多线程支持和线程安全（2） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/141/2021_2022__E5_A4_9A_E7_BA_BF_E7_A8_8B_E6_c29_141242.htm JDK平台中自己实现了的一套更为完善的线程模型，因此它的这套线程模型是可能独立于，也有区别于操作系统实现的线程模型。如图10-58所示，它包含的状态有运行（Running）、就绪（Ready）、暂停（Suspended）、休眠（Sleeping）、阻塞（Blocked）以及监控状态（Monitor States）。这其中线程的运行状态、就绪状态、暂停状态、休眠状态以及阻塞状态都比较容易理解，另外有一个比较特殊的线程状态，也即监控状态，它表示线程正处于查询对象锁的管理状态下，稍后会继续深入阐述这一特殊的状态。 JDK平台中，线程之间的互斥控制非常易于实现，因为Java语言对它有很好的支持，也即通过语言关键字synchronized来定义一个互斥访问的临界区。关键字synchronized的使用非常灵活，它定义的临界区资源的颗粒度可大可小，例如可以声明某个类的某个方法是被互斥访问的临界区，也可以定义某个局部作用域为一个临界区，甚至可以直接锁定某个对象为临界区资源。 而JDK平台中线程之间的同步控制，是通过wait()和notify()方法来实现的，实际上wait()方法的调用相当于执行P操作，而notify()方法的调用则相当于操作V，有关P，V操作的概念，请参阅操作系统中关于进程间的同步控制与通信设计等小节中的内容。另外为了提高安全性，JDK平台中还定义了一个notifyAll()的方法，它用于通知并释放所有处于阻塞等待状态中的线程，便于这些线程有机会重新获得退出的机会，避免了一些不必要的死

锁。 MFC中拥有一些专门实现线程间互斥和同步控制的对象，而与这形成鲜明对比的是，JDK平台中则没有提供这些类似的专有控制对象，这非常有意思，实际上JDK平台中的wait()和notify()方法是Object类（根基类）对象所提供的方法实现，所以说，程序中所有对象的运行实例都拥有wait()和notify()方法。并且它的同步与互斥有着非常紧密的联系，如图10-59所示，这是JDK平台中线程之间互斥与同步的转换模型。 