一. 进程:正在运行的程序,进程是资源分配的最小单元

线程:是进程中执行单元,线程是cup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,这些线程叫子线程;

```
!!!主线程
Thread mainThread = Thread.currentThread();
System.out.println(mainThread);
```

二. 创建线程

1. Thread子类
2. 实现Runnable接口
3. 线程池

```
MyThread myThread=new MyThread();
```

在当前线程执行,并没有开启新的线程.

```
myThread.run();
```

开始子线程. 并调用run方法

```
myThread.start();
```

三. 线程池

缓存(冲)线程池

```
ExecutorService executorService= Executors.newCachedThreadPool();
```

固定线程池

```
ExecutorService executorService = Executors.newFixedThreadPool(1);
```

向线程池中添加执行代码

```
for (int i = 1; i < 20; i++) {
Thread.sleep(10);
Runnable runnable=new PoolRunnable("张三"+i);
executorService.execute(runnable);
}
/*Runnable runnable=new PoolRunnable("张三");
executorService.execute(runnable);*/

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

```

抢票

```
SaleTicket saleTicket=new SaleTicket();
Thread thread1=new Thread(saleTicket,"学生窗口");
Thread thread2=new Thread(saleTicket,"军人窗口");
Thread thread3=new Thread(saleTicket,"普通窗口");

thread1.start();
thread2.start();
thread3.start();
```

四.方式一:1.创建实现了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MyRunnable对象

5.使用Thread的有参构造方法,把MyRunnable对象传过去.

6.Thread对象调用start方法

方式二: 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MyThead对象,并调用start方法

锁:

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

```
ReentrantLock lock = new ReentrantLock();
```

仓库的最大存储量

public static final int *MAX_SIZE* = 100;

仓库的载体

方式1,线程安全的类

```
private Vector arrayList=new Vector();
```

方式2:加锁

```
private ArrayList arrayList = new ArrayList();
```

```
存
public void add(int count) {
synchronized (arrayList) {
while (count > MAX_SIZE - arrayList.size()) {
System.out.printf("要添加%d个货物,当前空间不足为%d,空间不足!\n", count, MAX_SIZE - arrayList.size());
try {
arrayList.wait();
} catch (InterruptedException e) {
e.printStackTrace();
 }
 }
 for (int i = 0; i < count; i++) {
 arrayList.add(new Object());
 }
 System.out.printf("要添加%d个货物,添加成功,当前空间为%d\n", count, MAX_SIZE - arrayList.size());
 arrayList.notifyAll();
 }
 }
```

```

//取
public void minus(int count) {
synchronized (arrayList) {
while (count > arrayList.size()) {
System.out.printf("要取%d个货物,当前货物为%d,货物不足!\n", count, arrayList.size());
```

```java
try {
arrayList.wait();
} catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
 for (int i = 0; i < count; i++) {
 arrayList.remove(0);
 }
 System.out.printf("要取%d个货物,取货成功,当前空间不足为%d\n", count, arrayList.size());
 arrayList.notifyAll();
 }
 }
```