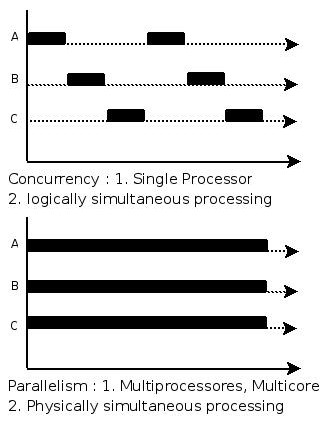
**并发和并行的区别**

并发和并行的区别就是一个处理器同时处理多个任务和多个处理器或者是多核的处理器同时处理多个不同的任务。  
前者是逻辑上的同时发生（simultaneous），而后者是物理上的同时发生．

并发性(concurrency)，又称共行性，是指能处理多个同时性活动的能力，并发事件之间不一定要同一时刻发生。

并行(parallelism)是指同时发生的两个并发事件，具有并发的含义，而并发则不一定并行。

来个比喻：并发和并行的区别就是一个人同时吃三个馒头和三个人同时吃三个馒头。

[](http://images.51cto.com/files/uploadimg/20090804/1503300.jpg)

这样，对于并发和并行的区别，您应该了解了吧。

线程级并发

在进程的抽象概念下引入了线程，而线程级并发的概念，就是指的多个线程在同一时间（并非是绝对同时的）活动。

操作系统从单处理器，直到现在多核多处理器系统，乃至超线程技术，已经经历了很大的变化。这也使得针对多线程编程变得更加重要，否则就无法利用多处理器带来的好处。

针对多处理器系统来说，比较好理解，其实就是物理上将多个CPU集中在一个集成电路的芯片上。而对于超线程技术来说，则是利用N个物理内核，模拟出2N个逻辑内核的技术。在硬件上来讲，超线程需要CPU的某些部分有多个备份，比如寄存器和程序计数器，但是其它部分只有一份，比如ALU。

指令级并行

在书(计算机系统原理)中，对指令级并行的解释是，如果处理器可以同时执行多条指令，则称这种属性为指令级并行。其实指令级并行就是利用了指令的执行过程中会有不同的阶段，或者更精确的说，是在同一时间只会利用部分CPU的硬件，因此可以利用这一点做到多个指令并行执行。

更好的情况下，现代的很多处理器能够做到执行一条指令的平均时间尚且不到一个周期，这种处理器就称为超标量处理器。

单指令、多数据并行

单指令、多数据的概念是指一条指令可以产生多个并行执行的操作的方式。当今的一些处理器中配备了特殊的硬件，可以达到这个效果。由于产生了多个并行执行的操作，因此就会涉及到多个数据，通俗的讲也可以理解为，一条指令操作多个数据。比如书中所提到的例子，一些处理器具有并行地对4对单精度浮点数做加法的指令。

简谈抽象

抽象的重要性就不需要再强调了，它在计算机科学领域有着不言而喻的地位。抽象可以使得一些具体的实现变的更加易于描述，而且也可以针对一些实现的方式作出规定。

举个简单的例子，就JAVA中的类来说，它其实是通过编译器和JVM来实现的，而JVM本身又是一个抽象的概念，它也会有具体的实现。倘若针对我们平时使用的hotspot虚拟机来说，类的实现是将类的信息存放在永久代，而后将实例存放在堆中，并且在每一个实例中，都会存放一个类信息的引用。从而我们在操作这个实例的时候，会通过类信息确定我们所做的操作并执行它。

以上是LZ对类实现的个人理解，对JVM不太熟悉的猿友们可能对这个描述比较迷惑。但是没关系，你只需要知道class可以声明一个类，并且在你创建了一个实例之后，使用实例名.方法名可以调用它的方法，实例名.变量名可以取得它的属性值就可以了（在此简单起见，忽略访问权限限制）。这就使得我们操作类的时候变的简单，也正是抽象的意义之一，也是上面对抽象描述的第一句话的解释。

对于后面一句就比较好理解了，JAVA虚拟机是一种抽象，有了这个抽象，我们就可以针对JAVA虚拟机制定规范，也就是JAVA虚拟机规范。