Volatile

Volatile variable value will be directly written to and read from the main memory instead of CPU cache.

In Java, the volatile keyword is used to indicate that a variable's value will be modified by different threads. Declaring a variable as volatile ensures that any read or write operation on that variable will be directly performed on the main memory, rather than using a local copy (cache). This guarantees visibility of changes across threads.

[*https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/multithreading/Volatile.java*](https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/multithreading/Volatile.java)

*public class* VolatileTest {  
 *private boolean* flag = *false*;  
  
 *public void* writer() {  
 flag = *true*;  
 }  
  
 *public void* reader() {  
 *if* (flag) {  
 System.*out*.println("Flag is true");  
 }  
 *else* {  
 System.*out*.println("Flag is false");  
 }  
 }  
  
 *public static void* main(String[] args) *throws* InterruptedException {  
 VolatileTest sharedData = *new* VolatileTest();  
  
 Thread writerThread = *new* Thread(() -> {  
 *try* {  
 Thread.*sleep*(1);  
 } *catch* (InterruptedException e) {  
 *throw new* RuntimeException(e);  
 }  
 sharedData.writer();  
 });  
  
 Thread readerThread = *new* Thread(() -> {  
 sharedData.reader();  
 });  
 writerThread.start();  
 readerThread.start();  
  
 }  
}

Flag is false

*public class* VolatileTest {  
 *private volatile boolean* flag = *false*;  
  
 *public void* writer() {  
 flag = *true*;  
 }  
  
 *public void* reader() {  
 *if* (flag) {  
 System.*out*.println("Flag is true");  
 }  
 *else* {  
 System.*out*.println("Flag is false");  
 }  
 }  
  
 *public static void* main(String[] args) *throws* InterruptedException {  
 VolatileTest sharedData = *new* VolatileTest();  
  
 Thread writerThread = *new* Thread(() -> {  
 sharedData.writer();  
 });  
  
 Thread readerThread = *new* Thread(() -> {  
 sharedData.reader();  
 });  
 writerThread.start();  
 readerThread.start();  
  
 }  
}

Flag is true











