Java Memory Model

- Communicating between threads
- Multi-threading bugs
- Memory model
- Synchronization
- Interthread signals
- Examples

Communicating between threads

Threads share Heap memory.

Threads’ objects can have references to shared objects.

So, one thread can write to the shared object and another can read from it.

Multi-threading bugs

Thread Interference

Interference happens when two operations, running in different threads, but acting on the same data, interleave. This means that the two operations consist of multiple steps, and the sequences of steps overlap.

Memory Consistency Errors

Memory consistency errors occur when different threads have inconsistent views of what should be the same data.

Java Memory Model

The Java Memory Model describes what behaviors are legal in multithreaded code, and how threads may interact through memory.

It describes the relationship between variables in a program and the low-level details of storing and retrieving them to and from memory or registers in a real computer system.

It does this in a way that can be implemented correctly using a wide variety of hardware and a wide variety of compiler optimizations.

Java includes several language constructs, including

- volatile,
- final,
- and synchronized,

which are intended to help the programmer describe a program's concurrency requirements to the compiler.

volatile

The Java volatile keyword guarantees visibility of changes to variables across threads.

final

If variable can’t be changed, it is safe to use.

synchronized

The synchronized key word marks a method as a Critical Section.

Interthread signals

lock object

Each object in Java can work as lock object.

If a thread calls wait() of an object it stops and wait till another thread calls notify() of the same object.