FACULTY OF Engineering & Technology

Starting a thread:

start() method of Thread class is used to start a newly created thread.

It performs following tasks

The thread moves from New state to the Runnable state.

When the thread gets a chance by cpu to execute, its target run() method will run.

1) **By extending Thread class**

```
class  rama extends Thread
{
public void run(){
System.out.println("thread is in running state");  //override the run() method of Thread class
}
public static void main(String args[])
{
rama r1=new rama();
r1.start();
 }
}
```

Output:thread is in running state

2. By implementing Runnable interface

```
class rama implements Runnable
{
  public void run()
  {
  System.out.println("thread runs by runnable interface");
  }

   public static void main(String args[])
   {
   rama r1=new rama();
   Thread t1 =new Thread(r1);
    t1.start();
   }
}
```

Output: thread runs by runnable interface

Synchronization

1. Multithreading introduces asynchronous behaviour to the programs. If a thread is writing some data another thread may be reading the same data at that time. This may bring inconsistency.
2. When two or more threads need access to a shared resource there should be some way that the resource will be used only by one resource at a time. The process to achieve this is called synchronization.
3. To implement the synchronous behavior java has synchronous method. Once a thread is inside a synchronized method, no other thread can call any other synchronized method on the same object. All the other threads then wait until the first thread come out of the synchronized block.

Syntax:

```
synchronized(object)
{
    // statement to be synchronized
}
```