优雅的结束线程

interrupt和InterruptedException

结束线程的方式

- ~~thread.stop()~~
- thread.setDaemo
- 共享变量
- 利用中断机制 thr

```
public clas

    private

    public v
        shoul
    }

    public v
        while

        }
    }

}
```

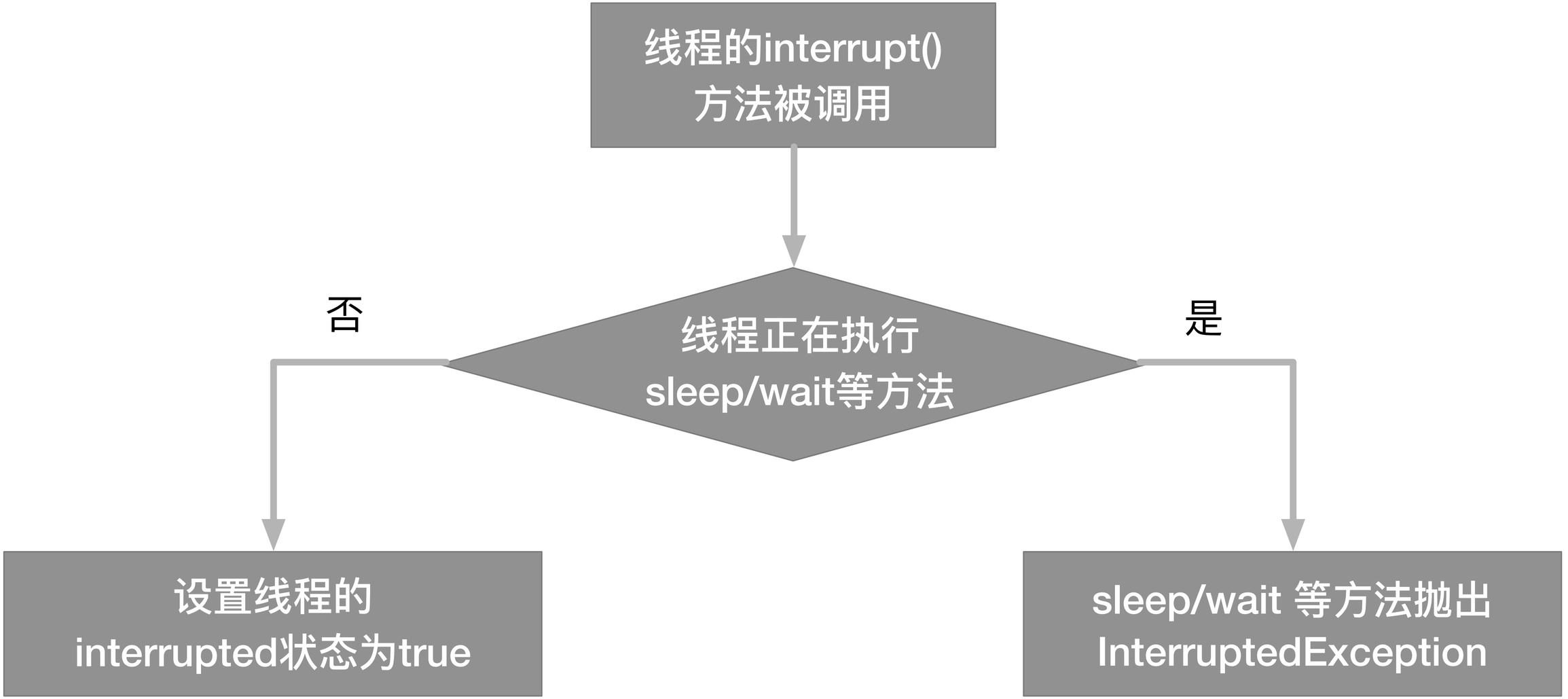
```
Thread t = new Thread(() -> {
    while(true){
        // do somthing
    }
}, “T-1");

t.setDaemon(true);

t.start();
```

中断相关的3个方法

- thread.interrupt()　　向某个线程发出中断信号
- boolean isInterrupted()　　查询某个线程是否有获得中断信号
- static boolean interrupted()　查询当前线程是否获得中断信号，查询后会重置中断的信号的状态；


线程的interrupt()
方法被调用
线程正在执行
sleep/wait等方法
否
是
设置线程的
interrupted状态为true
sleep/wait 等方法抛出
InterruptedException
可以通过 isInterrupted() 判断中断状态
可以catch(InterruptedException e)获得通知

被吞掉了的InterruptedException

```
try {
    Thread.sleep(100);
}catch(InterruptedException e){
    // 啥也不做，或者仅仅记录个日志
}
```

较优雅的处理方式

```
public void run(){
    while(!Thread.currentThread().isInterrupted()){
        //查询并处理

        try{
            Thread.sleep(100);
        }catch (InterruptedException e){
            //退出循环
            break;
        }
    }
    // 关闭打开的资源，退出线程
}
```