# 创建线程

## 三种方式

### 创建

继承Thread：线程代码存放Thread子类run方法中。

实现Runnable：线程代码存在接口的子类的run方法。

实现Callable接口

创建线程池

### 区别

源码方面的区别：

继承Thread : 由于子类重写了Thread类的run(), 当调用start()时, 直接找子类的run()方法

实现Runnable : 构造函数中传入了Runnable的引用, 成员变量记住了它, start()调用run()方法时内部判断成员变量Runnable的引用是否为空, 不为空编译时看的是Runnable的run(),运行时执行的是子类的run()方法

### 优缺点

继承Thread

好处是:可以直接使用Thread类中的方法,代码简单

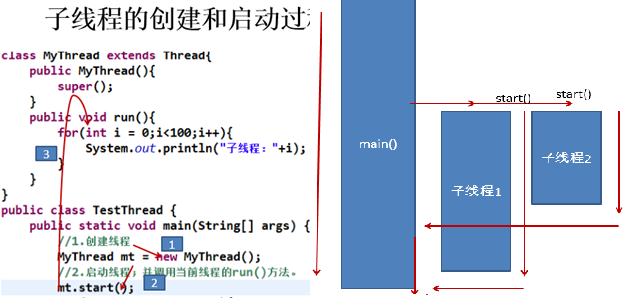
弊端是:如果已经有了父类,就不能用这种方法

实现Runnable接口

好处是:即使自己定义的线程类有了父类也没关系,因为有了父类也可以实现接口,而且接口是可以多实现的

弊端是:不能直接使用Thread中的方法需要先获取到线程对象后,才能得到Thread的方法,代码复杂

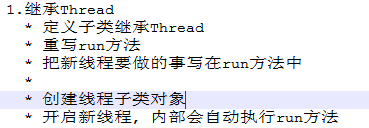
### 子线程创建过程

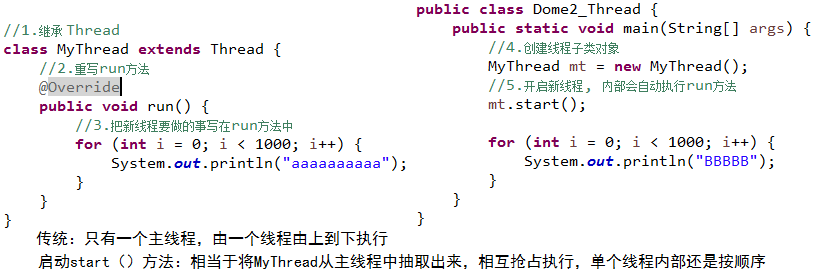


结果：两个线程相互抢占资源输出

## 实现方式（Thread和Runnable）

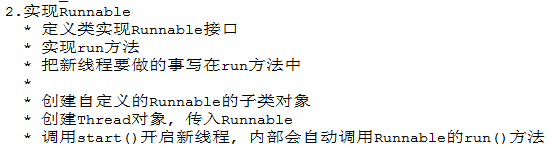
### 继承Thread

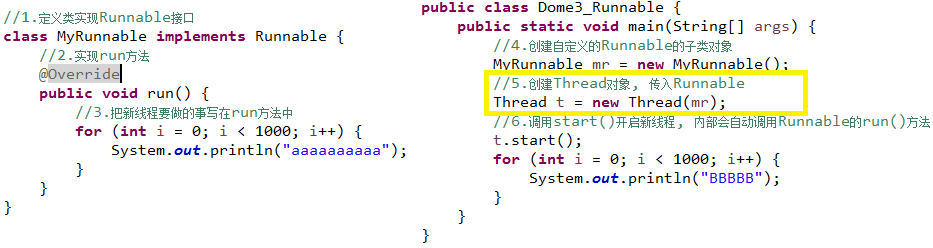




### 实现Runnable接口

需要thread 去实现它



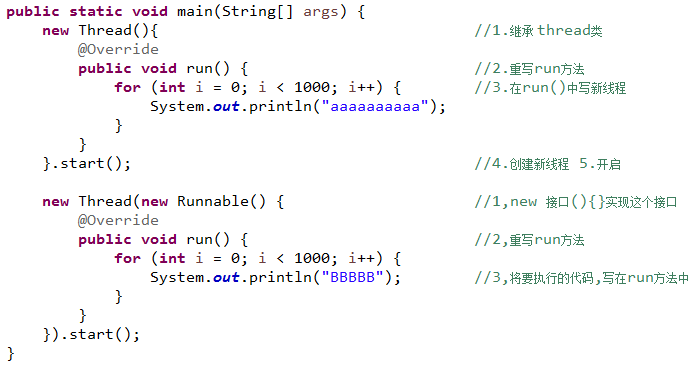


### 实现Callable接口

使用继承方式的好处是方便传参，你可以在子类里面添加成员变量，通过set方法设置参数或者通过构造函数进行传递，而如果使用Runnable方式，则只能使用主线程里面被声明为final的变量。

