**Runnable Interface:**class implementing runnable interface create thread by passing its object to Thread class .While class inheriting Thread class create threads by their own object.start.

**Joins:** current thread can wait until execution of thread on which join is called stops.

**Synchronisation:** Threads share memory and resources with each other .this can result in inconsistencies .So to resolve such issues threads are synchronised .

*Synchronised methods:* method are made synchronised by adding synchronised keyword in declaration .It means they can’t be simultaneously invoked by multiple threads .

*Monitor:*Locs of object’s instances are called monitors .when a thread access a field it first acquire that lock and then access and then release when returning .

*Reentrant Synchronisation*:A thread can acquire lock that it already owns .In situation when synchronised code calls itself i.e.recursion .

*Synchronised Statements:* By executing synchronised statements we can acquire lock on any object .Unlike methods in which we can acquire lock only on the calling thread .[Synchronised Statements](https://stackoverflow.com/questions/1268016/what-is-synchronized-statement-used-for/34811859#34811859)

Threads respond to each and every action of other thread if the action to which they are responding is also a response .they keep themselves busy in responding to each other ,situation is called LIVELOCK.

**Immutable Objects:**Objects whose state can’t be changed after their construction.

**Executors:**To separate thread creation and management part from rest of the application ,Executors come into the picture .

**Thread Pools :** pool of worker threads .worker threads are assigned to request from pool .

**Atomic** Variables :Variables on which atomic operation are supported .multiple threads can’t access at same time .