Java提供对多线程程序的完全控制， 也可以开发一个可以根据您的要求完全暂停，恢复或停止的多线程程序。 有各种静态方法可以用于线程对象来控制它们的行为。 下表列出了这些方法 -

编号	方法	说明描述
1	public void suspend()	该方法使线程处于挂起状态，可以使用resume()方法恢复。
2	public void stop()	该方法使线程完全停止。
3	public void resume()	该方法恢复使用suspend()方法挂起的线程。
4	public void wait()	导致当前线程等到另一个线程调用notify()。
5	public void notify()	唤醒在此对象监视器上等待的单个线程。

请注意，最新版本的Java已经不再使用suspend()，resume()和stop()方法，因此您需要使用可用的替代方法。

1、线程间通信

当您开发两个或多个线程交换一些信息的应用程序时，线程间通信很重要。

有三个简单的方法和一个小技巧，使线程通信成为可能。 所有三种方法都列在下面 -

编号	方法	描述
1	public void wait()	使当前线程等到另一个线程调用notify()方法。
2	public void notify()	唤醒在此对象监视器上等待的单个线程。
3	public void notifyAll()	唤醒所有在同一个对象上调用wait()的线程。

这些方法已被实现为Object中的最终(final)方法，因此它们在所有类中都可用。 所有这三种方法只能从同步上下文中调用。

2、死锁

死锁描述了两个或多个线程等待彼此而被永久阻塞的情况。 当多个线程需要相同的锁定但以不同的顺序获取时，会发生死锁。 Java多线程程序可能会遇到死锁状况，因为synchronized关键字会导致执行线程在等待与指定对象相关联的锁定或监视时出现阻止情况。

3、ThreadLocal类

ThreadLocal类用于创建只能由同一个线程读取和写入的线程局部变量。 例如，如果两个线程正在访问引用相同threadLocal变量的代码，那么每个线程都不会看到任何其他线程操作完成的线程变量。

线程方法

以下是ThreadLocal类中可用的重要方法的列表。

编号	方法	描述
1	public T get()	返回当前线程的线程局部变量的副本中的值。
2	protected T initialValue()	返回此线程局部变量的当前线程的“初始值”。
3	public void remove()	删除此线程局部变量的当前线程的值。
4	public void set(T value)	将当前线程的线程局部变量的副本设置为指定的值。

4、ThreadLocalRandom类

java.util.concurrent.ThreadLocalRandom是从jdk 1.7开始引入的实用程序类，当需要多个线程或ForkJoinTasks来生成随机数时很有用。 它提高了性能，并且比Math.random()方法占用更少的资源。

ThreadLocalRandom方法

以下是ThreadLocalRandom类中可用的重要方法的列表。

编号	方法	说明
1	public static ThreadLocalRandom current()	返回当前线程的ThreadLocalRandom。
2	protected int next(int bits)	生成下一个伪随机数。
3	public double nextDouble(double n)	返回伪随机，均匀分布在0(含)和指定值(独占)之间的double值。
4	public double nextDouble(double least, double bound)	返回在给定的least值(包括)和bound(不包括)之间的伪随机均匀分布的值。
5	public int nextInt(int least, int bound)	返回在给定的least值(包括)和bound(不包括)之间的伪随机均匀分布的整数值。
6	public long nextLong(long n)	返回伪随机均匀分布的值在0(含)和指定值(不包括)之间的长整数值。
7	public long nextLong(long least, long bound)	返回在给定的最小值(包括)和bound(不包括)之间的伪随机均匀分布的长整数值。
8	public void setSeed(long seed)	设置伪随机的种子值，抛出UnsupportedOperationException异常。