**What is synchronized keyword?**

It is a keyword applicable only methods and blocks but not for class and variables.

If multiple threads are trying to operate simultaneously on the **same object** then there may be chance of data inconsistency problem. To overcome this problem we should go for synchronized keyword.

Adv: Data inconsistency problem can be resolved.

DisAdv: Increases waiting time of the thread and creates performance problem.

**What is the internal implementation if synchronization?**

Is implemented using lock. Every object in has unique lock, whenever we are using synchronized keyword then only lock concept will come into the picture.

**How the thread executes the synchronized block or method?**

If a thread wants to execute **synchronized** method on the given object, first it has to get lock of that object once thread got the lock then, it is allowed to execute any synchronized method on that object. Once method execution completes automatically thread release lock.

**Who will take care of acquiring and releasing a lock?**

Acquiring and releasing lock will be internally take care by JVM and programmer is not responsible for this activity.

While a thread executing synchronized method on the given object the remaining threads are not allowed to execute any synchronized method simultaneously on the same object.

Refer SynchronizationTest.java for more details

**What are the two level of synchronization?**

**Class level lock**

**Synchronized block**

**What is Class level lock?**

**Every class in java has a unique lock which is class level lock**

**When a Thread require class-level lock?**

If a thread wants to execute a static synchronized method, then thread require class level lock. Once thread got class-level lock then it is allowed to execute any static synchronized method if that class. Once method execution completes automatically the thread releases the lock.

**Consider I having a class with following methods**

**static synchronized m1()**

**static synchronized m2()**

**static m3()**

**synchronized m4()**

**m5()**

**How will the exections happens if t1, t2, t3 has to execute static synchronized methods, t4 to execute m3() method, t5 to execute m4() method, t6 to m5() method.**

**Answer in theory for class-level lock:** While a thread executing static synchronized method, the remaining threads are not allowed to execute any static synchronized method of that class simultaneously.

But remaining thread are allowed to execute the following method simultaneously

1. Normal static method

2. synchronized instance methods

3. Normal instance methods

**What is synchronized block?**

If very few lines of the code synchronization, then it is not recommended to declare entire method as synchroized. We have to enclose those few lines of the code, by using synchronized block.

Advantage: It reduces waiting time of the threads and improves performance of the system.

**How we declare synchronized block?**

We can declare synchronized block as follows

To get lock of current object:

**synchronized(this) {}**

If the thread got lock of current object then only it is allowed to execute this area.

Refer SynchronizedBlockTest.java for more details

**How we declare synchronized block to get lock of particular object?**

To get lock of particular object b

**synchronized(b) {}**

If the thread got lock of particular object `b` then only it is allowed to execute this area.

**How we declare get class level lock?**

To get a class level lock

**synchronized(Display.class) {}**

If the thread got lock of this class, then only it is allowed to execute this area.

Refer SynchronizedBlockTest.java for more details

**What will happen if I trying to get lock on primitive type?**

**Lock concept applicable for object types and class types but not for primitives, hence we can’t pass primitive type as arugment to synchronized block.** Otherwise we will get compile time error.

We will get unexpected type, found int required referece

int x = 10;

synchronized(x) {

//...

}

**FAQs:**

**What is synchronized keyword where we can apply?**

**Advantage/dis of synchronized keyword?**

**What is race condition? If multiple thread operating on same object and causing data-inconsistency**

**What is Object lock and when it is required?**

**What is class level lock and when it is required?**

**Whenever if a thread wants to execute a static synchronized method, this lock is required**

**Difference b/w object and class level lock?**

**Object – for instance method**

**Class – for static method**

**While a thread executing synchronized method on the given object is the remaining threads are allowed to execute any other synchronized method simultanously on the same object?**

**No**

**What is synchronized block?**

**How to synchronized block to get lock of current object?**

**Is the thread can acquire can acquire multiple lock simultanously? Yes, of-course from different objects**

**Class X {**

**public** **synchronized void m1() {**

**Y y = new Y();**

**synchronized(y) {**

**Z z = new Z();**

**synchronized(z) {**

**}**

**}**

**}**

**}**