* Thread is like a lightweight process.
* Multiple processes or threads are called Multi-Threading.
* By using threads we can perform multiple operations at same time.

**Thread Life cycle:**

5 stages:

1. **New**: we will create a new Thread in this stage.
2. **Runnable**: we will call start() method, that start() method internally calls run() method.
3. **Running**: the logic whatever we write inside run() method that will get executed.
4. **Non-Runnable**: thread will be in pause/sleep in non-running state.
5. **Terminate**: once thread completes it tasks it will get terminated.

**How many ways of creating a Thread: 2 ways**

1. By extending **Thread class**.
2. By implementing **Runnable Interface**.

Thread methods:

* Start(): thread execution start by calling run() method.
* Run(): in run() method whatever logic you write It will execute.
* Join(): if 2 threads are running and you want 1 thread execution completion after another thread to be execute.
* Sleep(): If you to make thread to sleep for some seconds then we will use this method.

Run() method is void type and without parameters

Thread priority:

* Each Threads has a priority.
* This will be from 1 to 10.
* Thread Scheduler will schedules the priority.

2 methods,

SetPriority(int num) 🡺 will set priority

getPriority() 🡺 will fetch priority.

Default:

MIN\_priority: 1

NORM\_Priority: 5

MAX\_Priority: 10

Asynchronization:

Parallel execution of threads.

By default it will execute parallelly

Synchronization:

* Execution of a thread to happen one after another thread tasks completion.
* If there is any shared resource. In that place synchronization is best choice.

Types:

* Method level synchronization / Object level
* Synchronization block
* Static synchronization / class level

Deadlock:

Thread Pool:

Group of threads are waiting for the tasks to complete and such that we can use it multiple times.

4 request 🡺 5 threads pool

10 request 🡺5 threads in thread pool

Diagram

Description automatically generated