```java
public class BasicForJava extends Thread{


    boolean flag = false;

    int i = 0;

    public void run() {
        while(!flag) {
            System.out.println(1);
        }
    }

    public static void main(String [] args) throws InterruptedException {
        BasicForJava vt = new BasicForJava();
        vt.start();

        Thread.sleep(2000);

        vt.flag = true;

        System.out.println("stop" );
    }

}
```

上述代码可能能使线程中断，也可能使线程无法中断。每个线程在运行过程中都有自己的工作内存，那么线程1,在运行过程中，会将flag变量的值拷贝一份放在自己的内存中。那么当线程2更改了flag变量之后，但是还没来得及写入主存当中。由于线程1读取自己的内存，不知道线程2已经把flag进行改变了。

解释： 在jvm中，当线程访问一个对象值的时候，首先通过对象的引用栈找到对应在堆内存（称之为主内存）的变量的值，然后把堆内存变量的具体值load到线程本地内存中，建立一个变量副本，之后线程就不再和对象在堆内存变量值有任何关系了，而是直接修改副本变量的值，在修改完毕之后的某一时刻，自动把线程变量副本的值写回到堆中变量，这样在堆中的对象的值就产生变化了。

```java
public class VolatileTest extends Thread {

    volatile boolean flag = false;
    int i = 0;

    public void run() {
        while (!flag) {
            i++;
        }
    }

    public static void main(String[] args) throws Exception {
        VolatileTest vt = new VolatileTest();
        vt.start();
        Thread.sleep(2000);
        vt.flag = true;
        System.out.println("stope" + vt.i);
    }
}
```

在flag前面加上volatile关键字，强制线程每次读取该值的时候都去“主内存”中取值。在试试我们的程序吧，已经正常退出了。