Atomic

dasturlash.uz

Java Atomic

- In Java, **atomic variables** and **operations** used in concurrency.
- The shared entity such as objects and variables may be changed during the execution of the program. Hence, they may lead to inconsistency of the program.
- So, it is important to take care of the shared entity while accessing concurrently
- Java offers **two** solutions to overcome this problem:
  - By using lock and synchronization
  - By using atomic variable

- Javada atomic o'zgaruvchilar va operation lar paraller dasturlashda ishlatiladi.
- Dastur ishlash jarayonida umumiy bo'lgan o'zgaruvchi va ob'ektlar o'zgarishi mumkin. Demak, ular dasturning nomuvofiqligiga olib kelishi mumkin.
- Shunday qilib parallel dasturlashda umumiy o'zgaruvchilar ga ehtiyot bo'lish kerak.
- Javada bu muommoni 2ta yo'l bilan hal qilish mumkin.

Java Atomic Classes

- Java provides a **java.util.concurrent.atomic** package in which atomic classes are defined.
- The atomic classes provide a **lock-free** and **thread-safe** environment or programming on a single variable.
- It also supports atomic operations.
- All the atomic classes have the get() and set() methods that work on the volatile variable. The method works the same as read and writes on volatile variables.

- Java **java.util.concurrent.atomic package da atomic class larni taqdim etgan.**
- **Atomic class lari bitta o’zgaruvchi uchun lock-free (quluvlashdan ozon) va thread-save bo’lgan muhidni taqdim etadi.**
- Shuningdek, u atom operatsiyalarini qo'llab-quvvatlaydi.
- Barcha atomic class lar volatile o’zgaruvchi bilan get() va set() methodlar ni tagdim etadi.

dasturlash.uz

Java Atomic Classes

AtomicBoolean	It is used to update boolean value atomically.
AtomicInteger	It is used to update integer value atomically.
AtomicIntegerArray	An int array in which elements may be updated atomically.
AtomicIntegerFieldUpdater	A reflection-based utility that enables atomic updates to designated volatile int fields of designated classes.
AtomicLong	It is used to update long value atomically.
AtomicLongArray	A long array in which elements may be updated atomically.
AtomicLongFieldUpdater	A reflection-based utility that enables atomic updates to designated volatile long fields of designated classes.

Atomic Operations

- Those operations that always execute together is known as the **atomic operations** or **atomic action**.
- All the atomic operations either execute effectively happens all at once or it does not happen at all.

- Har doim birga bajariladigan operatsiyalar atom operatsiyalari yoki atom hodisalari deb ataladi.
- Barcha atom operatsiyalari bir vaqtning o'zida muofaqiyatli sodir bo'ladi yoki umuman bo'lmaydi.

Atomic Variable

- The atomic variable allows us to perform an atomic operation on a variable. Atomic variables minimize synchronization and help avoid memory consistency errors. Hence, it ensures synchronization.
- The atomic package provides the following five atomic variables:
  - AtomicInteger
  - AtomicLong
  - AtomicBoolean
  - AtomicIntegerArray
  - AtomicLongArray
- Atomic o’zgaruvchilar atomic amallarni o’zgaruvchilarda bajarish imkonini beradi. Atomic o’zgaruvchilar synchronization ni minimallashtiradi va xotira barqarorligi xatolaridan qochishga yordam beradi. Shunday qilib, u sinxronizatsiyani ta'minlaydi.
- Atomic package quyidagi 5ta atom o’zgaruvchilari bilan ishlashni taminlaydi.

dasturlash.uz

Atomic

- A multi-threaded environment may lead to unexpected output. The reason behind it that when two or more threads try to update the value at the same time then it may not update properly.
- Java offers **two** solutions to overcome this problem:
- By using lock and synchronization
- By using atomic variable

Atomic Example 1

```
public class MyThread extends Thread {
    private AtomicMainDemo atomicMainDemo;

    public MyThread(AtomicMainDemo atomicMainDemo) {
        this.atomicMainDemo = atomicMainDemo;
    }

    @Override
    public void run() {
        try {
            for (int i = 0; i < 100; i++) {
                Thread.sleep(10);
                atomicMainDemo.increase();
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

- Thread Class

dasturlash.uz

Atomic Example 2

- Main Class

```
public class AtomicMainDemo {
    public AtomicInteger atomicInteger;
    public static void main(String[] args) throws InterruptedException {
        new AtomicMainDemo().start();
    }

    public void start() throws InterruptedException {
        atomicInteger = new AtomicInteger(0);
        MyThread t1 = new MyThread(this);
        t1.start();
        MyThread t2 = new MyThread(this);
        t2.start();
        MyThread t3 = new MyThread(this);
        t3.start();

        t1.join(); t2.join();  t3.join();
        System.out.println(atomicInteger);
    }

    public void increase() { atomicInteger.addAndGet(1);  }
}
```

dasturlash.uz

Synchronized vs Volatile vs Atomic

Synchronized	Volatile	Atomic
1.It is applicable to only blocks or methods.	1.It is applicable to variables only.	1.It is also applicable to variables only.
2. Synchronized modifier is used to implement a lock-based concurrent algorithm, and i.e it suffers from the limitation of locking.	2.Whereas Volatile gives the power to implement a non-blocking algorithm that is more scalable.	2.Atomic also gives the power to implement the non-blocking algorithm.
3.Performance is relatively low compare to volatile and atomic keywords because of the acquisition and release of the lock.	3.Performance is relatively high compare to synchronized keyword.	3.Performance is relatively high compare to both volatile and synchronized keyword.
4.Because of its locking nature it is not immune to concurrency hazards such as deadlock and livelock.	4.Because of its non-locking nature it is immune to concurrency hazards such as deadlock and livelock.	4.Because of its non-locking nature it is immune to concurrency hazards such as deadlock and livelock.

dasturlash.uz

Atomic Links

- https://www.javatpoint.com/java-atomic
- https://www.baeldung.com/java-atomic-variables
- https://howtodoinjava.com/java/multi-threading/atomicinteger-example/
- https://www.geeksforgeeks.org/difference-between-atomic-volatile-and-synchronized-in-java/