1. **线程的实现**

**实现Runnable接口，继承Thread类**

Java中的多线程是一种抢占式的机制而不是分时机制。抢占式机制指的是有多个线程处于可运行状态，但是只有一个线程在运行。       当有多个线程访问共享数据的时候，就需要对线程进行同步。

实例方法 new 对象 对象.实例方法()

静态方法（类方法） 类名.方法名()

日期的API

Java.util : Date

Java.time

**二、线程的常用方法**

1.实例start():启动一个线程用的是thread.start()方法，如果直接调用run方法是同步调用，相当于一个普通的方法调用。start()方法使线程开始执行，JVM会自动调用线程的run方法。new出来线程，调用start()方法即处于RUNNABLE(可运行)状态了。处于RUNNABLE状态的线程可能正在Java虚拟机中运行，也可能正在等待处理器的资源，因为一个线程必须获得CPU的资源后，才可以运行其run()方法中的内容，否则排队等待。

2. Thread.currentThread静态方法: 返回当前代码段正在被哪个调用的线程。

### 3. 实例方法 isAlive()：　isAlive方法用于判断当前线程是否处于活动状态。什么是活动状态呢?活动状态就是已经启动尚未终止的，线程处于正在运行或者准备开始运行的状态就认为线程是"活动"的。（新建状态的线程isAlive()返回的是false）

### 4. 静态方法 sleep()：在指定的毫秒数内让当前"正在执行的线程"休眠(暂停执行)。 这个"正在执行的线程"是指Thread.currentThread()返回的线程。但是sleep不会释放锁。

### 5. Thread.yield()静态方法，没有参数

此方法的作用是放弃当前的CPU资源，将它让给其他的任务去占用CPU的时间。　实际上，yield()方法对应了如下操作；先检测当前是否有相同优先级的线程处于同可运行状态，如有，则把CPU的占有权交给次线程，否则继续运行原来的线程，所以yield()方法称为“退让”，它把运行机会让给了同等级的其他线程。

### **6.** join实例方法:

### join方法的主要作用就是同步，它可以使得线程之间的并行执行变为串行执行。在A线程中调用了B线程的join()方法时，表示只有当B线程执行完毕时，A线程才能继续执行**。**

### **7.wait实例方法：使线程进入等待队列**

### **8.sleep方法和wait方法有什么区别呢？**

### 9. notitf() notifyAll()

**三、线程的状态以及状态转换**

新建、就绪、运行、阻塞、死亡。



文件I/O

1. File类
2. 文件的读写

字节流：InputStream OutputStream

字符流：Reader Writer

1. 文本文件的读写
2. **用FileInputStream和FileOutputStream读写文本文件 字节流读取文件**

**（2） 用BufferedReader和BufferedWriter读写文本文件**

FileInputStream按字节读取文件

FileReader 按字符读取文件

FileOutputStream 按字节写入文件

FileWriter 按字符写文件

PrintStream

 \* 可以将各种基本数据类型（int、double、boolean）和各种引用数据类型（String、Date、Student...）  
 \* 的数据以【字符串的形式】【直接】写入到目的地中，并且具有换行的功能  
 \*  直接：不需要转换成字节数组

1. 数据库

Map HashMap 三个元素 put三个元素 第一元素 put 一个课程的名字（String）

第二个元素 put 一个teacher对象 第三个元素 put 一个list list里面存放学生对象

打印输出到文件

类：

1. 构造方法
2. this static final

继承：

1. extends
2. super
3. 访问修饰符