**Problem 3:**

**1. Threads with their life cycle (With code)**

Ans:

What is Thread?

Threads can help us to do parallel processing.Threads are useful when you want to run multiple pieces of code in parallel.

A thread can be defined as a lightweight process which can execute multiple codes in parallel.However,the thread is different from a process. In OS,for each process,a separate memory will be allocated.And the same is applicable for thread as well,it has separate memory.All the threads will run in the same memory which is allocated for the process.

Java is a *multi-threaded programming language.*A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

Multitasking is when multiple processes share common processing resources such as a CPU. **Multi-threading** extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each **thread** within an application.

## Life Cycle of a Thread

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The following diagram shows the complete life cycle of a thread.

Following are the stages of the life cycle −

**New** − A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a **born thread**.

**Runnable** − After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

**Waiting** − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

**Timed Waiting** − A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

**Terminated (Dead)** − A runnable thread enters the terminated state when it completes its task or otherwise terminates.

## Thread Priorities

Every Java thread has a priority that helps the operating system determine the order in which threads are scheduled.

Java thread priorities are in the range between MIN\_PRIORITY (a constant of 1) and MAX\_PRIORITY (a constant of 10). By default, every thread is given priority NORM\_PRIORITY (a constant of 5).

Threads with higher priority are more important to a program and should be allocated processor time before lower-priority threads. However, thread priorities cannot guarantee the order in which threads execute and are very much platform dependent

**A Thread can be created in Java in the following ways:**

1. By Extending Thread class
2. Implementing Runnable interface