SNS COLLEGE OF TECHNOLOGY

(an autonomous institution)

Coimbatore - 35

19CST102 - OBJECT ORIENTED PROGRAMMING

I YEAR / II SEMESTER

UNIT IV – MULTITHREADING IN JAVA

TOPIC : THREADS SYNCHRONIZATION

Guided by :

Mr.Selvakumar. N

AP/CSE

Presented by:

Rithika M

(713522CS127)

Rohini R

(713522CS128)

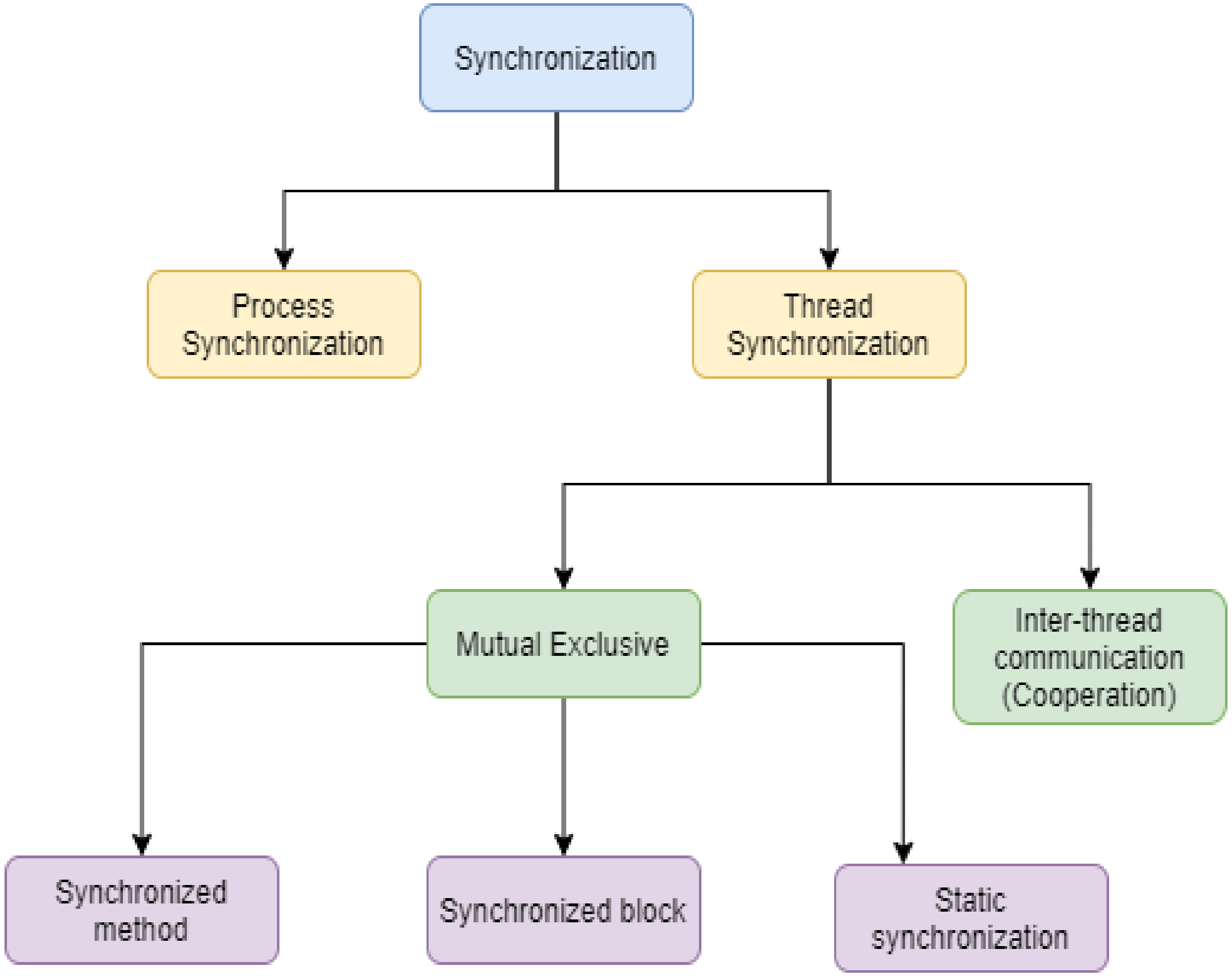
THREADS :

- Threads allows a program to operate more efficiently by doing multiple things at the same time.
- Threads can be used to perform complicated tasks in the background without interrupting the main program.

