How To Do In Java

Thread synchronization, object level locking and class level locking

Synchronization refers to multi-threading. A synchronized block of code can only be executed by one thread at a time.

Java supports multiple threads to be executed. This may cause two or more threads to access the same fields or objects. Synchronization is a process which keeps all concurrent threads in execution to be in synch. Synchronization avoids memory consistence errors caused due to inconsistent view of shared memory. When a method is declared as synchronized; the thread holds the monitor for that method's object If another thread is executing the synchronized method, your thread is blocked until that thread releases the monitor.

Synchronization in java is achieved using ***synchronized*** keyword. ***You can use synchronized keyword in your class on defined methods or blocks. Keyword can not be used with variables or attributes in class definition.***

Object level locking

Object level locking is mechanism when you want to synchronize a non-static method or non-static code block such that only one thread will be able to execute the code block on given instance of the class. This should always be done to make instance level data thread safe. This can be done as below :

```
public class DemoClass
{
    public synchronized void demoMethod(){}
}

or

public class DemoClass
{
    public void demoMethod(){
        synchronized (this)
        {
            //other thread safe code
        }
    }
}

or

public class DemoClass
```

```
{
    private final Object lock = new Object();
    public void demoMethod(){
        synchronized (lock)
        {
            //other thread safe code
        }
    }
}
```

Class level locking

Class level locking prevents multiple threads to enter in synchronized block in any of all available instances on runtime. This means if in runtime there are 100 instances of DemoClass, then only one thread will be able to execute demoMethod() in any one of instance at a time, and all other instances will be locked for other threads. This should always be done to make static data thread safe.

```
public class DemoClass
{
    public synchronized static void demoMethod(){}
}

or

public class DemoClass
{
    public void demoMethod(){
        synchronized (DemoClass.class)
        {
            //other thread safe code
        }
    }
}

or

public class DemoClass
{
    private final static Object lock = new Object();
    public void demoMethod(){
        synchronized (lock)
        {
            //other thread safe code
        }
    }
}
```

Some Important notes

1. Synchronization in java guarantees that no two threads can execute a synchronized method which requires same lock simultaneously or concurrently.
2. synchronized keyword can be used only with methods and code blocks. These methods or blocks can be static or non-static both.
3. When ever a thread enters into java synchronized method or block it acquires a lock and whenever it leaves java synchronized method or block it releases the lock. Lock is released even if thread leaves synchronized method after completion or due to any Error or Exception.
4. java synchronized keyword is re-entrant in nature it means if a java synchronized method calls another synchronized method which requires same lock then current thread which is holding lock can enter into that method without acquiring lock.
5. Java Synchronization will throw NullPointerException if object used in java synchronized block is null. For example, in above code sample if lock is initialized as null, the synchronized (lock) will throw NullPointerException.
6. Synchronized methods in Java put a performance cost on your application. So use synchronization when it is absolutely required. Also, consider using synchronized code blocks for synchronizing only critical section of your code.
7. It's possible that both static synchronized and non static synchronized method can run simultaneously or concurrently because they lock on different object.
8. According to the Java language specification you can not use java synchronized keyword with constructor it's illegal and result in compilation error.
9. Do not synchronize on non final field on synchronized block in Java. because reference of non final field may change any time and then different thread might synchronizing on different objects i.e. no synchronization at all. Best is to use String class, which is already immutable and declared final.

Happy Learning !!

Related Posts:

1. **Difference between sleep() and wait()?**
2. **How to Use Locks in Java | java.util.concurrent.locks.Lock Tutorial and Example**
3. **What is Thread Safety?**
4. **Writing a deadlock and resolving in java**
5. **Difference between “implements Runnable” and “extends Thread” in java**
6. **Non-blocking Thread-safe List – ConcurrentLinkedDeque Example**
7. **How to Work With wait(), notify() and notifyAll() in Java?**
8. **Core java interview questions series : Part 3**

LOCKING MULTI THREADING SYNCHRONIZED

39 THOUGHTS ON “THREAD SYNCHRONIZATION, OBJECT LEVEL LOCKING AND CLASS LEVEL LOCKING”

Suda

MARCH 30, 2015 AT 8:38 AM

Thanks

Amreesh

OCTOBER 23, 2014 AT 3:38 PM

```
==================================How to lock Object===================================
class A
{
synchronized void test1()
{
for(int i=0;i<100;i++)
{
System.out.println(i);
}
}
synchronized void test2()
{
for(int i=200;i<300;i++)
{
System.out.println(i);
}
```