Synchronization

dasturlash.uz

Synchronization 1

- Synchronization is a modifier applicable only for methods and blocks but not for classes and variables
- 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.
- Sinxronizatsiya - bu faqat Method va bloklar uchun qo'llaniladigan modifikator, lekin class va o'zgaruvchilar uchun emas
- Java-da sinxronizatsiya - bu har qanday umumiy manbaga bir nechta Thread larni kirishini boshqarish qobiliyati.
- Java Sinxronizatsiyasi eng yaxshi variant bo'lib, biz faqat bitta Thread ga umumiy manbaga kirishga ruxsat berishni xohlaymiz.

Synchronization 2

- If multiple thread are trying to operate simultaneously under same Java Object then there may be chance of data inconsistency problem.
- To overcome this problem we should use Synchronized keyword.
- Agar bir xil Java ob'ekti bilan bir vaqtning o'zida bir nechta Thread lar ishlashga harakat qilsa, data inconsistency problem(ma'lumotlarning nomuvofiqligi muammosi) bo'lishi mumkin.
- Ushbu muammoni hal qilish uchun biz Synchronized kalit so'zidan foydalanishimiz kerak

Synchronization 3

- If a method or block declared as Synchronized then at a time only one thread is allowed to execute that method or block under given Object. So data inconsistency problem will be resolved.
- The main advantage of synchronized keyword is we can resolve data inconsistency problem.
- But main disadvantage of synchronized keyword is it increases waiting time of thread and created performance problems.
- Agar method yoki blok Synchronized deb e'lon qilingan bo'lsa, bir vaqtning o'zida faqat bitta Thead ushbu ob'ekt ostida ushbu method yoki blokni ishlatishga ruxsat beriladi. Shu bilan ma'lumotlarning nomuvofiqligi muammosi hal qilinadi.
- Sinxronlashtirilgan kalit so'zning asosiy afzalligi shundaki, biz ma'lumotlarning nomuvofiqligi muammosini hal qila olamiz.
- Ammo sinxronlashtirilgan kalit so'zning asosiy kamchiligi shundaki, u Thread ni kutish vaqtini oshiradi va ishlash muammolarini keltirib chiqaradi

Lock

dasturlash.uz

Lock 1

- Internally synchronized concept is implemented by using lock.
- Every Object in java has a unique lock.
- When ever we are using synchronized key word then only lock concept will come to the picture.
- Aslida synchronized kontseptsiya lock yordamida amalga oshiriladi.
- Java-dagi har bir ob'ekt o'ziga xos qulfga ega.
- Agar biz synchronized kalit so'zdan foydalansak, ko’zimiz o’ngiga faqat lock tushunchasi keladi.

Lock 2

- If a thread wants to execute synchronized method under given object first it has to get lock of that object.
- Once a thread got a lock then it is allowed to execute any synchronized method on that object.
- Once method execution completes automatically thread releases lock.
- Requiring and releasing lock internally takes care by JVM
- Agar Thread berilgan ob'ekt ostida synchronized methodini ishlatmoqchi bo'lsa, avval u ushbu ob'ektning lock qilishi kerak.
- Thread ob’ectni lock qilganidan so'ng, ushbu ob'ektda har qanday synchronized methodini bajarishga ruxsat beriladi.
- Method bajarilishi tugagach, Thread avtomatik ravishda lock qo’yib yuboradi.
- O’bektni lock qilish va lock ni bo'shatish JVM tomonidan amalga oshiriladi

Lock 3

```
Class X{
        synchronized m1() {......}
        synchronized m2 () {......}
        m3(){......}
}
```

- While a thread executing synchronized method user given object the remaining threads not allowed to execute any synchronized method simultaneously under same object.
- But remaining threads allowed to execute non synchronized methods simultaneously.
- Agar Thread berilgan o'bekt uchun bironta synchronized metodini ishlatayotgan bo'lsa qolgan Thread lar shu ob'kectni boshqa synchronized methodlarini ishlata olmaydi.
- Ammo synchronized bo'lmagan methodlarini bir vaqtda birnechta thead bemalol ishlatishi mumkin.