不好的地方是Java不支持多继承，如果继承了Thread类，那么子类不能再继承其他类，而Runable则没有这个限制。前两种方式都没办法拿到任务的返回结果，但是Callable方式可以

### 匿名内部类实现方式



# 原理

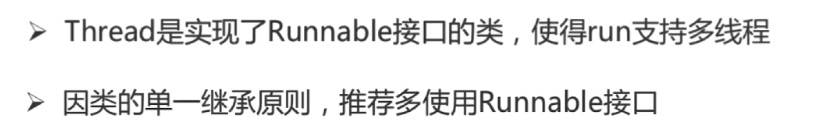
## Thread和runnbale 接口的关系

Thread类也是实现Runnable接口

1,看Thread类的构造函数,传递了Runnable接口的引用

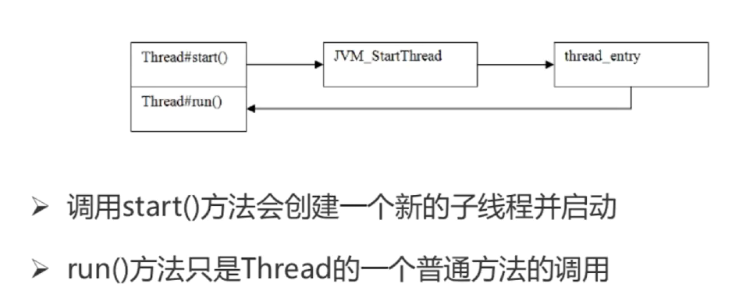
2,通过init()方法找到传递的target给成员变量的target赋值

3,查看run方法,发现run方法中有判断,如果target不为null就会调用Runnable接口子类对象的run方法



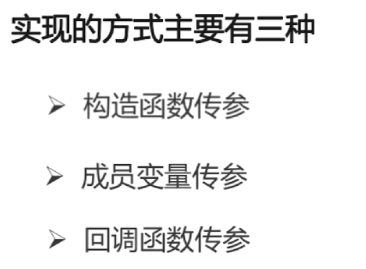
## Start、run不能直接调run（）方法

调用相当与直接的对象调方法，如果想要并发执行操作，实现线程抢占，就调用start（）才是启动线程

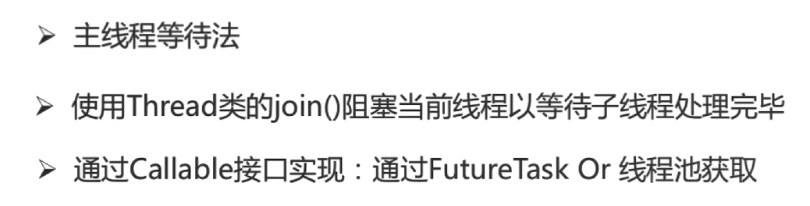


## 如何给run()方法传参

https://blog.csdn.net/saycheesenn/article/details/52956331

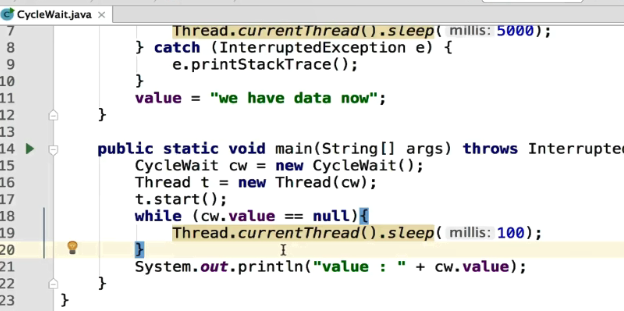


## 如何处理线程返回值



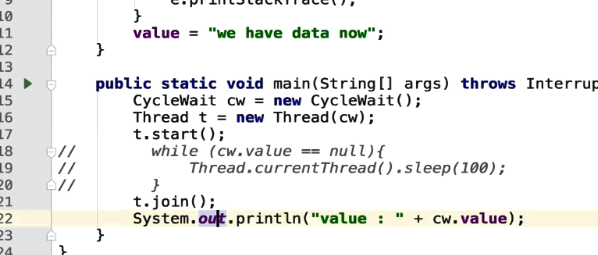
### 主线程等待法

在没有得到返回值，无限循环去暂停主线程



### Join阻塞当前线程

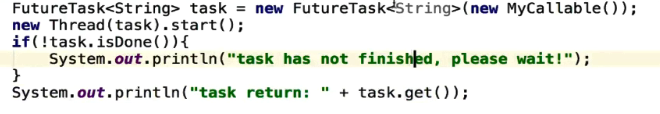




### Callable 接口



#### FutureTask



#### 线程池获取

