**进程：**

1. 进程是程序的一次动态执行过程。它经历了从代码加载、执行到执行完毕的一个完 整过程。
2. 多进程操作系统能同时运行多个进程（程序），由于CPU速度非常快，使得所有程 序好像是在“同时”运行一样。

**线程：**

1. 线程是比进程更小的执行单位，是在进程的基础上进一步划分的，一个进程包含1~n 个线程。也就是多线程。
2. 多个线程可以同时存在、同时运行。
3. 线程是指程序的运行流程，多线程的机制是指可以同时运行多个程序块，使程序运 行的效率变得更高。

**线程和进程之间的关系：**

1. 一个进行有1个或多个线程组成。
2. 线程与进程都是实现并发控制的一种手段。

**Java中线程的实现：**

在Java中，实现多线程有两种方式，一是继承Thread类，另一种是实现Runnable接口。

**继承Thread类：**

Thread类是在java.lang包中定义的。一个类只要继承了Thread类，此类就称为线程 类。

1. 继承Thread后，必须覆写Thread类的run( )方法，此方法是线程的主体，也就 是线程要执行的方法。
2. 要想启动线程，必须调用从Thread类中继承而来的start( )方法
3. 如果一个类通过继承Thread类实现多线程，则每个线程对象只能调用一次start( ) 方法。否则会出现异常。

**实现Runnable接口：**

Runnable接口定义在java.lang包中的，一个类只要实现了Runnable接口，此类就称为 线程类。

1. 子类实现Runnable接口后，必须覆写Runnable接口中的run( )方法，此方法是线程 的主体，也就是线程要执行的操作。
2. 通过Runnable接口实现多线程，如果要想启动多线程，依然需要依靠Thread类中 的start( )方法。在Thread类中提供了public Thread(Runnable target)和public Thread(Runnable target,String name)两个构造方法可以接收Runnable子类的实例 对象。

**线程的运行原则：**

多个线程是交错运行的，哪个线程抢到了CPU资源，哪个线程就可以运行。但抢到CPU 的速度是极快的，所以多个线程看上去像同时运行一样。

可以理解为在Java中，所有的线程都是同时启动的，哪个线程先抢到了CPU资源，哪 个线程就先运行。

**Thread类和Runnable接口的区别：**

通过继承Thread类和实现Runnable接口都可以实现多线程，但两种方式是有区别的：

1. Thread类具有单继承的局限，实现Runnable接口能够避免单继承的局限。
2. Thread类是Runnable接口的子类，这两个类运用了类似于代理设计模式。
3. 如果一个类继承Thread类，则不适合用于多个线程共享资源，而通过实现Runnable 接口，可以方便的实现资源共享。

所以，在开发中建议使用Runnable接口实现多线程。

**线程的状态：**

要想实现多线程，必须在创建线程对象启动线程，每一个线程类对象表示一个线程。线 程一般具有5种状态，即创建、就绪、运行、阻塞、终止。

1. 创建状态：当调用线程类中的构造方法创建线程对象后，新的线程对象便处于创建 状态。此时，它已经具有了相应的内存空间和其他资源，但还处于不可 运行状态。
2. 就绪状态：线程对象调用start( )方法启动线程时，线程进入就绪转态。此时，线程 进入线程队列排队，等待CPU服务，具备了运行条件。
3. 运行状态：当就绪状态的线程被调用并获得处理器资源时，线程进入运行状态，自 动调用该线程对象的run( )方法。run( )方法中定义了该线程的操作和功 能。
4. 堵塞状态：一个正在执行的线程在某些特殊情况下（需要输入时，睡眠时）会被挂 起，此时线程进入阻塞状态。阻塞状态下的线程会让出CPU并暂时终止 自己的执行。只要当引起阻塞的原因被消除后，线程进入就绪状态。
5. 死亡状态：线程调用stop( )方法时或run( )方法执行结束后，线程处于死亡状态。死 亡状态下的线程不具有再运行的能力。

**线程操作的相关方法：**

虽然Thread类继承了Runnable接口，但是操作线程的主要方法都定义在Thread类中。

1. public Thread( Runnable target )：接收Runnable接口子类对象，并实例化Thread类 对象。
2. public Thread( Runnable target , String name )：接收Runnable接口子类对象，并实例 化Thread类对象，同时设置线程的名称。
3. public Thread( String name )：设置线程的名称，并实例化Thread类对象。

4）public static Thread currentThread( )：返回在正在执行的线程。

5）public final String getName( )：得到线程的名称。

6）public final int getPriority( )：得到线程的优先级。

7）public boolean isInterrupted( )：判断线程是否被中断，返回布尔值。

8）public boolean is Alive( )：判断线程是否启动或正在运行。

9）public final void join( ) throws InterruptedException：让线程死亡。

10）public final synchronized void join( long millis ) throws InterruptedException：等待millis 毫秒后，线程死亡。

