- 创建状态——(启动线程)——>就绪状态
  - 实例化Thread对象
- 就绪状态——(获得CPU资源)——>运行状态
  - 调度
- 运行状态——(释放CPU资源)——>就绪状态
  - 调度
- 运行状态——(线程自然执行完毕、外部干涉终止线程)——>死亡状态
  - 中断或结束
- 运行状态——(等待用户输入、线程休眠等)——>阻塞状态
  - 调用sleep方法、wait方法或同步锁定后
- 阻塞状态——(阻塞解除)——>就绪状态
  - 调用start方法
- 线程状态观测：Thread.State枚举
  - NEW：尚未启动的线程状态，未调用start方法的线程
  - RUNNABLE：在虚拟机中执行的线程状态，包含了就绪状态
  - BLOCKED：被阻塞等待监视器锁定的线程状态
  - WAITING：正在等待另一线程执行指定动作的线程状态
  - TIMED_WAITING：正在等待另一线程执行到指定时间的线程状态，Thread.sleep()的线程处于此状态
  - TERMINATED：已退出的线程状态
    - 结束之后的线程不能再start开启，会报错IllegalThreadStatException

- Thread类方法
  - voidsetPriority(int x) 设置线程优先级
  - static void sleep(long ms) 线程休眠指定毫秒
    - 1000ms=1s
    - sleep结束后进入就绪状态
    - 可以模拟网络延时(放大问题的发生性，看不出线程并发问题可以延时一下)、倒计时

        - 每个对象都有一把锁，sleep不会释放锁
    - void join() 合并线程
        - 此线程插队立即执行
        - 其它线程阻塞，等待插队线程执行完毕
    - static void yield() 礼让
        - 暂停当前执行的线程，先执行其他线程
        - 让CPU重新调度，礼让不一定成功，好比坐公交，有老人们上来了，你站起来让座，然后跟老人一起抢位置，可能还是你抢到了
    - void interrupt() 中断线程
        - 别用这个方式，同样不推荐stop和destroy方法
        - 建议让线程自己停止，或者定义一个标志位终止变量，当flag=false时跳出while循环终止线程运行
    - boolean isAlive() 判断该线程是否处于活动状态