Synchronization in Java

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.

### 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. This can be done by three ways in java:

1. by synchronized method
2. by synchronized block
3. by static synchronization

Concept of Lock in Java

Synchronization is built around an internal entity known as the lock or monitor. Every object has an 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:

1. **class** Table{
2. **void** printTable(**int** n){//method not synchronized
3. **for**(**int** i=1;i<=5;i++){
4. System.out.println(n\*i);
5. **try**{
6. Thread.sleep(400);
7. }**catch**(Exception e){System.out.println(e);}
8. }
10. }
11. }
13. **class** MyThread1 **extends** Thread{
14. Table t;
15. MyThread1(Table t){
16. **this**.t=t;
17. }
18. **public** **void** run(){
19. t.printTable(5);
20. }
22. }
23. **class** MyThread2 **extends** Thread{
24. Table t;
25. MyThread2(Table t){
26. **this**.t=t;
27. }
28. **public** **void** run(){
29. t.printTable(100);
30. }
31. }
33. **class** TestSynchronization1{
34. **public** **static** **void** main(String args[]){
35. Table obj = **new** Table();//only one object
36. MyThread1 t1=**new** MyThread1(obj);
37. MyThread2 t2=**new** MyThread2(obj);
38. t1.start();
39. t2.start();
40. }
41. }

Output: 5

100

10

200

15

300

20

400

25

500

Java 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.

1. //example of java synchronized method
2. **class** Table{
3. **synchronized** **void** printTable(**int** n){//synchronized method
4. **for**(**int** i=1;i<=5;i++){
5. System.out.println(n\*i);
6. **try**{
7. Thread.sleep(400);
8. }**catch**(Exception e){System.out.println(e);}
9. }
11. }
12. }
14. **class** MyThread1 **extends** Thread{
15. Table t;
16. MyThread1(Table t){
17. **this**.t=t;
18. }
19. **public** **void** run(){
20. t.printTable(5);
21. }
23. }
24. **class** MyThread2 **extends** Thread{
25. Table t;
26. MyThread2(Table t){
27. **this**.t=t;
28. }
29. **public** **void** run(){
30. t.printTable(100);
31. }
32. }
34. **public** **class** TestSynchronization2{
35. **public** **static** **void** main(String args[]){
36. Table obj = **new** Table();//only one object
37. MyThread1 t1=**new** MyThread1(obj);
38. MyThread2 t2=**new** MyThread2(obj);
39. t1.start();
40. t2.start();
41. }
42. }

Output: 5

10

15

20

25

100

200

300

400

500

# Synchronized Block in Java

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

Suppose you have 50 lines of code in your method, but you want to synchronize only 5 lines, you can use synchronized block.

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

**Points to remember for Synchronized block**

* Synchronized block is used to lock an object for any shared resource.
* Scope of synchronized block is smaller than the method.

**Syntax to use synchronized block**

1. **synchronized** (object reference expression) {
2. //code block
3. }

Example of synchronized block

Let's see the simple example of synchronized block.

***Program of synchronized block***

1. **class** Table{
3. **void** printTable(**int** n){
4. **synchronized**(**this**){//synchronized block
5. **for**(**int** i=1;i<=5;i++){
6. System.out.println(n\*i);
7. **try**{
8. Thread.sleep(400);
9. }**catch**(Exception e){System.out.println(e);}
10. }
11. }
12. }//end of the method
13. }
15. **class** MyThread1 **extends** Thread{
16. Table t;
17. MyThread1(Table t){
18. **this**.t=t;
19. }
20. **public** **void** run(){
21. t.printTable(5);
22. }
24. }
25. **class** MyThread2 **extends** Thread{
26. Table t;
27. MyThread2(Table t){
28. **this**.t=t;
29. }
30. **public** **void** run(){
31. t.printTable(100);
32. }
33. }
35. **public** **class** TestSynchronizedBlock1{
36. **public** **static** **void** main(String args[]){
37. Table obj = **new** Table();//only one object
38. MyThread1 t1=**new** MyThread1(obj);
39. MyThread2 t2=**new** MyThread2(obj);
40. t1.start();
41. t2.start();
42. }
43. }

Output:5

10

15

20

25

100

200

300

400

500