SNS COLLEGE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
COIMBATORE – 35
DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING

UNIT 4

What Is a Thread In Java?

- We use threads to make Java applications faster by doing multiple things at the same time.
- Thread helps us to achieve parallelism in Java programs.
- Since the CPU is high-speed and it even contains multiple cores, just one Thread cannot take advantage of all the cores.
- It helps in the usage of multiple threads running at the same time and performing different tasks in a single program.
- A very good example of thread-based multithreading is a word processing program that checks the spelling of words in a document while writing the document.

Life Cycle Of a Thread

1) NEW:

- Whenever a new thread is created, it is always in the new state.
- For a thread in the new state, the code has not been run yet and thus has not begun its execution.

2) ACTIVE:

- When a thread invokes the start() method, it moves from the new state to the active state. This state has two states within it.

  i) Runnable:

  A thread, that is ready to run is then moved to the runnable state. In the runnable state, the thread may be running or may be ready to run at any given instant of time. It is the duty of the thread scheduler to provide the thread time to run, i.e., moving the thread to the running state.

ii) Running:

When the thread gets the CPU, it moves from the runnable to the running state. Generally, the most common change in the state of a thread is from runnable to running and again back to runnable.

3) BLOCKED OR WAITING:

- Whenever a thread is inactive for a span of time (not permanently) then, either the thread is in the blocked state or is in the waiting state.
- When the main thread invokes the join() method then, it is said that the main thread is in the waiting state. The main thread then waits for the child threads to complete their tasks. When the child threads complete their job, a notification is sent to the main thread, which again moves the thread from waiting to the active state.
- If there are a lot of threads in the waiting or blocked state, then it is the duty of the thread scheduler to determine which thread to choose and which one to reject, and the chosen thread is then given the opportunity to run.

4) TIMED WAITING:

- Sometimes, waiting for leads to starvation.
- For example, a thread (its name is A) has entered the critical section of a code and is not willing to leave that critical section. In such a scenario, another thread (its name is B) has to wait forever, which leads to starvation. To avoid such scenario, a timed waiting state is given to thread B. Thus, thread lies in the waiting state for a specific span of time, and not forever.
- A real example of timed waiting is when we invoke the sleep() method on a specific thread. The sleep() method puts the thread in the timed wait state. After the time runs out, the thread wakes up and start its execution from when it has left earlier.

5) TERMINATED:

- A terminated thread means the thread is no more in the system. In other words, the thread is dead, and there is no way one can respawn (active after kill) the dead thread.

Java Thread Methods:

1) start() : It is used to start the execution of the thread
2) run() : It is used to do an action for a thread
3) sleep() : It sleeps a thread for a specific amount of time
4) join() : It waits for a thread to die
5) suspend() : It is used to suspend the thread
6) resume() : It is used to resume the suspended thread
7) stop() : It is used to stop the thread
8) destroy() : It is used to destroy the thread group and all of its subgroup

Starting a Thread:

- The **start() method** of Thread class is used to start a newly created thread.
- A new thread starts(with new callstack).
- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

How To Create a Thread In Java:

1) By extending thread class:
    - Thread class provide constructors and methods to create and perform operations on a thread.Thread class extends Object class and implements Runnable interface.
    - Commonly used Constructors of Thread class:
        1.Thread()
        2.Thread(String name)
        3.Thread(Runnable r)
        4.Thread(Runnable r,String name)

2) By implementing runnable interface:
    - The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.
    - Runnable interface have only one method named run().

Java Thread Example By Extending Thread Class:

```
1. class Multi extends Thread{
2. public void run(){
3. System.out.println("thread is running...");
4. }
5. public static void main(String args[]){
6. Multi t1=new Multi();
7. t1.start();
8.  }
9. }
```

OUTPUT: thread is running...

Java Thread Example By Using Runnable Interface:

```
1. class Multi3 implements Runnable{
2. public void run(){
3. System.out.println("thread is running...");
4. }
5. public static void main(String args[]){
6. Multi3 m1=new Multi3();
7. Thread t1 =new Thread(m1);   // Using the constructor Thread(Runnable r)
8. t1.start();
9.  }
10. }
```

OUTPUT: thread is running...