Concurrency and Multi Threading

* Program :- A program exist as a passive entity. Passive entity means, just as code which is not yet executed. Once a program starts it can start multiple processes.
* Process :- It is a active entity. It refers to a program that has been loaded into the main memory (RAM). A process can also be described as an instance of program on sec. mem. running on a computer.
* Thread :- One unit of CPU execution. Software concept. Executed on cores. Big Task, breaks into smaller one, each getting executed by individual thred. Core executes Threads. Every process need to have atleast one thread.
* Concurrency :- One processor executing multiple tasks sequencially by context switching thereby creating an illusion of parallelism.
* Parallelism :- Multiple processors executing multiple process at the same time achieving actuall parallelism.

Thred is a single sequence stream within a process. Basically stream of instruction. Also called lightweight process. Threds share process's memory.

* Monitors in synchronization :- A Monitor can be a class or an object. For example

```
class A {

    sync void fun1 (...){
    //...
    }
    sync void fun2(...){
    //...
    }
}
```

synced on class. class is the Monitor

```
class B {
  Object lock1 = new Objet();

  void fun1(...){
    sync (lock1){
    //....
    }
  void fun2 (...){
    sync(lock1){
    //...
    }
}
```

synced on lock1. lock1 is the monitor

* creating multiple lock object (can be any object), and syncing on those object can cause different results.

new Thread (A:: fun1)	new Thread (B:: fun1)
new Thread (A:: fun2)	new Thread (B:: fun2)

in Both cases the thred will move sequencially, first fun1 then fun2. as A is locked on class and B is locked on same object

```
class C {
   Obj lock1 = ...
   Obj lock2 = ...
   void fun1(...){
      sync(lock1){
         //...
      }
   }
   void fun2(...){
      sync(lock2){
         //...
      }
   }
}
```

both methods are locked on different lock objects.

new Thread (C:: fun1)
new Thread (C:: fun2)
both the threads above will behave async.

* (obj :: method) ⇒ returns a Runnable.
* (class :: method) ⇒ returns a Consumer <>
* FutureTask $\xrightarrow{I}$ Runnable Future $\xrightarrow{E}$ Runnable
  Future $\xrightarrow{E}$ Future <V>
* Callable is just a functional interface.