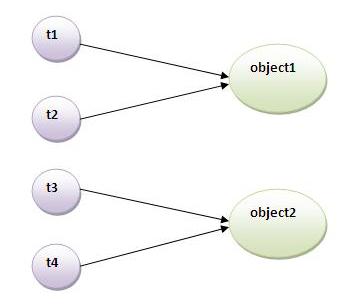
Static synchronization

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.I want no interference between t1 and t3 or t2 and t4.Static synchronization solves this problem.

Example of static synchronization

In this example we are applying synchronized keyword on the static method to perform static synchronization.

1. **class** Table{
3. **synchronized** **static** **void** printTable(**int** n){
4. **for**(**int** i=1;i<=10;i++){
5. System.out.println(n\*i);
6. **try**{
7. Thread.sleep(400);
8. }**catch**(Exception e){}
9. }
10. }
11. }
13. **class** MyThread1 **extends** Thread{
14. **public** **void** run(){
15. Table.printTable(1);
16. }
17. }
19. **class** MyThread2 **extends** Thread{
20. **public** **void** run(){
21. Table.printTable(10);
22. }
23. }
25. **class** MyThread3 **extends** Thread{
26. **public** **void** run(){
27. Table.printTable(100);
28. }
29. }



34. **class** MyThread4 **extends** Thread{
35. **public** **void** run(){
36. Table.printTable(1000);
37. }
38. }
40. **public** **class** TestSynchronization4{
41. **public** **static** **void** main(String t[]){
42. MyThread1 t1=**new** MyThread1();
43. MyThread2 t2=**new** MyThread2();
44. MyThread3 t3=**new** MyThread3();
45. MyThread4 t4=**new** MyThread4();
46. t1.start();
47. t2.start();
48. t3.start();
49. t4.start();
50. }
51. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestSynchronization4)

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

Same example of static synchronization by annonymous class

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

1. **class** Table{
3. **synchronized** **static**  **void** printTable(**int** n){
4. **for**(**int** i=1;i<=10;i++){
5. System.out.println(n\*i);
6. **try**{
7. Thread.sleep(400);
8. }**catch**(Exception e){}
9. }
10. }
11. }
13. **public** **class** TestSynchronization5 {
14. **public** **static** **void** main(String[] args) {
16. Thread t1=**new** Thread(){
17. **public** **void** run(){
18. Table.printTable(1);
19. }
20. };
22. Thread t2=**new** Thread(){
23. **public** **void** run(){
24. Table.printTable(10);
25. }
26. };
28. Thread t3=**new** Thread(){
29. **public** **void** run(){
30. Table.printTable(100);
31. }
32. };
34. Thread t4=**new** Thread(){
35. **public** **void** run(){
36. Table.printTable(1000);
37. }
38. };
39. t1.start();
40. t2.start();
41. t3.start();
42. t4.start();
44. }
45. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestSynchronization5)

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:

1. **static** **void** printTable(**int** n) {
2. **synchronized** (Table.**class**) {       // Synchronized block on class A
3. // ...
4. }
5. }