进程：

正在运行的程序，资源分配的最小单元

线程：

进程中的一个执行单元，线程是CPU调度的最小单元

一个进程中至少要有一个线程，这个线程叫主线程：还可以添加新的线程，这个线程叫子线程

主线程：

currentThread方法可以获得主线程

！！！

在main中需要在运行方法前，开启子线程，否则默认为使用同一个线程

创建线程：

方法一：Thread子类

1:继承thread类

2:重写run方法

3:在run方法内写准备在子线程中执行的代码

4.创建该子类的对象，并调用start方法

方法二：实现Runnable接口

1.创建实现Runnable接口类

2.重写run方法

3.在run方法内写准备在子线程中执行的代码

4.创建该接口类的对象

5.使用Thread的有参构造方法，把对象传进去

6.Thread对象调用start方法

方法三：线程池

缓存线程池：

```
ExecutorService executorService1 = Executors.newCachedThreadPool();
//手写版
```

```
ExecutorService executorService=new ThreadPoolExecutor(0, Integer.MAX_VALUE,
 60L, TimeUnit.SECONDS,
 new SynchronousQueue<Runnable>());
```

固定池：

```
ExecutorService executorService = Executors.newFixedThreadPool(线程个数);
```

线程池中执行：

```
 Runnable runnable1 = new PoolRunnable("李四");
 executorService.execute(runnable1);
```

资源抢夺：

当多个线程使用相同的数据时，会出现资源抢夺

解决方案：对共享数据的处理，一个时刻，只能有一个线程在处理

生产者和消费者：

使用集合时需要注意：

1.使用线程安全的集合

2.当不使用安全线的集合时，要加锁

加锁方式： 使用：

Reentrantlock方法

```
ReentrantLock lock = new ReentrantLock();
lock.lock(); 上锁
look.unlock(); 解锁
```

synchronized方法

```
synchronized (arrayList){arrayList.wait(); notifyAll();}
```