dasturlash.uz

Lock 4

- Lock concept is implemented based on object but not based on method.
- Lock kontseptsiyasi ob'ekt asosida amalga oshiriladi, lekin method asosida emas.

dasturlash.uz

Class level lock

dasturlash.uz

Class level lock 1

- Class level in java has a unique lock which is nothing but class level lock.
- If a thread wants to execute a static synchronized method then Thread required class level lock.
- Once thread gets class level lock then it is allowed to execute any static synchronized method of that class.
- Once method executing completes automatically thread releases the lock.
- Java Class darajasida methodlarni quluflash class level lock deyiladi.
- Agar Thread statik synchronized methodni ishlatmoqchi bo'lsa, Thread class darajasidagi blokirovkani talab qiladi.
- Thrad class darajasidagi lock ni olganidan so'ng, u classdagi har qanday statik synchronized methodini ishlatishi mumkin.
- Methodni ishlatilganidan keyin Thread lock/quluf ni qo'yib yuboradi.

dasturlash.uz

Class level lock 2

- While a thread executing static synchronized method the remaining threads not allowed to execute any static synchronized method simultaneously of that class.
- But remaining threads are allowed to execute the following methods simultaneously:
    - normal instance methods
    - synchronized instance methods
    - normal static methods.
- Thread class ning bironta static synchronized methodini ishlatib turgan bo’lsa boshqa thread lar shu vaqtda, shu class ning boshqa static synchronized methodini ishlata olmaydi.
- Ammo boshqa Thread lar quyidagi methodlarni ishlatishi mumkin:
    - Mazgi tepada yozilgan.

dasturlash.uz

Class level lock exaemple

```
Class X{
          static synchronized m1() {......}
          static synchronized m2 () {......}
                      synchronized m3(){......}
                      synchronized m4(){......}
                                   m5(){......}
}
```

Synchronized block

Synchronized block 1

- If a very few lines of the code required a synchronization then it is not recommended to declare entire method as synchronized.
- We have to enclose those of few lines of code by using Synchronized block.
- The main advantage of synchronized block is it reduces waiting time of threads and improves performance of the system.

- Agar kodning juda oz qatorlari synchronization talab qilsa, butun methodni synchronized deb e'lon qilish tavsiya etilmaydi.
- synchronized blokdan foydalanib, biz bir nechta qator kod larni synchronized qilishimiz mumkin.
- Synchronized metodini asosiy afzalligi shundaki, u Thread larni kutish vaqtini qisqartiradi va tizimning ish faoliyatini yaxshilaydi.

Synchronized block implementation 1

- To get lock of current Object (hozirgi ob’ektni lock qilish)

```
synchronized(this){
....
}
```

- If a thread get lock of current object then only this thread allowed to execute this area.
- Agar thread hozirgi ob’ekt ni lock qilsa, faqat shu thead shu maydonni ishlatishi mumkin.
- Shu ob’ekt bo’ylab bir vatni o’zida bitta thread shu maydonni ishlatishi mumkin.

Synchronized block implementation 2

- To get lock of particular object (qaysidir ob’ekct lock qilish)

```
synchronized(someObject){
....
}
```

- If a Thread get lock of particular Object ‘someObject’ then only this thread allowed to execute this area.
- Agar Thread qaysidir ob’ekt (‘someObject’) ni lock qilsa, faqat shu thread shu maydonni ishlatishi mumkin.
- ‘SomeObject’ bo’ylab bir vatni o’zida bitta thread shu maydonni ishlatishi mumkin.

Synchronized block implementation 3

- To get class level lock (class darajasida lock qilish)

```
synchronized(Some.class){
....
}
```

- If a thread got class level lock of ‘Some’ class then only this thread allowed to execute this area.
- Agar Thread class ni lock qilsa, faqat shu thread bir vaqtni o’zida shu maydonni ishlatishi mumkin.

