在并发编程中，有两个基本的执行单元：进程和线程。在java语言中，并发编程最关心的是线程，然而，进程也是非常重要的。

即使在只有单一的执行核心的计算机系统中，也有许多活动的进程和线程。因此，在任何给定的时刻，只有一个线程在实际执行。处理器的处理时间是通过操作系统的时间片在进程和线程中共享的。  
现在具有多处理器或有多个执行内核的多处理器的计算机系统越来越普遍，这大大增强了系统并发执行的进程和线程的吞吐量–但在不没有多个处理器或执行内核的简单的系统中，并发任然是可能的。

**进程**

进程具有一个独立的执行环境。通常情况下，进程拥有一个完整的、私有的基本运行资源集合。特别地，每个进程都有自己的内存空间。  
进程往往被看作是程序或应用的代名词，然而，用户看到的一个单独的应用程序实际上可能是一组相互协作的进程集合。为了便于进程之间的通信，大多数操作系统都支持进程间通信（IPC），如pipes 和sockets。IPC不仅支持同一系统上的通信，也支持不同的系统。  
Java虚拟机的大多数实现是单进程的。Java应用可以使用的ProcessBuilder对象创建额外的进程，多进程应用超出了本课的范围。

**线程**

线程有时也被称为轻量级的进程。进程和线程都提供了一个执行环境，但创建一个新的线程比创建一个新的进程需要的资源要少。  
线程是在进程中存在的 — 每个进程最少有一个线程。线程共享进程的资源，包括内存和打开的文件。这样提高了效率，但潜在的问题就是线程间的通信。  
多线程的执行是Java平台的一个基本特征。每个应用都至少有一个线程 – 或几个，如果算上“系统”线程的话，比如内存管理和信号处理等。但是从程序员的角度来看，启动的只有一个线程，叫主线程。这个线程有能力创建额外的线程，我们将在下一节演示。