复习：

1. 合并流---逻辑串联多个流，将这多个流合并成一个流进行读取操作。

2. 序列化/反序列化流---将对象进行完整存储---接口Serializable，关键字static/transient，版本号seriaVersionUID,集合

3. Properties---可持久化的映射

4. 可变参数---本质上是一个数组，只能定义一个并且必须在参数列表的尾部

5. 枚举---enum

## 线程

进程---计算机在执行的任务或者逻辑---服务（没有界面的进程）

线程---进程中任务的一个小任务---QQ，下载软件，JVM

记事本是单进程多线程

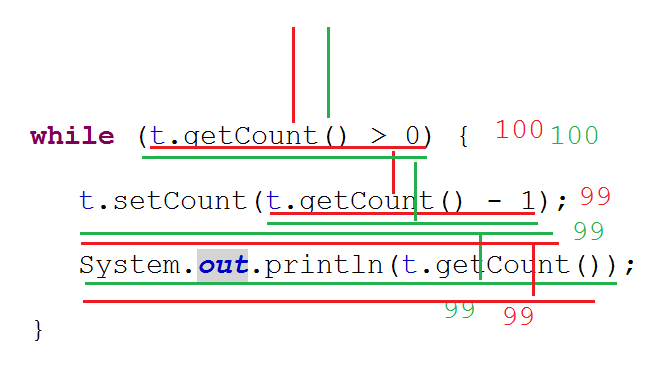
### 自定义线程

1. 继承 Thread类，重写其中的run方法，将线程要执行的逻辑写到run方法。通过线程对象身上的start方法来启动线程

2. 实现Runnable接口，重写的run方法。通过Thread对象来启动这个线程

3. 实现Callable<T>接口，重写的call方法。---现阶段要求知道即可

线程的执行并不是有序的，是相互抢占，而且抢占并不是只在线程执行的开始而是发生在线程执行的每一步过程中---由于多个线程之间相互抢占资源导致出现了不符合常理的情况，称之为多线程的并发安全问题



同步代码块---将可能出现问题的代码放到了synchronized中，需要一个锁对象---必须是所有线程都认识的资源---共享资源，类的字节码，this

同步方法---synchronized修饰方法---锁对象是this

死锁---产生的原因：多个线程，共享资源过多，锁对象不统一，锁的嵌套---避免死锁：统一锁对象，减少锁的嵌套

同步/异步---如果一个对象在某个时间段内只允许一个线程操作--同步

同步一定是安全的，不安全一定是异步的。

HashMap---异步式线程不安全

Hashtable---同步式线程安全

ConcurrentHashMap---异步式线程安全

通过等待唤醒机制调节线程之间的执行顺序---线程之间的相互通信

线程在等待期间是位于线程池中的。---线程池本质上是一个存储线程的队列

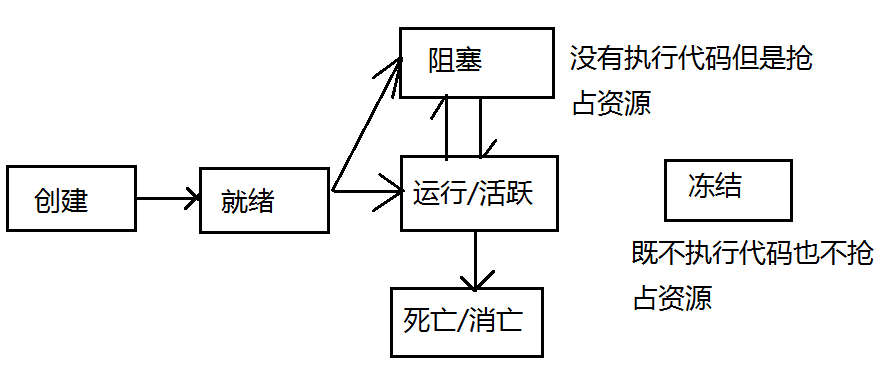
a1,a2,c1,c2() -> a1 running -> a1,a2,c1,c2() -> a1 running -> a2,c1,c2(a1) -> a2 running -> c1,c2(a1,a2) -> c1 running -> a1,c1,c2(a2) -> c1 running -> a1,c2(a2,c1) -> c2 running -> a1(a2,c1,c2) -> a1 running -> a1,a2(c1,c2) -> a1 running -> a2(c1,c2,a1) -> a2 running -> (c1,c2,a1,a2)

总结：sleep和wait有什么区别？

sleep方法需要指定睡眠时间，到点自然醒。释放执行权，不释放锁。被设计在Thread类上，是一个静态方法

wait方法可以指定等待时间也可以不指定，如果不指定时间需要唤醒。释放执行权，释放锁。被设计在了Object类上，是一个普通的方法、wait方法必须结合锁来使用。

### 线程的状态



### 守护线程

守护其他线程的执行。当被守护的线程结束之后，守护线程无论完成与否都会随之结束。

只要代码中出现了守护线程，要么这个线程是守护线程要么就是被守护的线程---如果出现了多个被守护的线程，那么以最后一个被守护的线程作为结束标志。

### 线程的优先级

线程有1-10这10个优先级。优先级越高，理论上线程抢到资源的概率越大。但是相邻两个优先级之间几乎看不出差别，至少相差5个优先级才会略有差别。

如果没有设置优先级，默认优先级为5

### 单例模式

保证全局过程中只存在一个唯一实例的模式

饿汉式相对懒汉式来说耗费内存。懒汉式相对饿汉式而言，存在线程安全问题。