创建线程有两种方法：

第一种方式：

1.继承Thread
1.1定义一个类继承Thread;
1.2重写run方法
1.3创建子类对象，就是创建线程对象
1.4调用start方法，开启线程并让线程执行，同时还会告诉jvm去调用run方法

```
public class DemoThread extends Thread {
    public void run(){
        for(int i=0;i<10;i++){
        System.out.println(Thread.currentThread().getName());
        }
    }
    public static void main(String[] args) {
        DemoThread d1 = new DemoThread();
        d1.start();
        DemoThread d2 = new DemoThread();
        d2.start();
    }
}
```

为什么要这么做？

继承Thread类，因为Thread类描述线程事务，具备线程该有功能

那为什么不直接创建Thread类的对象？

Thread t1 = new Thread(); t1.start();//这么做没错，但是该start调用的是Thread类中的run方法
而这个run方法没有做什么事情，更重要的是这个run方法中并没有定义我们需要让线程执行的代码。

创建线程的目的是什么？

是为了建立单独的执行路径，让多部分代码实现同时执行
也就是说线程创建并执行需要给定的代码（线程的任务）
主线程它的任务定义在main函数中
自定义线程需要执行的任务定义在run方法中
Thread类中的run方法内部的任务并不是我们所需要的，只要重写这个run方法。
既然Thread类已经定义了线程任务的位置，只要在位置中定义任务代码即可，所以重写run方法。

多线程执行时，在栈内存中，其实每一个执行线程都有一片自己所属的栈内存空间进行方法的压栈和出栈，当执行线程的任务结束了，线程自动在栈内存中释放了，当时所有的执行线程都结束了，晋城就结束了。

线程对象调用run方法和调用start方法的区别？

调用run方法不开启线程，仅是对象调用方法
调用start方法开启线程，并让jvm调用run方法在开启的线程中执行

获取线程名称：

Thread:currentThread()获取当前线程对象，获取名称：getName()
Thread.currentThread().getName();
主线程的名称：main
自定义线程的名称：Thread-1 线程多个时，数字顺延，Thread-2,Thread-3 ....

第二种方式 ：

1.实现Runnable接口
1.1定义实现Runnable接口 1.2覆盖接口中的run方法，将线程任务定义到run()方法中
1.3创建Thread类的对象 ：只有创建Thread类的 对象才可以创建线程 1.4将Runnable接口的子类对象作为参数传递给Thread类的构造函数
因为线程已被封装到Runnable接口的run()方法中，而这个run方法所属于Runnable接口的z子类对象，所以将这个子类对象作为参数传递给Thread的构造函数，这样，线程对象创建时就可以明确要运行的线程任务。
1.5调用Thread类的start方法开启线程

```
public class DemoRunnable implements  Runnable {
    @Override
    public void run() {
        for(int i=0;i<10;i++){
            System.out.println(Thread.currentThread().getName());
        }
    }

    public static void main(String[] args) {
        DemoRunnable d1 = new DemoRunnable();
        DemoRunnable d2 = new DemoRunnable();
        Thread t1 = new Thread(d1);
        Thread t2 = new Thread(d2);
        t1.start();
        t2.start();
    }
}
```

通过源码的形式讲解一下将runnable接口的子类对象作为参数传递给Thread构造函数的原因。

```
class Thread{
    private Runnable target;
   public  Thread(Runnable target){
        this.target = target;
    }
    public void run(){
       if(target !=null){
           target.run();
```

```
        }
    }
    public void start(){
        run();
    }
}
```

两者的区别：
1.实现Runnable避免了单继承的局限性，较为常用
2.实现Runnable接口的实现，更加符合面向对象，线程分为两部分，一部分是线程对象，一部分是线程任务。继承Thread，线程对象和线程任务耦合在一起，一旦创建Thread类的子类对象，即是线程对象，又有线程任务;实现runnable接口，将线程任务单独分离出来封装成对象，类型就是Runnable接口类型。Runable接口对线程对象和线程任务进行解耦。

线程任务中通常都有循环结构！