**Volatile:**

Declaring a volatile Java variable means:

* The value of this variable will **never be cached thread-locally**: all reads and writes will go straight to "main memory"

Note: A [common bug with volatile](https://www.javamex.com/tutorials/synchronization_volatile_dangers.shtml) is to assume that x++ or x = x+1 is atomic. For cases where you need this functionality, consider [AtomicInteger](https://www.javamex.com/tutorials/synchronization_concurrency_7_atomic_integer_long.shtml) or related classes.

### [Atomic operation](http://www.vogella.com/tutorials/JavaConcurrency/article.html#atomic-operation):

The Java language specification guarantees that reading or writing a variable is an atomic operation (unless the variable is of type long or double). Operations variables of type long or double are only atomic if they declared with the volatile keyword.

**Asynchronous vs Synchronous:**

Let’s say thread1 execute something synchronously, thread1 wait for it to finish before moving on to another task.

But if thread1 execute task asynchronously, thread1 can move on to another task before it finishes.