**13. Multithreading**

• **Theory:**

**1.Introduction to Threads:-** Thread is a smallest unique of process at a time only single thread can be execute .  A Java application can have multiple threads running concurrently.

**2.Creating Threads by Extending Thread Class or Implementing Runnable Interface:-**

* **by Extending Thread** :- When you extend the Thread class, you create a new class that inherits from Thread. You need to override the run() method to define the code that will execute in the thread. is a simple and direct, suitable for small tasks where the thread's functionality is the primary concern.

**Syntax**:- Class A extends thread{

@override

Public void run()

{

//code

}

* **Implementing Runnable Interface:-**Implementing the Runnable interface is a more flexible approach. You create a class that implements Runnable and define the run() method. You then pass an instance of your class to a Thread object. is recommended for more complex scenarios, especially when you need to share the same task among multiple threads or when your class needs to extend another class.

**Syntax:**- Class A implements Runable{

Public void main(){

//code;

}

}

Class B{

Public static void main(){

A r=newA()

Thread t=new thread();

Thread(r);

t.start();

}

**3.Thread Life Cycle:-**

New

|

| start()

v

Runnable <-------------------+

| |

| (scheduler) |

| (synchronized lock) |

v |

Running <----------------- Blocked

| |

| (waits for CPU) |

| (wait, join, sleep) |

v |

Terminated <----------------+

**4.Synchronization and Inter-thread Communication**

* **Synchronization**:- object to binary stream converting process is called Synchronization.
* **Inter-thread communication:-** in Java is a mechanism that allows threads to communicate and synchronize their actions, ensuring they can coordinate access to shared resources effectively.