Java

一、集合

1、HashMap和HashTable

HashMap是java中哈希表的标准实现方式，通过链地址的方法处理冲突

HashTable是线程安全的哈希表，通过在HashMap实现的部分方法中添加synchronized关键字来实现线程安全

此外，HashTable不允许值为null

具体介绍：http://blog.csdn.net/shohokuf/article/details/3932967

2、Concurrent包内的部分具体实现

Vector、HashTable等使用synchronized实现的，因此效率较低。Concurrent包内有效率更高额实现方式，分别为ConcuuentArrayList和ConcurreentHashMap，其中ConcurrentArrayList使用读写锁（乐观锁）和volatile等轻量级并行方式来完成，但是适合批量增删的情况。CurrentHashMap把哈希表变成了多个segment，每次只锁住需要的那一部分，以此提高效率

3、集合的内部实现

ArrayList：数组，每次会进行扩容

LikedList：双向链表，get(i)实现通过判断i与1/2legth比较，确定是从什么地方开始查找

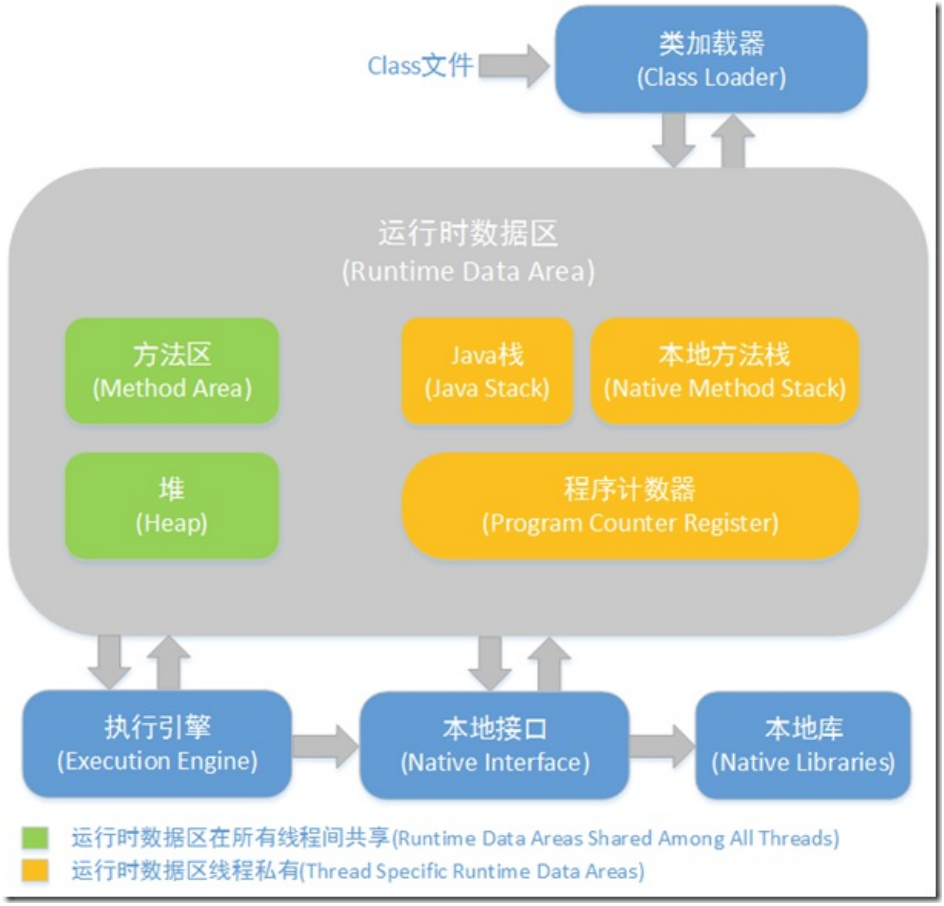
二、设计模式

1、单例模式

http://blog.csdn.net/jason0539/article/details/23297037/

三、java内存相关

1、jvm的内存模型，jvm的内存模型可以向下面来表示



方法区和堆是所有线程共享的，方法区存储的是常量和类相关的信息，堆中是新分配的对象。

Java栈中、本地方法栈和程序计数器是线程私有的，Java栈中存储的是一些基本类型、方法运行时的参数、引用、返回值等

堆中主要有两部分，分别是young代和old代，其中young代又分为eden、fromsurivor和tosurivor，比例为8：1：1，其中生命周期短的放在young代，生命周期长的对象放在old代中，surivor是young代和old代的过渡空间

2、内存分配过程

1. JVM试图为新创建的对象在Eden区域中开辟一块空间
2. 当Eden中空间充足时，申请结束。否则跳至3
3. JVM试图释放Eden中不活动的对象（采用复制回收算法），如果此时Edem中的存储还不够，则把部分活跃对象放入surivor区
4. surivor作为Eden区域和old代的中间区域，当Old区域空间充足，则把surivor区域对象移至old，否则会保留在surivor区
5. 当Old区空间不足时，执行垃圾回收（标记-压缩）
6. 完全垃圾回收之后，若还没有地方存放新对象（新对象可以直接进入Old区），则抛异常

3、垃圾回收算法

标记-清除：最基本，最low

复制回收：需要额外空间，且对象生命周期长的话，对象复制移动代价太大

标记-压缩：适合大多数生命周期长的对象

按代回收：young代用复制(minor gc)，old代用标记-压缩(full gc)，采用System.gc调用的是full gc

4、垃圾收集器

Serial：单线程

PartNew：多线程,young代并行，old代串行

参考资料：http://www.cnblogs.com/ityouknow/p/5610232.html

http://www.cnblogs.com/ityouknow/p/5614961.html

http://blog.csdn.net/u012152619/article/details/46968883

http://blog.csdn.net/zhangpengju999/article/details/11773183

四、锁

java中的锁分为两类，悲观锁和乐观锁，悲观锁总是会锁住全部的内容，无论当前是否有竞争条件，比如synchronized和ReentrantLock，Concurrent包中的相关实现采用了乐观锁，以及Atomatic的基础类型。

乐观锁这样认为，他不管当前的线程状态，他总是认为当前共享内存不存在竞争状态，因此直接拿过来操作，如果发现了冲突，则进行补偿操作，对于CAS来说，就是不断尝试，直到不发生冲突为止

乐观锁和悲观锁的介绍：http://m.blog.csdn.net/article/details?id=58323471

CAS介绍：http://blog.csdn.net/hsuxu/article/details/9467651

五、IO

IO中byte转string的两个类分别是OutputStreamWriter和IntputStreamReader

OutputStreamWriter将输出的字符流变成字节流

InputStreamReader将输入的字节流转换成字符流

六、多线程

1、java线程的几种状态

http://www.cnblogs.com/mengdd/archive/2013/02/20/2917966.html

七、java基础

1、finalize()方法

在内存回收之前调用，且不一定能调用，且Java不推荐使用，

http://blog.csdn.net/shanghui815/article/details/6787855