wxy

###### hashmap和hashtable的区别？

1. **线程是否安全：**hashmap非线程安全，hashtable线程安全，因为hashtable的内部方法基本都经过synchronized修饰。
2. **效率**：因为线程安全的问题，hashmap比hashtable效率高。
3. **关于null：**hashmap可以存储 null 的 key 和 value，但 null 作为键只能有一个，null 作为值可以有多个；hashtable不允许有 null 键和 null 值，否则会抛出 NullPointerException。
4. **初始容量大小不同**：如果创建时不指定容量初始值，hashmap默认初始大小为16，hashtable默认初始大小为11；如果创建时指定了容量初始值，hashmap会将其扩充为2的幂次方大小，hashtable直接使用给定的大小。
5. **扩容时不同：**hashmap每次扩容为之前的2倍，hashtable每次扩容为之前的2n+1
6. **底层数据结构：**hashmap为数组+链表/红黑树，hashtable为数组+链表。JDK1.8 以后的 HashMap 在解决哈希冲突时有了较大的变化，当链表长度大于阈值（默认为 8）时，将链表转化为红黑树（将链表转换成红黑树前会判断，如果当前数组的长度小于 64，那么会选择先进行数组扩容，而不是转换为红黑树），以减少搜索时间。Hashtable 没有这样的机制。

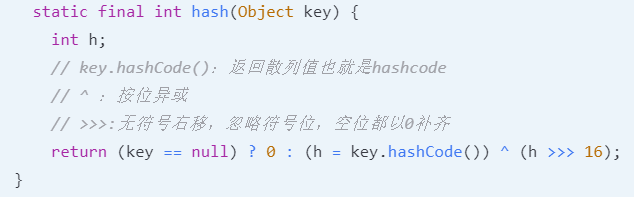
PS：之所以hashmap长度是2的幂次方的原因——

hashmap添加<key,value>对时确定添加到数组的哪个下标位置的步骤如下：调用一个hash()方法，先判断key是否为null，为null的话直接令hash=0；不为null的话，先取得key的hashcode，然后计算hash=hashcode^(hashcode>>>16)，然后用hash%当前数组长度取得下标，但%取模操作比较慢，使用位运算会很快，当数组长度为2的幂次方时，（数组长度-1）的二进制数刚好全是1，所以可以把这个取模运算转换为hash&(数组长度-1)这样的位运算。

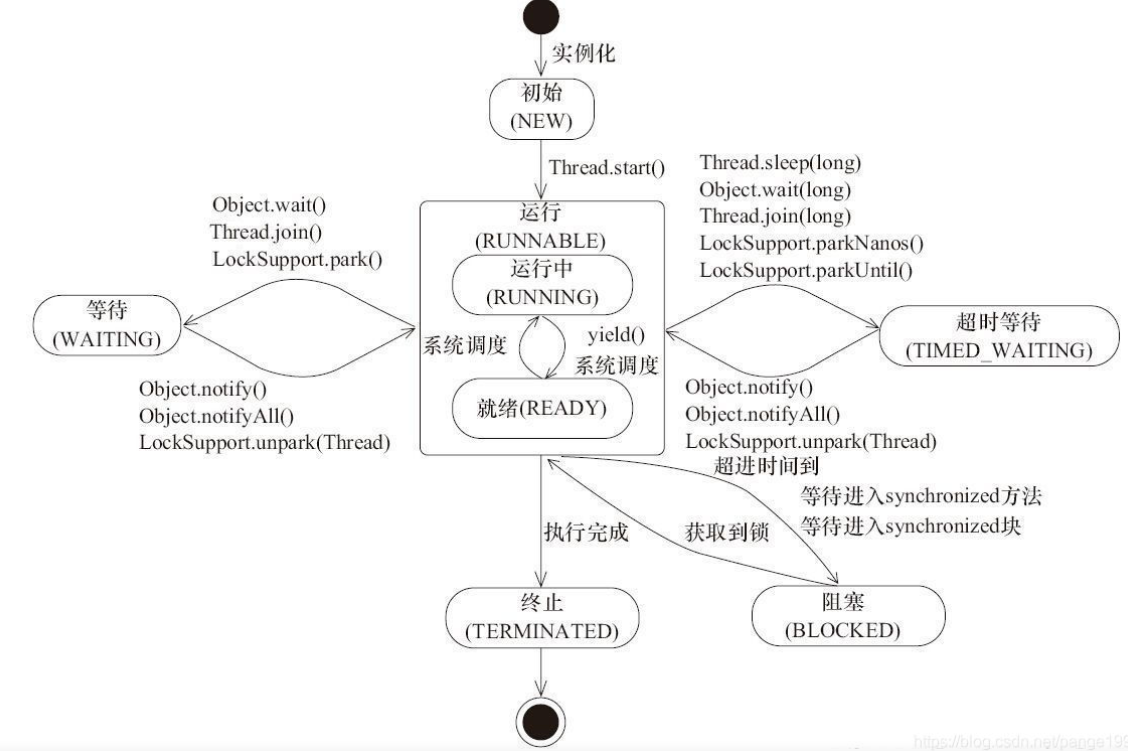
###### hashmap的key类型有什么要求，key为什么可以为null？