Volatile

- Volatile – o’zgaruvchan deb tarjima qilinadi.

Volatile – description 1

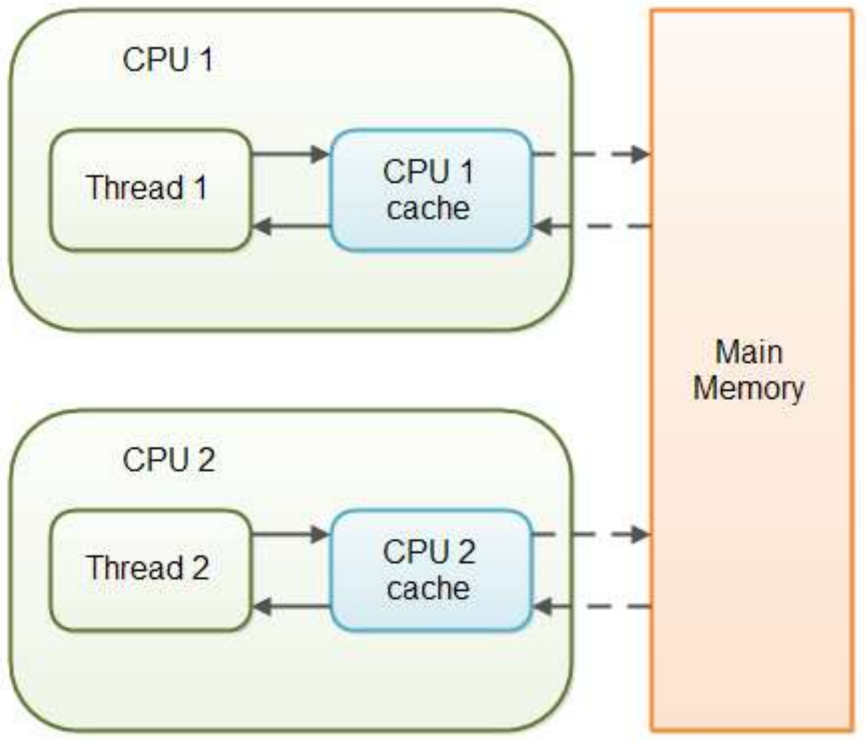
- The Java volatile keyword is used to mark a Java variable as "being stored in main memory".
- More precisely that means, that every read of a volatile variable will be read from the computer's main memory, and not from the CPU cache.
- Every write to a volatile variable will be written to main memory, and not just to the CPU cache.
- Java volatile kalit so'zi Java o'zgaruvchisini "asosiy xotirada saqlangan" deb belgilash uchun ishlatiladi.
- Aniq aytsam har ga o’zgzruvchini qiymati o’qilayotganda CPU ni cache dan(hotirasidan) emas, balki Kompyuterning asosiy xotirasi (RAM) dan bo’ladi.
- Hargal volatile o’zgaruvchining qiymati o’zgarsa u asosiy xotira (RAM) dan o’zgaridi.

Volatile – description 2

- The Java volatile keyword guarantees visibility of changes to variables across threads.
- Java volatile kalit so’zi o’zgaruvchining o’zgargan qiymadi boshqa Thread larda ham birdik ko’rinishini taminlaydi.

- In a multithreaded application where the threads operate on non-volatile variables, each thread may copy variables from main memory into a CPU cache while working on them, for performance reasons. If your computer contains more than one CPU, each thread may run on a different CPU. That means, that each thread may copy the variables into the CPU cache of different CPUs. This is illustrated in next page:

