java多线程学习(2021.03.06~~2021.03.20)

一.创建线程的方法

```
1.继承Thread类
->子类继承Thread类具备多线程能力
->启动线程：子类对象.start()
->!注意:不建议使用(避免OOP单继承局限性)

2.实现Runnable接口
->实现Runnable接口具备多线程的能力
->启动线程：传入目标对象+Thread对象.start()
->!注意：推荐使用(避免OOP单继承局限性，灵活方便，且方便一个对象被多个线程使用)

3.实现Callable接口
->实现Callable接口,需要返回值类型
->重写call方法，需要抛出异常
->创建目标对象
->创建执行服务：ExecutorService ser = Executors.newFixedThreadPool(1);
->提交执行:Future<Boolean> result1 = ser.submit(t1);
->获取结果:boolean r1 = result1.get();
->关闭服务:ser.shutdownNow();
```

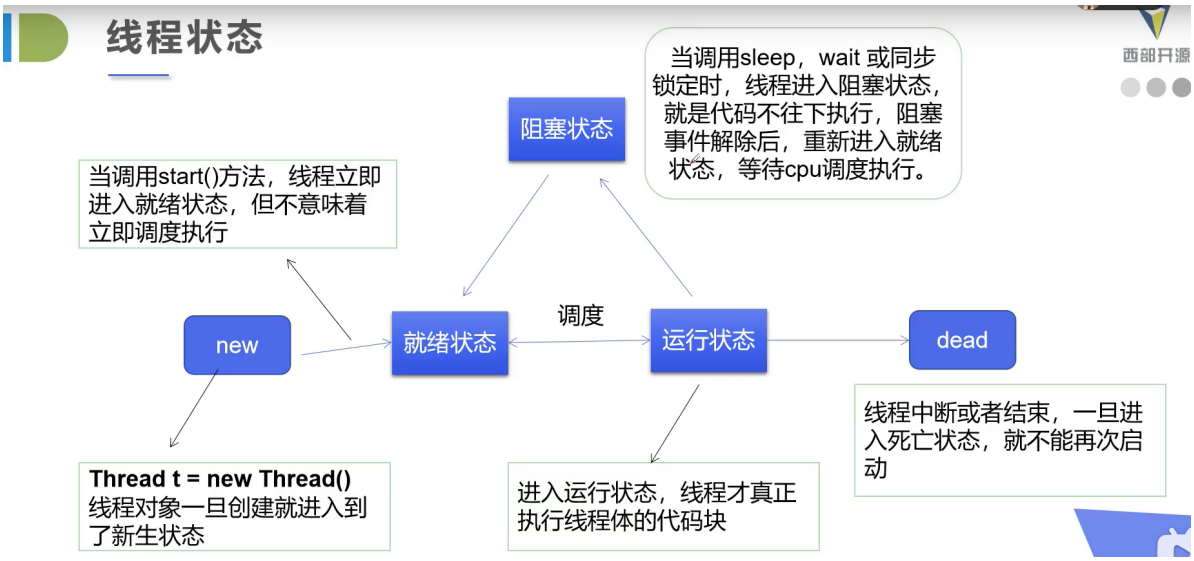
二.其他

```
1.函数式接口
->任何接口，只包含唯一一个抽象方法
->可以通过lambda表达式创建该接口的对象。
    eg:
        public interface Runnable{
            public abstract void run();
        }
2.lambda表达式
->表达式只有一行代码时才能简化成一行，多行代码时要加上花括号。
->对应接口必须是函数式接口
```

```
->可以不带参数，可以带1个或多个参数。

3.静态代理
->1.真实对象和代理对象都要实现同一个接口
->2.代理对象代理真实对象的事
优点：
 1.代理对象可以实现真实对象更多的方法
 2.使真实对象专心做一件事

```



3.线程停止

```
1.不建议死循环
2.建议使用标志位->>if(flag)run();
3.不使用stop或destory等已过时的或JDK不建议使用的方法
```

4.线程休眠（Thread.sleep()）

```
1.Thread.stop(XX)：设置当前线程阻塞的毫秒数。
2.sleep存在异常InterruptedException.
3.sleep时间到达后线程进入就绪状态。
4.sleep可以模拟网络延迟、倒计时等。
5.每个对象都有一个锁，sleep不会释放锁。
```

