Multithreading

LifeCycle

- **new** -> when you create a thread
- **runnable** -> after invocation of start()
- **running** -> when thread scheduler actually selects the thread
- **non-runnable (blocked)** -> sleep(), wait(), suspend()
- **terminated** -> end of run() or stop()

Creation

- By extending thread class
- By implementing runnable interface

Eg1 :-

```
class Example extends Thread {
    public void run {
        syso("Thread running..");
    }

    pvsm(){
        Example e = new Example();
        e.start();
    }
}
```

Eg2 :-

```
class Example implements Runnable {
    public void run {
        syso("Thread running..");
    }

    pvsm(){
        Example e = new Example();
        Thread t = new Thread(e);
        t.start();
    }
}
```

- ***IllegalThreadStateException*** *is throw if a thread is started twice.*
- Each thread starts in a separate call stack. If we call ***run()*** method directly, the ***run()*** method goes onto the current call stack rather than at the beginning of a new call stack.
- The ***join()*** method waits for a thread to die. It causes the currently

running threads to stop executing until the thread it joins with completes its task.

- ***Daemon thread*** in java is a service provider thread that provides services to the user thread. Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically. e.g. gc, finalizer etc
- A user thread must be marked as Daemon before it starts otherwise it will throw ***IllegalThreadStateException.***
    - ***t1.setDaemon(true);***
    - ***t1.start();***

Java Thread Pool

- a group of fixed size threads are created. A thread from the thread pool is pulled out and assigned a job by the service provider. After completion of the job, thread is contained in the thread pool again.
- It saves time because there is no need to create new thread.

Eg :-

```
class WorkerThread implements Runnable {
    public void run(){
        syso("running thread" +
Thread.currentThread().getName());
    }
}

class ThreadPoolTest {
    pvsm(){
        ExecutorService e = Executors.newFixedThreadPool(5);

        for(int i=0;i<10;i++) {
            Runnable worker = new Worker();
            e.execute(worker);
        }
        e.shutdown();
    }
}
```