Java 写程序三部分组成：

1、JDK 系统类库

JRE：Java Runtime Enviroment（Java 运行环境），仅供运行程序的。

JDK：Java Development Kit（Java 开发工具包），如果需要进行程序开发，必须安装 JDK。

String、Scanner、包装类。。。

java.lang.Thread

javax.servlet.Servlet

2、第三方类库

非 Java 官方的组织提供的一些成熟好用的工具，C3P0 数据库连接池、Spring 框架、DBUtils、Dom4J...

github：全球最大的同性交友网站

3、开发者自定义的代码

根据具体的业务需求编写的业务代码。

Java 中线程的使用

- 继承 Thread 类

1、创建自定义类并继承 Thread 类。

2、重写 Thread 类中的 run 方法，并编写该线程的业务逻辑代码。

```
package com.southwind.test;

public class MyThread extends Thread {

  @Override
  public void run() {
    // TODO Auto-generated method stub
    //定义业务逻辑
    for(int i = 0;i<10;i++) {
      System.out.println("-------------MyThread");
    }
  }

}
```

3、使用。

```
package com.southwind.test;

public class Test {
  public static void main(String[] args) {
    //开启两个子线程
    MyThread thread1 = new MyThread();
    MyThread2 thread2 = new MyThread2();
    thread1.start();
    thread2.start();
  }
}
```

注意：不能通过 run 方法来调用线程的任务，因为 run 方法调用相当于普通对象的执行，并不会去抢占 CPU 资源。

只有通过 start 方法才能开启线程，进而去抢占 CPU 资源，当某个线程抢占到 CPU 资源后，会自动调用 run 方法。

- 实现 Runnable 接口

1、创建自定义类并实现 Runnable 接口。

2、实现 run 方法，编写该线程的业务逻辑代码。

```
package com.southwind.test;

public class MyRunnable implements Runnable {

  @Override
  public void run() {
    // TODO Auto-generated method stub
    for(int i=0;i<1000;i++) {
      System.out.println("========MyRunnable=======");
    }
  }

}
```

3、使用。

```
MyRunnable runnable = new MyRunnable();
Thread thread = new Thread(runnable);
thread.start();
MyRunnable2 runnable2 = new MyRunnable2();
Thread thread2 = new Thread(runnable2);
thread2.start();
```

线程和任务：

线程是去抢占 CPU 资源的，任务是具体执行业务逻辑的，线程内部会包含一个任务，线程启动(start)，当抢占到资源之后，任务就开始执行(run)。

两种方式的区别：

1、MyThread，继承 Thread 类的方式，直接在类中重写 run 方法，使用的时候，直接实例化 MyThread，start 即可，因为 Thread 内部存在 Runnable。

2、MyRunnbale，实现 Runnable 接口的方法，在实现类中重写 run 方法，使用的时候，需要先创建 Thread 对象，并将 MyRunnable 注入到 Thread 中，Thread.start。

实际开发中推荐使用第二种方式。

在线画图软件：

https://www.processon.com/diagrams

线程的状态

线程共有 5 种状态，在特定的情况下，线程可以在不同的状态之间切换，5 种状态如下所示。

- 创建状态：实例化一个新的线程对象，还未启动。
- 就绪状态：创建好的线程对象调用 start 方法完成启动，进入线程池等待抢占 CPU 资源。
- 运行状态：线程对象获取了 CPU 资源，在一定的时间内执行任务。
- 阻塞状态：正在运行的线程暂停执行任务，释放所占用的 CPU 资源，**并在解除阻塞状态之后也不能直接回到运行状态，而是重新回到就绪状态，等待获取 CPU 资源。**
- 终止状态：线程运行完毕或因为异常导致该线程终止运行。

线程状态之间的转换图。