5.线程礼让(Thread.yield())

```
1.礼让线程，即：当前正在执行的线程暂停，但不阻塞。
2.将线程从运行状态 转变为 就绪状态。
3.让CPU重现调度，单礼让不一定成功！！看CPU心情。
```

6.线程插队，强制执行(Thread.join())

```
1.直接插队，其他线程等待。
```

7.线程状态(Thread.State)

```
1.New
    尚未启动的线程处于new状态
2.RUNNABLE
    在java虚拟机中处于执行状态
3.BLOCKED
    被阻塞等待监视器锁定
4.WAITING
    正在等待另一个线程执行
5.TIMED_WAITING
    等待另一个线程执行到指定时间
6.TERMINATED
    线程已退出
```

8.线程优先级

```
1.java提供了一个线程优先级调度起来监控启动后就绪状态的所有线程。
2.线程优先级用数字代替
    Thread.MIN_PRORITY = 1;
    Thread.MAX_PRORITY = 10;
    Thread.NORM_PRORITY = 5;
3.获取设置
    getPriority,setPriority(int xx)
4.！！！注意:优先级低只是被CPU调度的概率低。不是必然的。
```

9.守护线程(daemon)

```
1.线程分为：用户线程、守护线程
2.虚拟机必须确保用户线程执行完毕、但不用等待守护线程完毕。
3.如：后台记录操作日志、内存监控、垃圾回收等....
4.设置守护线程：Thread.setDaemon(true) //默认是false。
```

10.线程同步

```
1.并发：多个线程同事操作同一个对象
2.队列和锁
3.synchronized是Java中的关键字，格式:synchronized(Obj){},
  是一种同步锁：
    1. 修饰一个代码块，被修饰的代码块称为同步语句块，其作用的范围是大括号{}括起来的代码，作用的对象是调用这个代码块的对象；
    2. 修饰一个方法，被修饰的方法称为同步方法，其作用的范围是整个方法，作用的 对象是调用这个方法的对象；
    3. 修改一个静态的方法，其作用的范围是整个静态方法，作用的对象是这个类的所 有对象；
    4. 修改一个类，其作用的范围是synchronized后面括号括起来的部分，作用主的对象是这个类的所有对象。

```

11.死锁

```
1.死锁：多个线程同时被阻塞，它们中的一个或者全部都在等待某个资源被释放。由于线程被无限期地阻塞，因此程序不可能正常终止。
2.先知道产生条件，再避免：
    1.互斥使用。当资源被一个线程占用时，其他资源不能使用该资源。
    2.不可抢占。资源请求者不能从占有者手中夺取，资源只能由占有者
                主动释放。
    3.请求和保持。当资源请求者请求其他资源时，对已有资源保持不放。
    4.循环等待。存在一个等待队列：P1占有P2的资源，P2占有P3的资源，P3占有P1的资源。这样就形成了一个环路。程序卡死。
```

12.Lock（可重入锁）

```

```

13.线程通信

```
1.Thread.wait():线程等待，直到被唤醒，期间会释放锁。(sleep掐尖不会释放锁)。
2.Thread.wait(long timeout):指等待的毫秒数。
3.Thread.notify():唤醒一个处于等待状态的线程。
4.Thread.notifyAll():唤醒一个对象上的所有在wait的线程，优先级高的优先调度。
```

14.生产者消费者模型

```
1.-->管程法
    生产者：负责生产数据的模块(方法，对象，线程，进程等)。
    消费者：负责处理数据的模块(方法，对象，线程，进程等)。
    缓冲区：相当于使者，生产者将生产好的数据放入缓冲区，消费者从缓冲区拿出数据。
2.-->信号灯法
```

15.线程池

```
1.思路：
    提前创建线程池，然后吧多个线程放进去，使用时直接获取，是用完再放回池中。避免频繁创建销毁、实现重复利用。类似于现实中的交通工具。
2.优点：
    1.提高效率(较少创建线程的时间)
    2.降低消耗(重复利用)
    3.便于线程管理()
        corePoolSize():线程池的大小
        maximumPoolSize:最大线程数
        keepAliveTime:线程闲置时间
3.JDK提供了线程池相关API：ExecutorService 和 Executors
    1.ExecutorService：真正的线程池接口，常见子类:TheradPoolExecutor.
        1.void execute(Runnable command):执行任务/命令，没有返回值。
        2.void shutdown():关闭连接池。
    2.Executors:工具类、线程池的工厂类，英语创建冰返回不同类型的线程池。

```