，不能在多线程下发生并发修改（迭代过程中被修改）。

二：安全失败（fail—safe）

采用安全失败机制的集合容器，在遍历时不是直接在集合内容上访问的，而是先复制原有集合内容，在拷贝的集合上进行遍历。

原理：由于迭代时是对原集合的拷贝进行遍历，所以在遍历过程中对原集合所作的修改并不能被迭代器检测到，所以不会触发Concurrent Modification Exception。

缺点：基于拷贝内容的优点是避免了Concurrent Modification Exception，但同样地，迭代器并不能访问到修改后的内容，即：迭代器遍历的是开始遍历那一刻拿到的集合拷贝，在遍历期间原集合发生的修改迭代器是不知道的。

场景：java.util.concurrent包下的容器都是安全失败，可以在多线程下并发使用，并发修改。

|  |  |  |
| --- | --- | --- |
|  | **Fail Fast Iterator** | **Fail Safe Iterator** |
| Throw ConcurrentModification Exception | Yes | No |
| Clone object | No | Yes |
| Memory Overhead | No | Yes |
| Examples | HashMap,Vector,ArrayList,HashSet | CopyOnWriteArrayList, ConcurrentHashMap |

20 哪些集合是线程安全的？

Vector,HashTable,Properties 是同步类，是线程安全的。

ConcurrentHashMap，CopyOnWriteArrayList是并发的线程安全

ArrayBlockQueue