根据J2ME虚拟机对程序编写的优化方式Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E6_A0_B9_E6_8D_AEJ2ME_c104_644444.htm 1、关于虚拟机 我认为，目前客户端虚拟机技术，应该说是发展到一个转折点，未来可能会出现重大的技术突破。目前无论是Java还是.net基本上采用的都是分代式垃圾回收和分支预测JIT技术。因此目前这两个虚拟机的性能相差不是很大，因此对于程序的优化，基本上真对这两种技术来进行。 关于分代式垃圾回收 分代式垃圾回收技术应该是目前客户端虚拟机垃圾回收技术的主流，就是说增加对寿命短对象的收集，而减少对长寿命对象的收集。这里使用术语第一代和第二代的术语来描述。第一代应当说是程序新分配的内存，从统计数据来看，对象越年轻，被回收的可能性就越大，因此，对第一代的垃圾回收进行的频繁一些。而一旦进行了第一代的垃圾回收，未被回收的对象将会成为第二代对象。对于第二代对象的回收，由于它的生命期更长，因此在到达第二代对象空间阈值的时候并不会收集，这样就减少了回收的次数。防止垃圾回收带来的系统停顿。 关于分支预测 JIT技术在虚拟机中已经被证明能极大地加快程序速度，而目前基本上采用的是带有分支预测技术的JIT编译器，因此越背频繁执行的语句，就越有可能被JIT编译器编译，从而加快程序的速度。但是目前的对J2ME虚拟机来说，性能有限，因此JIT的应用的也有限，在程序写作上，应当帮助进行JIT 编译。 2、关于J2ME的虚拟机 J2ME虚拟机最严重影响程序性能的就是进行垃圾回收操作，一旦进行垃圾回收，程序的主线程一般会暂时挂起，以方便进行垃圾回

收操作。在程序上的表现就是程序出现暂时的停顿，这将极大地影响用户的体验。因此必须尽量减少垃圾回收进行的次数，近最大努力控制垃圾回收的执行，尽量减少垃圾回收执行的时间。 J2ME虚拟机中JIT技术的应用，目前我没有发现有显著的文章来进行描述，由于JIT技术需要实时编译并且耗费内存较多，因此，我认为 MIDP1.0中并没有采用JIT技术。而MIDP2.0中有可能采用JIT技术，但是J2ME的JIT技术与J2SE的性能应该不是一个数量级上的，由于CPU性能有限，注定无法做复杂的JIT编译，而且由于JIT耗费内存，也注定无法对代码进行大规模的JIT编译。 3、J2ME(MIDP)对于这这两种技术的优化 针对垃圾回收的优化 垃圾回收优化的核心思想就是减少垃圾回收的次数，增加垃圾回收中资源的回收量。 先说垃圾回收次数的优化，在上文中已经说明，分代式垃圾回收，第一代的垃圾回收进行的次数较频繁，因此垃圾回收首先针对第一代垃圾回收，就是说应该避免在成员函数中产生新的对象。 这里在书中有明显的应用，在滚屏游戏的设计中，需要进行矩形的判断，这里采用的方式都是定义的到类中的矩形，而不是定义函数矩形，在碰撞判断中调用的矩形都是在类的构造函数中生成的，并不在函数中生成，由于需要频繁调用碰撞检测，因此如果定义在函数中将会增加垃圾回收的次数。具体情况如下： class Sprite { public Rect rect1 = new Rect(). public Rect rect2 = new Rect(). bool checkCollision() { //位置判断后，进行初始化，但是不分配产生新的对象 rect1.x,rect1.y,rect1.dx,ect1.dy rect2.x,rect2.y,rect2.dx,rect2.dy } } 这样就显著减少了对象的生成，注意这里还有一个优化的地方，基本上每次判断，dx,dy值是固定的，因此在构造函数中

应当初始化dx,dy，这样将会减少初始化。因此书中的程序中有很多看起来不应当定义为类程序的变量，都被定义为类成员，这极大地减少了垃圾回收的次数。 而对于二代的垃圾回收，就是在程序中不要随时把无用的资源置为null，这样可能会激发二代回收。 而对于尽可能增加垃圾回收的资源回收量，使调用显示垃圾回收，在进行大规模的资源更换的时候。这个优化技术书中没有说明，但与程序源代码中有体现。对于类程序对象，不要无故置为null，而是在资源更换的时候，一般来说这个时候是更换关卡的时候，此时应当把上关卡中不再使用的资源全部置为 null，然后显示调用垃圾回收。再栽入新的资源，这个时候新的资源能够顺利栽入，旧的资源也能够顺利回收。而由于关卡切换，进行稍微的等待也是可行的，不会影响用户体验。 针对JIT编译的优化 JIT编译的优化就是用户模拟JIT编译器来进行程序编译的优化，对于频繁执行的地方，尽量优化。对于程序中的分支判断，要把经常调用的分支写到判断前面。而在我前面的文章中提到的关于成员函数使用get和set，这个在JIT编译器中基本上是被优化掉的，没有多少意义。如果不是太频繁，也不用时全部应用破坏程序结构，只把最频繁的进行优化即可。 这里还有一个优化的地方，对于特别频繁的执行语句，如下的取得数据成员的方式： a.b，如果是int型，最好使用一个临时变量 int temp = a.b. 这将减少类程序的取得时间，虽然只有几个时钟周期，但是对于特别频繁的调用和复杂算法，性能还是有一定的提升。 4、总结 这里提到的优化措施每一处对程序的影响都很小，但是积少成多，水滴石穿，注意的地方多了，就会对程序性能产生影响，并且很多优化是全局性的，这在程序的架

构设计之初就应该考虑好的。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com