通信方式：

1. 同步
2. while轮询的方式
3. wait/notify机制
4. 管道通信

### 1、线程间的通信方式

#### 1.1 同步

这里的同步是指多个线程通过synchronized关键字这种方式来实现线程间的通信。

参考示例：

**public** **class** **MyObject** {

    **synchronized** **public** **void** **methodA**() {  
        *//do something....*  
    }  
  
    **synchronized** **public** **void** **methodB**() {  
        *//do some other thing*  
    }  
}  
  
**public** **class** **ThreadA** **extends** **Thread** {  
  
    **private** MyObject object;  
*//省略构造方法*  
    **@Override**  
    **public** **void** **run**() {  
        **super**.run();  
        object.methodA();  
    }  
}  
  
**public** **class** **ThreadB** **extends** **Thread** {  
  
    **private** MyObject object;  
*//省略构造方法*  
    **@Override**  
    **public** **void** **run**() {  
        **super**.run();  
        object.methodB();  
    }  
}  
  
**public** **class** **Run** {  
    **public** **static** **void** **main**(String[] args) {  
        MyObject object = **new** MyObject();  
  
        *//线程A与线程B 持有的是同一个对象:object*  
        ThreadA a = **new** ThreadA(object);  
        ThreadB b = **new** ThreadB(object);  
        a.start();  
        b.start();  
    }  
}

由于线程A和线程B持有同一个MyObject类的对象object，尽管这两个线程需要调用不同的方法，但是它们是同步执行的，比如：线程B需要等待线程A执行完了methodA()方法之后，它才能执行methodB()方法。这样，线程A和线程B就实现了 通信。

这种方式，本质上就是“共享内存”式的通信。多个线程需要访问同一个共享变量，谁拿到了锁（获得了访问权限），谁就可以执行。

#### 1.2 轮询

轮询的条件的可见性问题，关于内存可见性问题。可参考：Java多线程之volatile与synchronized的比较中的第一点“一、volatile关键字的可见性”。

线程都是先把变量读取到本地线程栈空间，然后再去修改的本地变量。因此，如果线程B每次都在取本地的条件变量，那么尽管另外一个线程已经变了轮询的条件，它也察觉不到，这样也会造成死循环。

#### 1.3 wait/notify机制

线程A要等待某个条件满足时(list.size()==5)，才执行操作。线程B则向list中添加元素，改变list 的size。

A,B之间如何通信的呢？也就是说，线程A如何知道 list.size() 已经为5了呢？

这里用到了Object类的 wait() 和 notify() 方法。

当条件未满足时(list.size() !=5)，线程A调用wait() 放弃CPU，并进入阻塞状态。---不像②while轮询那样占用CPU

当条件满足时，线程B调用 notify()通知 线程A，所谓通知线程A，就是唤醒线程A，并让它进入可运行状态。

这种方式的一个好处就是CPU的利用率提高了。

但是也有一些缺点：比如，线程B先执行，一下子添加了5个元素并调用了notify()发送了通知，而此时线程A还执行；当线程A执行并调用wait()时，那它永远就不可能被唤醒了。因为，线程B已经发了通知了，以后不再发通知了。这说明：通知过早，会打乱程序的执行逻辑。

#### 1.4 管道通信

就是使用java.io.PipedInputStream 和 java.io.PipedOutputStream进行通信

具体就不介绍了。分布式系统中说的两种通信机制：共享内存机制和消息通信机制。感觉前面的①中的synchronized关键字和②中的while轮询 “属于” 共享内存机制，由于是轮询的条件使用了volatile关键字修饰时，这就表示它们通过判断这个“共享的条件变量“是否改变了，来实现进程间的交流。

而管道通信，更像消息传递机制，也就是说：通过管道，将一个线程中的消息发送给另一个。