11）public void run( )：执行线程。

12）public final void setName( String name )：设置线程的名称。

13）public final void setPiority( int newPriority )：设置线程的优先级。

14）public static void sleep( long millis ) throws InterruptedException：使当前线程休眠millis 毫秒。

15）public void start( )：启动线程。

16）public String toString( )：返回代表线程的字符串。

17）public static void yield( )：将目前正在执行的线程暂停，允许其他线程执行。

18）public final void setDaemon( boolean on )：将一个线程设置成后台运行。

**设置和取得线程的名称：**

线程对象可以调用Thread类中的setName( )设置线程的名称，调用getName( )取得线程 的名称。

如果没有设置线程的名称，系统会自动为线程分配一个名称，名称格式为Thread-Xxx。

**判断线程是否启动：**

可以调用Thread类中的isAlive( )方法来判断一个线程是否启动或正在运行，结果返回布 尔值。

**使线程强制运行：**

可以使用join( )方法让一个线程强制运行。在线程的强制运行期间，其他线程无法运行， 必须等待此线程完成之后才可以继续执行。

**运行Java程序启动的最少线程数：**

Java程序运行的时候也是以线程的方式运行的，对于一个Java程序来说，当一个Java 程序运行时至少会启动两个线程。

1. 主线程，即main( )方法。
2. 垃圾收集线程。

因为线程操作的不确定性，主线程有可能比其他线程最先执行完。

**线程的休眠：**

直接调用Thread.sleep( )的可以实现线程的休眠。

**中断线程：**

调用Thread类中interrupted( )方法可以中断线程的运行。

**线程的优先级：**

线程处于就绪状态时，哪个线程的优先级高，哪个线程就有可能会被先执行。

使用Thread类中的setPriority( )方法可以设置一个线程的优先级，使用getPriority( )方法 可以得到线程的优先级。在Java中共存在三种优先级：

1. putlic static final int MIN\_PRIORITY：最低优先级，表示的常量为1
2. public static final int NORM\_PRIORITY：中等优先级，也是线程默认的优先级，表示的 常量为5
3. public static final MAX\_PRIORITY：最高优先级，表示的常量为10

但并不是线程的优先级高就一定会先执行，哪个线程先执行将由CPU的调度决定。

**线程的礼让：**

在线程操作中，可以使用yield( )方法将一个线程的操作暂时让给其他线程执行。

**同步：**

通过Runnable接口实现多线程，则意味着该线程类中的属性可以被多个线程共享。但 多个线程共享属性时，可能会出现问题。此时就需要使用同步来解决问题。

同步指的是多个线程在操作同一属性时的时间段内只能有一个线程进行，其他线程要等 待此线程完成之后才可以继续执行。且不管该线程是否休眠，就算是休眠也必须等待此 线程休眠后并执行完。

**使用同步解决问题：**

要想解决多个线程共享同一资源所带来的问题，则可以使用同步代码块或同步方法两种 方式完成。

1. 同步代码块是使用synchronized关键字声明的代码块。在使用synchronized声明同 步代码块时必须指定一个需要同步的对象，一般将当前对象最为同步对象，其格式 为：

synchronized( this ){

需要同步的代码；

}

1. 同步方法是使用synchronized关键字声明的方法，由于run( )方法是覆写的，所以一 般将run( )方法中的操作提取出来形成一个同步方法，然后由run( )方法进行调用。

**死锁：**

1）同步可以保证多个线程共享同一资源操作的正确性，但是过多的同步会产生死锁。 也就是同步带来的弊端。

2）死锁是指两个线程都在等待彼此先完成，造成了程序的停滞状态。

3）死锁是一种需要回避的代码，并且死锁是需要通过大量测试后才可以被检测出 来的一种程序非法状态。

**Java中方法定义的完整格式：**

访问权限{public default protected private} [final] [static] [synchronized] 返回值类型 方法 名称（参数类型 参数名称,…）[throws Exception1,…]{

方法体；

}

**Object类对线程的支持：**

1. public final void wait( ) throws InterruptedException：线程等待。
2. public final void wait( long timeout ) throws InterruptedException：线程等待，并指定 等待的最长时间，以毫秒为单位。
3. public final void notify( )：唤醒所有第一个等待的线程。
4. public final void notifyAll( )：唤醒全部等待的线程。

**Thread类中的其他方法：**

Thread类中除了常用的方法外，还有三个不常用的方法。

·suspend( )方法：暂时挂起线程

·resume( )方法：恢复挂起的线程

·stop( )方法：停止线程

它们都是使用@Deprecated注解声明的方法，都是不建议使用的方法，因为这三个方法 在使用时容易产生死锁。

**停止线程：**

由于stop( )方法是建议使用的方法，所以在开发中一般通过设置标志位的方式停止一个 线程的运行。