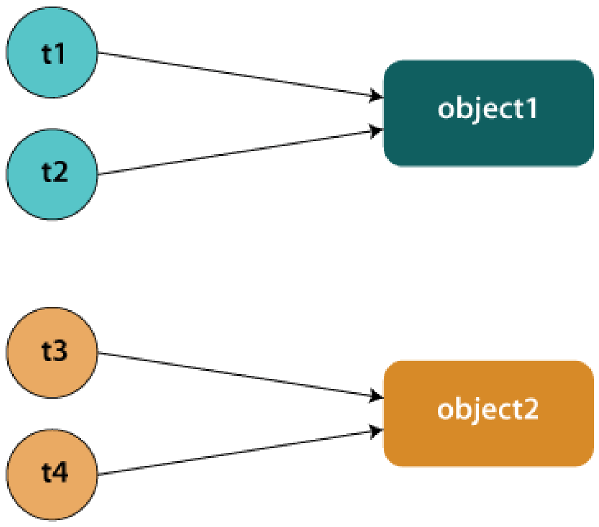
Static Synchronization

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.

Example of Static Synchronization

In this example we have used **synchronized** keyword on the static method to perform static synchronization.

TestSynchronization4.java

```
class Table
{
 synchronized static void printTable(int n){
   for(int i=1;i<=10;i++){
     System.out.println(n*i);
     try{
       Thread.sleep(400);
     }catch(Exception e){}
   }
 }
}
class MyThread1 extends Thread{
public void run(){
Table.printTable(1);
}
}
class MyThread2 extends Thread{
public void run(){
Table.printTable(10);
}
}
class MyThread3 extends Thread{
public void run(){
Table.printTable(100);
}
}
class MyThread4 extends Thread{
public void run(){
Table.printTable(1000);
}
}
public class TestSynchronization4{
public static void main(String t[]){
MyThread1 t1=new MyThread1();
MyThread2 t2=new MyThread2();
MyThread3 t3=new MyThread3();
MyThread4 t4=new MyThread4();
t1.start();
t2.start();
t3.start();
t4.start();
}
}
```

Test it Now

Output:

```
1
2
```

```
3
4
5
6
7
8
9
10
10
20
30
40
50
60
70
80
90
100
100
200
300
400
500
600
700
800
900
1000
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
```

Example of static synchronization by Using the anonymous class

In this example, we are using anonymous class to create the threads.

TestSynchronization5.java Hello

```
class Table{
 synchronized static  void printTable(int n){
   for(int i=1;i<=10;i++){
     System.out.println(n*i);
     try{
       Thread.sleep(400);
     }catch(Exception e){}
   }
 }
}

public class TestSynchronization5 {
public static void main(String[] args) {

    Thread t1=new Thread(){
        public void run(){
            Table.printTable(1);
        }
    };

    Thread t2=new Thread(){
        public void run(){
            Table.printTable(10);
        }
    };

    Thread t3=new Thread(){
        public void run(){
            Table.printTable(100);
        }
    };

    Thread t4=new Thread(){
        public void run(){
            Table.printTable(1000);
        }
    };
    t1.start();
    t2.start();
    t3.start();
    t4.start();

}
}
```

Test it Now

Output:

```
1
2
3
4
5
6
7
8
9
10
10
20
30
40
50
60
70
80
90
100
100
200
300
400
500
600
700
800
900
1000
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
```

Synchronized block on a class lock:

The block synchronizes on the lock of the object denoted by the reference .class name .class. A static synchronized method printTable(int n) in class Table is equivalent to the following declaration:

```
static void printTable(int n) {
    synchronized (Table.class) {       // Synchronized block on class A
        // ...
    }
}
```

← Prev Next →