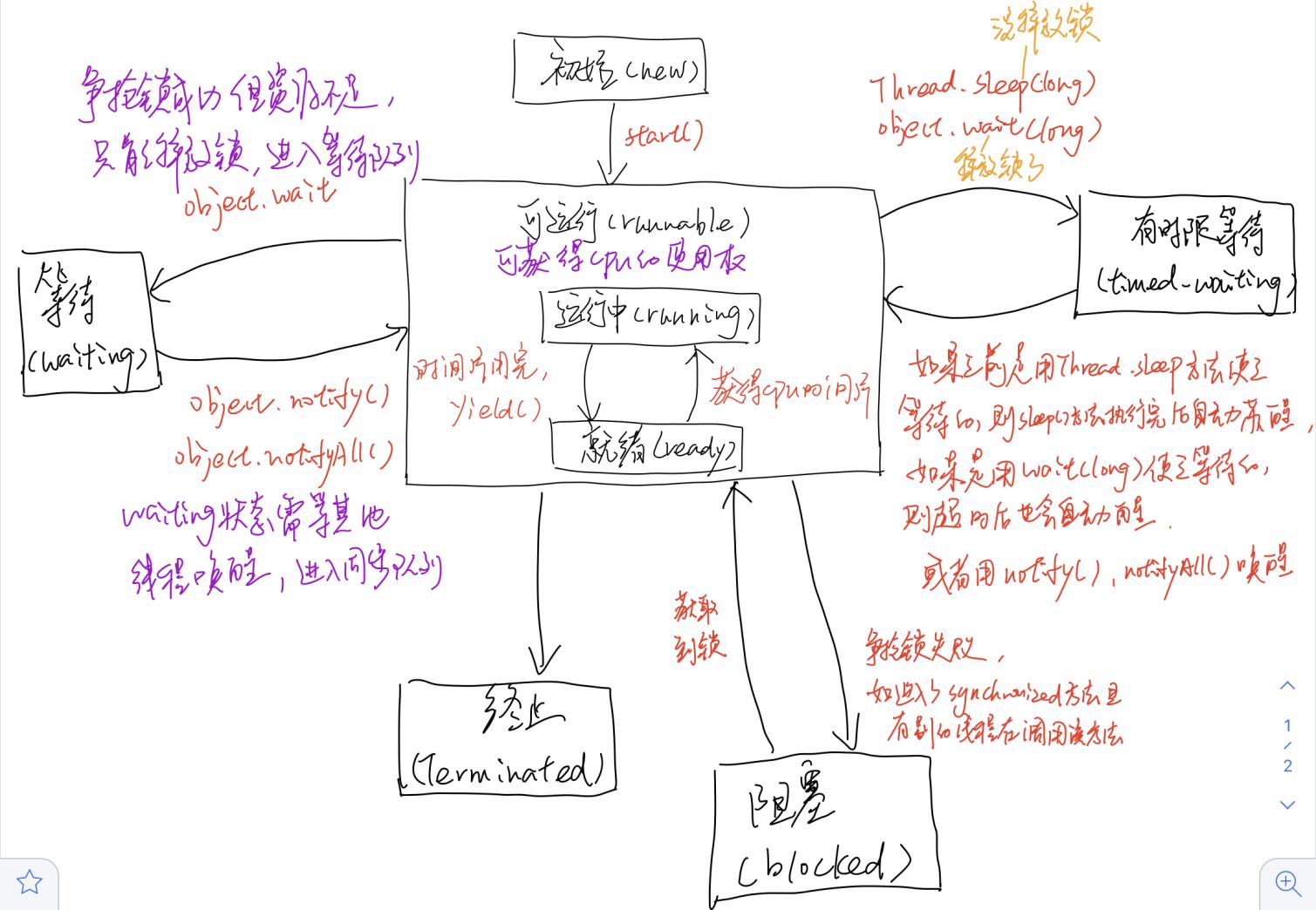
Map的key和value都必须是基本数据类型的包装类或自定义的类（也就是引用类型），或者是null，不能是基本数据类型。

