守护线程（deemon）与线程联合（join）

1. 线程一般分为**守护线程（Daemon Thread）和非守护线程**，其中非守护线程又称为**用户线程（User Thread）**。线程创建后，默认为非守护线程，故一般不需要调用setDaemon（false）方法声明此线程为非守护线程。**只有在需要设置为守护线程时，调用setDaemon（true）即可。**
2. Daemon Thread 守护线程：与用户线程的区别，就是在所有的用户线程运行结束时，守护线程即使还有需要执行的语句，守护线程也要立刻结束运行。故守护线程一般做一些不是很严格的工作，线程的随时结束不会产生不良后果。
3. daemon 英 ['diːmən] 美 ['dimən] n. 守护进程；后台程序
4. 要想把一个线程设置为守护线程，必须在**调用start（）方法之前**调用setDaemon（true）方法。可以利用isDaemon（）来判断这个线程是否是守护线程。

注意：一个线程必须在运行之前设置自己是否是守护线程，即**setDaemon（true）**必须在start方法的前面进行调用。





1. 示例：

public class DaemonThreadTest {

public static void main(String[] args) {

Thread th1 = new Thread(new Runnable() {

@Override

public void run() {

int i = 0;

while (true) {

System.out.println(

Thread.currentThread().getName() + " " + i++);

}

}

}, "线程第一个");//利用封装任务的方法创建线程

Thread th2 = new Thread("线程第二个") {

public void run() {

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

System.out.println(

Thread.currentThread().getName() + " " + i);

}

}

};//利用继承的方法创建线程

th1.setDaemon(true);

th1.start();

th2.start();

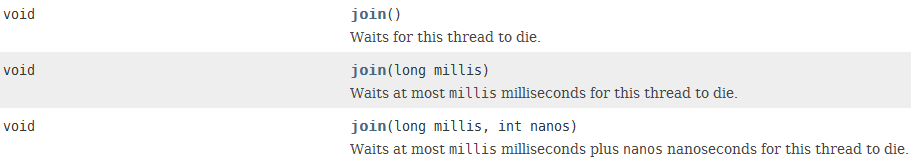
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **线程联合**，也叫线程合并：一个线程A在占有CPU资源期间，可以让其他线程调用**join（）方法和本线程联合**，如B.join();，我们称之为线程A在运行期间联合了B。
2. **如果线程A在占有CPU资源期间一旦联合了线程B，那么线程A将立刻中断执行，进入了阻塞状态，一直等到线程B执行结束，线程A再重新排队等待CPU资源，以便恢复执行。**
3. **join方法**里面又调用了（invoke）wait方法，使当前线程等待，所以B.join（）方法需要写在线程A的run方法中，只有这样线程A才能让出CPU和同步锁。
4. **join方法**：

有三个：**join（） 、join（long millis） 、join（long millis,int nanos）** 。

注意：**join（）方法等价于join（0）。**



1. elapsed v. 时间过去；消逝（elapse的过去分词）adj. 过去的；经过的
2. 示例：

public class DaemonThreadTest {

static Thread A ,B;

public static void main(String[] args) {

A = new Thread(new Runnable() {

@Override

public void run() {

for (int i = 0; i < 1000; i++) {

if(i == 201) {

try {

B.join();//A只有输出到200，变为等待状态，直到B执行完毕。

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(

Thread.currentThread().getName() + " " + i);

}

}

}, "线程第一个");

B = new Thread("线程第二%%%%%%%%%%%%%个") {

public void run() {

for (int i = 0; i <= 300; i++) {

System.out.println(

Thread.currentThread().getName() + " " + i);

}

}

};

A.start();;

B.start();

}

1. **yield方法：暂停当前线程，执行其他线程。**