**Concurrency**

Computer users take it for granted that their systems can do more than one thing at a time. They assume that they can continue to work in a word processor, while other applications download files, manage the print queue, and stream audio.

In concurrent programming, there are two basic units of execution: ***processes*** and ***threads***

**Process:**

A process has a self-contained execution environment. A process generally has a complete, private set of basic run-time resources; in particular, each process has its own memory space.

**Thread:**

A thread is a light-weight smallest part of a process that can run concurrently with the other parts (other threads) of the same process.

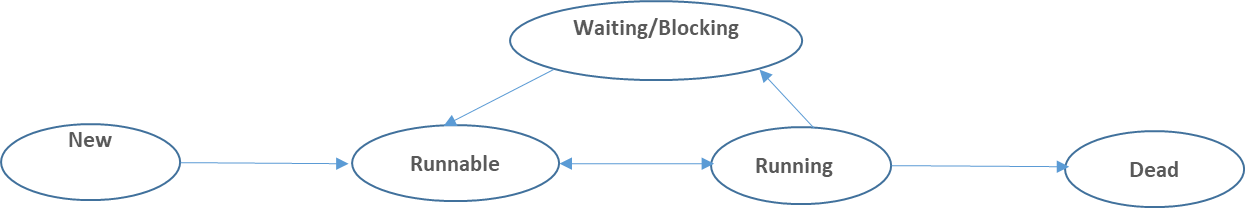
**Multi-Threading**:

The process of executing multiple threads simultaneously is known as multithreading.

**Process Vs Thread:**

* A program in execution is often referred as process. A thread is a subset(part) of the process.
* A process consists of multiple threads. A thread is a smallest part of the process that can execute concurrently with other parts(threads) of the process.
* A process is sometime referred as task. A thread is often referred as lightweight process.
* A process has its own address space. A thread uses the process’s address space and share it with the other threads of that process.
* Threads have control over the other threads of the same process. A process does not have control over the sibling process, it has control over its child processes only.

**Thread life cycle:**



**New:** This is the state the thread is in after the ***Thread*** instance has been created, but the ***start ()*** method has not been invoked on the thread.

**Runnable:** This is the state the thread is in when it is eligible to run, but the thread scheduler has not selected it to be the running thread.

**Running:** This is the state a thread is in when thread scheduler selects it for execution by allocating the needed resources.

**Waiting/ Blocking:** This is the state a thread is in when it is not eligible to run. (The thread may be blocked waiting for resources, like I/O or an object that is needed by the thread is locked)

**Dead:** A thread is considered *dead* when its ***run ()*** method completes. If once the thread is dead, it can never be brought back to **run** or **runnable** state.