何时需多线程：
程序需同时执行多个任务
程序需要实现一些需等待的任务
后台运行的程序时

创建线程的两种方式：
① 继承Thread类：
定义子类继承Thread类
子类中重写Thread类中的run
创建Thread子类对象，即创建了线程
调用线程对象start方法：启动线程，调用run方法
② Runnable接口
定义类，实现Runnable接口
类中重写run方法
通过Thread类含参构造器创建线程对象
将Runnable接口对象作为参数传给Thread构造方法
调用Thread的start方法，开启线程，调用Runnable子类接口的run方法

```
package com.test.thread;

public class TestThread extends Thread {  // 继承thread
    @Override
    public void run() {  // 重写run方法
        System.out.println("多线程运行的代码");
        for (int i = 0; i < 5; i++) {

            System.out.println("这是多线程的代码运行时间" + i);
        }
    }
}
```

```
package com.test.thread;

public class TestRunnable implements Runnable{
    @Override
    public void run() {  // 重写run方法
        System.out.println(Thread.currentThread().getName()+"多线程运行的代码");
        for (int i = 0; i < 5; i++) {
            System.out.println("这是多线程的代码runnable" + i);
        }
    }
}
```

```
package com.test.thread;



public class Test {
    public static void main(String[] args) {
```

```java
        Thread t = new TestThread();  // 创建了线程对象，启动了线程
        Thread t1 = new Thread(new TestRunnable());
        Thread t2 = new Thread(new TestRunnable(), "t-2");  // 第二个参数为线程名
        t.start();
        t1.start();
        t2.start();
        System.out.println("-----------------------");
        System.out.println("-----------------------");
        System.out.println("-----------------------");
        System.out.println("-----------------------");

    }
}
```

实现run方法比重写run方法更好，可以new一个Runnable对象，创建多个线程，就可以实现对同一资源的多线程操作。

Thread类相关方法：都是通过Thread对象调用的

getName()　　Thread.currentThread() 获取当前线程，可在run中使用

setName()

设置线程的优先级：

getPriority();

setPriority();

线程让步：暂停当前正在执行的进程，把执行机会让给相同或更高级线程

static void yield(); //在run中加入

join()：将run方法插入代码中，线程阻塞（更像函数调用），要start后join

sleep() 睡眠

stop() 强制停止线程

isAlive()：判断线程是否存活

线程的生命周期：

新建　Thread类对象被声明并创建

就绪　start之后，线程加入CPU时间片

运行　就绪线程获取CPU

阻塞

死亡　完成工作或被强制中止

线程的同步和死锁：

在多个线程调用同一资源时，需对（访问临界资源的）方法加上 synchronized 同步锁

public synchronized void drawing(int m);　锁是锁了当前对象，即使static是所有对象共用的，还是锁不住

若静态方法加锁，锁的是类

使用代码块加锁，在方法内部

public void drawing(　　){

synchronized (this) {

}

}

每个方法内部有 synchronized(this), 锁当前对象

synchronized (Object obj); 锁传入的对象