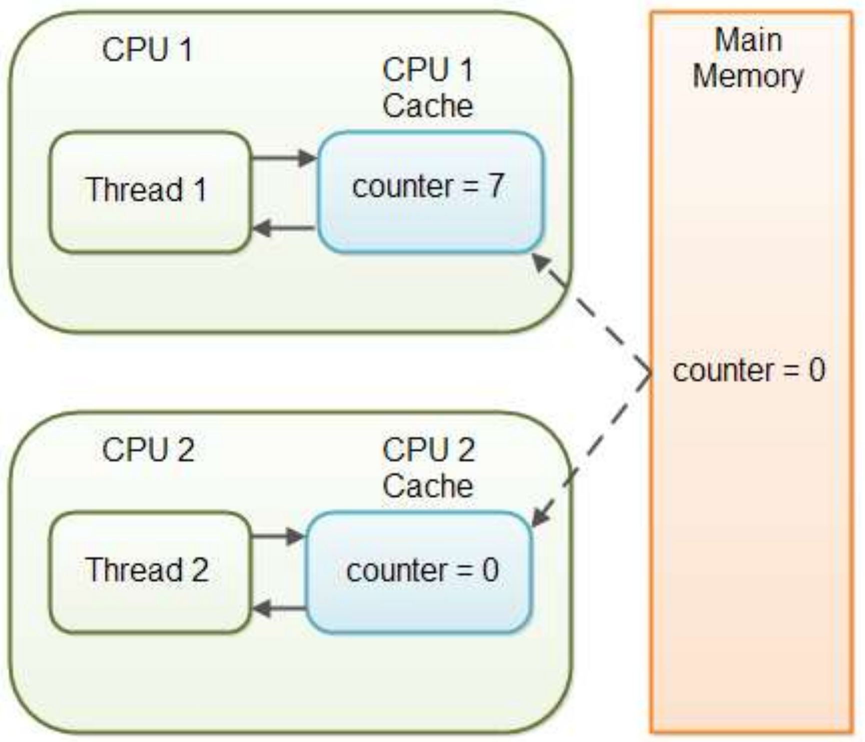
Variable without volatile 1



dasturlash.uz

Variable without volatile 2

- Imagine too, that only Thread 1 increments the counter variable, but both Thread 1 and Thread 2 may read the counter variable from time to time.
- If the counter variable is not declared volatile there is no guarantee about when the value of the counter variable is written from the CPU cache back to main memory. This means, that the counter variable value in the CPU cache may not be the same as in main memory. This situation is illustrated here:



Volatile example

```
public class SharedObject {
    public volatile int counter = 0;
}
```

- Declaring a variable volatile thus *guarantees the visibility* for other threads of writes to that variable.
- O’zgaruvchini volatile deb elon qilish, shu o’zgaruvchining oxirgi qiymati boshqa Thread larga ko’rinishini taminlaydi.

Volatile not enough

- In the scenario given above, where one thread (T1) modifies the counter, and another threads (T2,T3,T4) reads the counter (but never modifies it), declaring the counter variable volatile is enough to guarantee visibility for T2 of writes to the counter variable.
- If, however, both T1 and T2 were incrementing the counter variable, then declaring the counter variable volatile would not have been enough.
- Bitta holatni ko’raylik. T1- Thread qaysidir o’zgruvchini qiymatini o’zgartirsa ba boshqa T2,T3,T4… Thread lar o’zgaruvchini faqat qiymatini o’qisa volatile kalit so’zidan foydalansak bo’ladi. Bunda volatile kalit so’zi T2,T3,T4 Thread lar o’zgaruvchining ohirgi qiymatini o’qishini taminlaydi.
- Agar. T1 va T2 Thread lar bir vaqtni o’zida o’zgaruvchini qiymatini o’zgartirsa, o’zgaruvchini volatile kalit so’zi bilan belgilash foyda bermaydi.

Volatile not enough

- If multi thread changes value of any variable we should use either synchronized keyword or one of the many atomic data types found in the java.util.concurrent package.
- Agar birnechta Thread birvaqtni o’zida o’zgaruvchining qiymatini o’zgartirsa biz Synchronized kalit so’zini yokiy java.util.concurrent package **dagi data tiplar dan bittasini ishlatishimiz kerak.**

Volatile links

- https://jenkov.com/tutorials/java-concurrency/volatile.html
- https://www.javatpoint.com/volatile-keyword-in-java