key可以为null的原因：hashmap在put的时候调用一个hash方法，先判断key是否为null，为null的话直接令hash=0，不为null才取hashcode值然后再用hashcode^(hashcode>>>16)算hash。反观hashtable直接就令hash=key.hashcode()了，当key为null时没有hashcode，所以会报空指针异常。



###### java线程有哪几种状态，以及如何转换线程的状态





* 初始：这个线程对象刚刚被new
* 可运行：这个线程对象位于可运行线程池中，等待被线程调度选中、获得CPU的使用权。新建线程调用Thread.start()方法就进入可运行状态
  + 分为运行中和就绪两种子状态，当获得CPU时间片时为运行中，时间片用完就调用该线程的yield()方法变为就绪
* 阻塞：可运行状态争抢锁失败，如等待进入synchronized代码块或被lock，进入阻塞状态，当获取到锁时回到可运行状态
* 等待：可运行阶段争抢锁成功但资源不够，调用该线程对象的wait()方法，进入等待队列，要被其他线程调用notify()或notifyAll()的时候才被唤醒，转为可运行状态
* 有时限等待（超时等待）：也是可运行阶段争抢锁成功但资源不够，调用该线程对象的wait(long)方法，或者调用静态方法Thread.sleep(long)。

【wait和sleep的区别：

1. wait()释放了锁，sleep()没释放
2. sleep()方法执行完成后，线程会自动苏醒，wait(long timeout) 超时后线程会自动苏醒，普通的wait()一定要notify()或notifyAll()唤醒
3. sleep() 是 Thread 类的静态本地方法，wait() 则是 Object 类的本地方法】

所以若使线程超时等待的是sleep，那sleep方法执行完后线程自动苏醒回到可执行状态；若使线程超时等待的是wait(long)，那超时后线程自动苏醒回到可执行状态。

