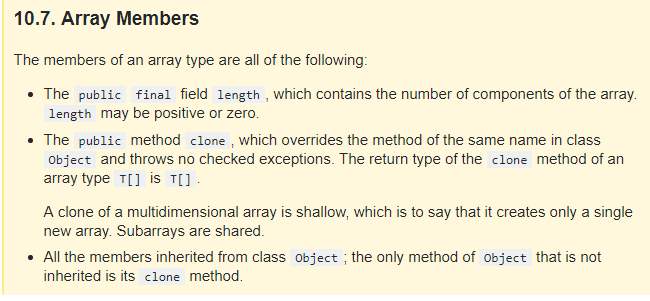
# 数组



10.7。数组成员 数组类型的成员都是以下的: 公共final字段长度，其中包含数组的组件数量。长度可能为正或为零。

public方法克隆，它覆盖类对象中相同名称的方法，并且不抛出检查异常。数组类型T的克隆方法的返回类型为T。 多维数组的克隆是很浅的，也就是说它只创建一个新数组。子队列共享。

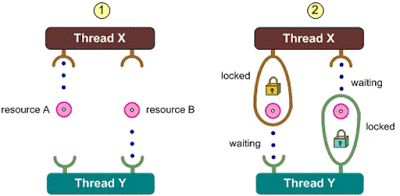
所有的成员都继承了类对象;唯一没有inh的对象方法

# 线程

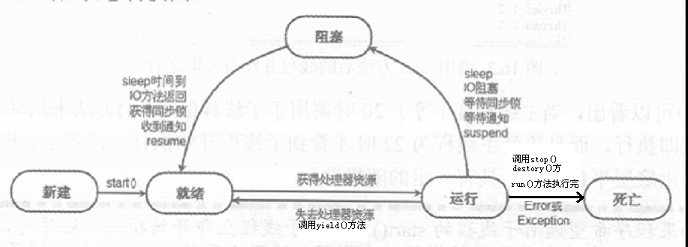
18th Of Row 15, west of Bajie Road, Xugezhuang Town, Fengnan District, Tangshan City, Hebei Province, P.R.China

## 1怎样判断线程是否拿着锁

1)I thought about **IllegalMonitorStateException**which [wait() and notify() methods](http://javarevisited.blogspot.com/2011/05/wait-notify-and-notifyall-in-java.html)throw when they get called from non-synchronized context so I said I would call newspaper.wait() and if this call [throws an exception](http://javarevisited.blogspot.com/2012/02/difference-between-throw-and-throws-in.html) it means thread in Java is not holding the lock, otherwise thread holds the lock.



2) Later I discovered that thread is a static method called **holdsLock(Object obj)** which returns true or false based on whether threads holds a lock on the object passed.  
  
Read more: [http://javarevisited.blogspot.com/2010/10/how-to-check-if-thread-holds-lock-on.html#ixzz4ndSfd0DT](http://javarevisited.blogspot.com/2010/10/how-to-check-if-thread-holds-lock-on.html" \l "ixzz4ndSfd0DT)



当发生如下情况是，线程会从运行状态变为阻塞状态：

①、线程调用sleep方法主动放弃所占用的系统资源

②、线程调用一个阻塞式IO方法，在该方法返回之前，该线程被阻塞

③、线程试图获得一个同步监视器，但更改同步监视器正被其他线程所持有

④、线程在等待某个通知（notify）

⑤、程序调用了线程的suspend方法将线程挂起。不过该方法容易导致死锁，所以程序应该尽量避免使用该方法。

Java多线程中的死锁  
死锁是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。这是一个严重的问题，因为死锁会让你的程序挂起无法完成任务，死锁的发生必须满足以下四个条件：

* 互斥条件：一个资源每次只能被一个进程使用。
* 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
* 不剥夺条件：进程已获得的资源，在末使用完之前，不能强行剥夺。
* 循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。

避免死锁最简单的方法就是阻止循环等待条件，将系统中所有的资源设置标志位、排序，规定所有的进程申请资源必须以一定的顺序（升序或降序）做操作来避免死锁。[这篇教程](http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html" \t "http://www.importnew.com/_blank)有代码示例和避免死锁的讨论细节。

多个线程读取单个文件的不同部分

package com.alibaba.middleware.race;

import java.io.IOException;

import java.io.RandomAccessFile;

public class Test2 {

public static void main(String[] args) throws IOException {

byte[] th1 = "线程1".getBytes();

byte[] th2 = "线程2".getBytes();

byte[] th3 = "线程3".getBytes();

final int len = th1.length;

final RandomAccessFile raf = new RandomAccessFile(

"F:/tianchi/index.data", "rw");

raf.seek(1);

raf.write(th1);

raf.seek(200);

raf.write(th2);

raf.seek(300);

raf.write(th3);

Runnable runnable1 = new Runnable() {

@Override

public void run() {

while (true) {

byte[] b = new byte[len];

try {

raf.seek(1);

raf.read(b);

if ("线程1".equals(new String(b))) {

System.out.println("ok");

} else {

System.out.println("线程1-" + new String(b));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

};

Runnable runnable2 = new Runnable() {

@Override

public void run() {

while (true) {

try {

byte[] b = new byte[len];

raf.seek(200);

raf.read(b);

if ("线程2".equals(new String(b))) {

System.out.println("ok");

} else {

System.out.println("线程2-" + new String(b));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

};

Runnable runnable3 = new Runnable() {

@Override

public void run() {

while (true) {

try {

byte[] b = new byte[len];

raf.seek(300);

raf.read(b);

if ("线程3".equals(new String(b))) {

System.out.println("ok");

} else {

System.out.println("线程3-" + new String(b));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

};

for (int i = 0; i < 20; i++) {

new Thread(runnable1).start();

new Thread(runnable2).start();

new Thread(runnable3).start();

}

System.out.println("主线程完了");

}

}

-################

package com.alibaba.middleware.race;

import java.io.IOException;

import java.io.RandomAccessFile;

public class Test2 {

public static void main(String[] args) throws IOException {

byte[] th1 = "线程1".getBytes();

byte[] th2 = "线程2".getBytes();

byte[] th3 = "线程3".getBytes();

final int len = th1.length;

final RandomAccessFile raf = new RandomAccessFile(

"F:/tianchi/index.data", "rw");

raf.seek(1);

raf.write(th1);

raf.seek(200);

raf.write(th2);

raf.seek(300);

raf.write(th3);

Runnable runnable1 = new Runnable() {

@Override

public void run() {

while (true) {

byte[] b = new byte[len];

try {

synchronized (raf) {

raf.seek(1);

raf.read(b);

}

if ("线程1".equals(new String(b))) {

System.out.println("ok");

} else {

System.out.println("线程1-" + new String(b));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

};

Runnable runnable2 = new Runnable() {

@Override

public void run() {

while (true) {

try {

byte[] b = new byte[len];

synchronized (raf) {

raf.seek(200);

raf.read(b);

}

if ("线程2".equals(new String(b))) {

System.out.println("ok");

} else {

System.out.println("线程2-" + new String(b));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

};

Runnable runnable3 = new Runnable() {

@Override

public void run() {

while (true) {

try {

byte[] b = new byte[len];

synchronized (raf) {

raf.seek(300);

raf.read(b);

}

if ("线程3".equals(new String(b))) {

System.out.println("ok");

} else {

System.out.println("线程3-" + new String(b));

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

};

for (int i = 0; i < 20; i++) {

new Thread(runnable1).start();

new Thread(runnable2).start();

new Thread(runnable3).start();

}

System.out.println("主线程完了");

}

}