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.

TYPES OF SYNCHRONIZATION :



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

- Synchronized Method
- Synchronized Block
- Static Synchronization

SYNCHRONIZED METHOD :

- If you declare any method as synchronized, it is known as synchronized method.
- Synchronized method is used to lock an object for any shared resource.
- When a thread invokes a synchronized method, it automatically acquires the lock for that object and releases it when the thread completes its task.

Example program without using synchronization :

```
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();
}
}
```

Output:

```
 5
100
10
200
15
300
20
400
25
500
```

Example program using synchronization :

```
class Table{
Synchronized void printTable(int n){//method
   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();
}
}
```

Output :

```
5
10
15
20
25
100
200
300
400
500
```

Synchronized block :

- Synchronized block can be used to perform synchronization on any specific resource of the method.

- Suppose we have 50 lines of code in our method, but we want to synchronize only 5 lines, in such cases, we can use synchronized block.

- If we put all the codes of the method in the synchronized block, it will work same as the synchronized method.

Example of Synchronized Block :

```
class Table
{
 void printTable(int n){
   synchronized(this){//synchronized block
     for(int i=1;i<=5;i++){
```

```
System.out.println(n*i);
      try{
       Thread.sleep(400);
      }catch(Exception e)
System.out.println(e);}
     }
   }
 }//end of the method
}
 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);
}
}
public class TestSynchronizedBlock1{
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();
}
}
```

Output:

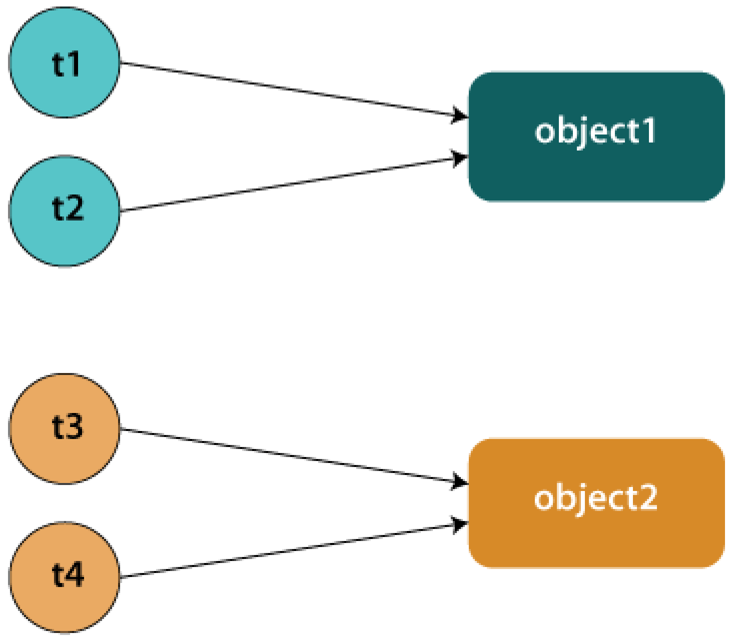
```
5
10
15
20
25
100
200
300
400
500
```

Static Synchronization :

If you make any static method as synchronized, the lock will be on the class not on object.

Problem without static synchronization

Suppose there are two objects of a shared class (e.g. Table) named object1 and object2. In case of synchronized method and synchronized block there cannot be interference between t1 and t2 or t3 and t4 because t1 and t2 both refers to a common object that have a single lock. But there can be interference between t1 and t3 or t2 and t4 because t1 acquires another lock and t3 acquires another lock. We don't want interference between t1 and t3 or t2 and t4. Static synchronization solves this problem.



thank you