SNS COLLEGE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

COIMBATORE – 35

DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING

# UNIT IV
# MULTITHREADING IN JAVA

## Thread Synchronization

### Thread synchronization

Synchronization in Java is the capability to control the access of multiple threads to any shared resource.

Java Synchronization is better option where we want to allow only one thread to access the shared resource.

### Why use Synchronization?

The synchronization is mainly used to

1. To prevent thread interference.
2. To prevent consistency problem.

### Types of Synchronization

There are two types of synchronization

1. Process Synchronization
2. Thread Synchronization

### Thread Synchronization

There are two types of thread synchronization mutual exclusive and inter-thread communication.

1. Mutual Exclusive
    1. Synchronized method.
    2. Synchronized block.

    3. Static synchronization.
2. Cooperation (Inter-thread communication in java)

## Mutual Exclusive

Mutual Exclusive helps keep threads from interfering with one another while sharing data. It can be achieved by using the following three ways:

1. By Using Synchronized Method
2. By Using Synchronized Block
3. By Using Static Synchronization

## Concept of Lock in Java

Synchronization is built around an internal entity known as the lock or monitor. Every object has a lock associated with it. By convention, a thread that needs consistent access to an object's fields has to acquire the object's lock before accessing them, and then release the lock when it's done with them.

From Java 5 the package java.util.concurrent.locks contains several lock implementations.

## Understanding the problem without Synchronization

In this example, there is no synchronization, so output is inconsistent. Let's see the example:

```
class Table{

                    void printTable(int n){//method not synchronized
                      for(int i=1;i<=5;i++){
                        System.out.println(n*i);
                        try{
                         Thread.sleep(400);
                        }catch(Exception e){System.out.println(e);}
                      }

                     }
                    }

                    class MyThread1 extends Thread{
                    Table t;
                    MyThread1(Table t){
                    this.t=t;
                    }
                    public void run(){
```

```
t.printTable(5);
}

}
class MyThread2 extends Thread{
Table t;
MyThread2(Table t){
this.t=t;
}
public void run(){
t.printTable(100);
}
}

class TestSynchronization1{
public static void main(String args[]){
Table obj = new Table();//only one object
MyThread1 t1=new MyThread1(obj);
MyThread2 t2=new MyThread2(obj);
t1.start();
t2.start();
}
}
```