* 终止：线程已执行完毕或主线程的main()方法完成。

注意：这里文字部分写的不全，全的看第一张图

进程的状态

和线程很像，但分为5种状态：

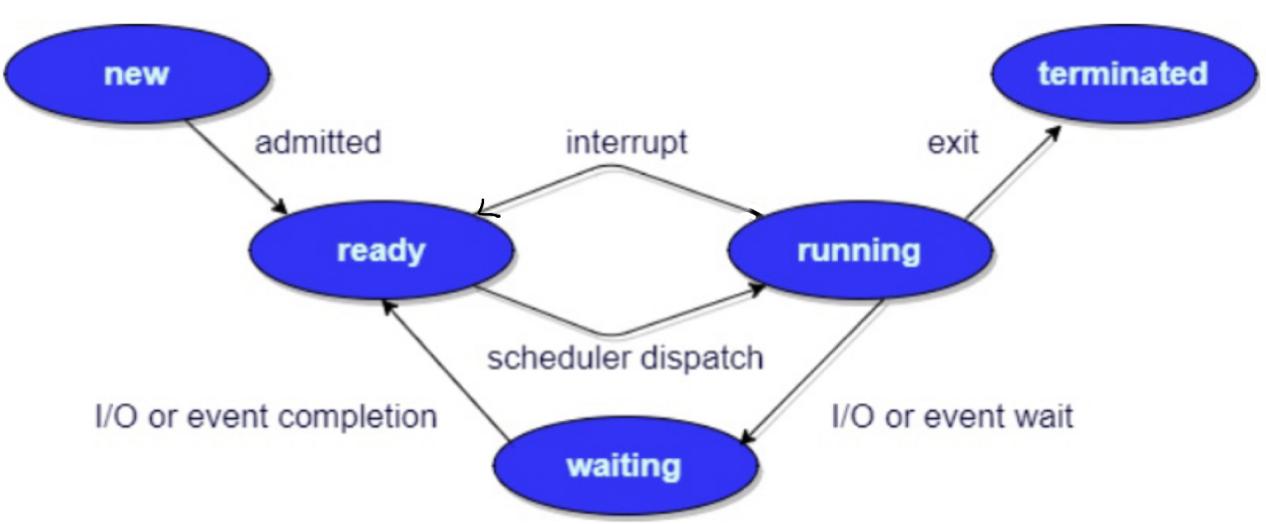
· **创建状态(new)** ：进程正在被创建，尚未到就绪状态。

· **就绪状态(ready)** ：进程已处于准备运行状态，即进程获得了除了处理器之外的一切所需资源，一旦得到处理器资源(处理器分配的时间片)即可运行。

· **运行状态(running)** ：进程正在处理器上上运行(单核 CPU 下任意时刻只有一个进程处于运行状态)。

· **阻塞状态(waiting)** ：又称为**等待状态**，进程正在等待某一事件而暂停运行如等待某资源为可用或等待 IO 操作完成。即使处理器空闲，该进程也不能运行。

· **结束状态(terminated)** ：进程正在从系统中消失。可能是进程正常结束或其他原因中断退出运行。



Collection接口下有哪几种类型，每种下面常用的有哪些

List、Set、Queue、Map

* List：（用于需要顺序的情况）元素有序、可重复。
  + Vector：Object[] 数组
  + ArrayList： Object[] 数组
  + LinkedList： 双向链表(JDK1.6 之前为循环链表，JDK1.7 取消了循环)
* Set：（用于要求元素不可重复的情况）元素无序、不可重复
  + HashSet(无序，唯一): 基于 HashMap 实现的，底层采用 HashMap 来保存元素
  + LinkedHashSet: LinkedHashSet 是 HashSet 的子类，并且其内部是通过 LinkedHashMap 来实现的。有点类似于我们之前说的 LinkedHashMap 其内部是基于 HashMap 实现一样，不过还是有一点点区别的
  + TreeSet(有序，唯一): 红黑树(自平衡的排序二叉树)，底层通过TreeMap实现
* Queue：（用于要求按特定规则排序的情况）元素按特定规则排序、有序、可重复
  + PriorityQueue: Object[] 数组来实现二叉堆，与一般Queue 的区别在于元素出队顺序是与优先级相关的，即总是优先级最高的元素先出队。
  + ArrayQueue: Object[] 数组 + 双指针

【Queue还有一种子接口叫Deque，Queue是单端队列，元素只能从一段进、另一端出，Deque是双端队列，在队列的两端均可以插入或删除元素。】

LinkedList这个类很特殊，既实现了List，又实现了Deque

* Map：（用于处理键值对）处理键值对<key,value>，key 是无序的、不可重复的，value 是无序的、可重复的，每个键最多映射到一个值。
  + HashMap
  + LinkedHashMap：和HashMap相比增加了一条双向链表，使得可以保持键值对的插入顺序。
  + HashTable
  + TreeMap：红黑树（自平衡的排序二叉树）

JDBC怎么建立数据库连接

mybatis的$和#有什么区别

**进程内存结构**

**32位计算机寻址空间**

**java包装类和基本类型的区别**

**抽象类和接口区别**