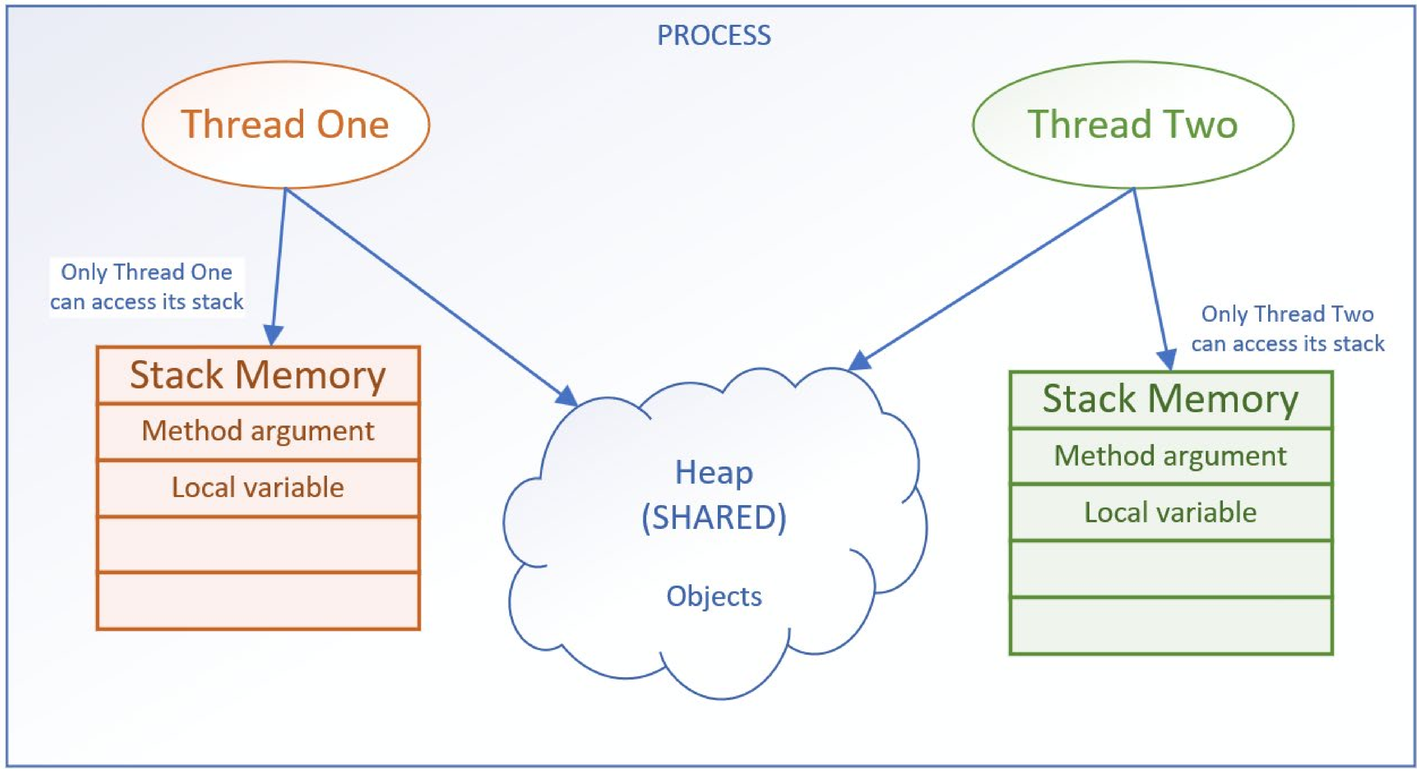
Threads accessing memory

Each thread has its own stack for local variables and method calls.

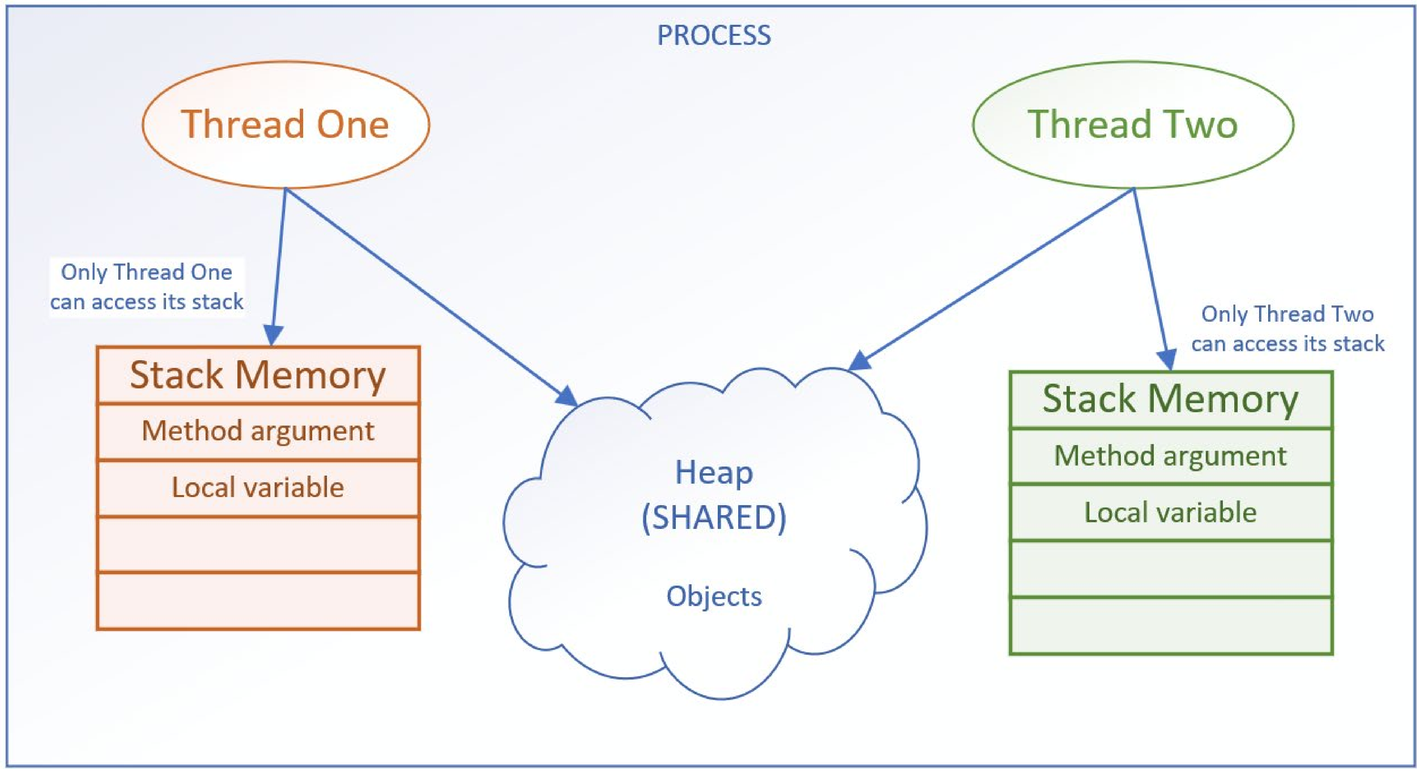
One thread doesn't have access to another thread's stack.



Threads accessing memory

Every concurrent thread additionally has access to the process memory, or the heap.

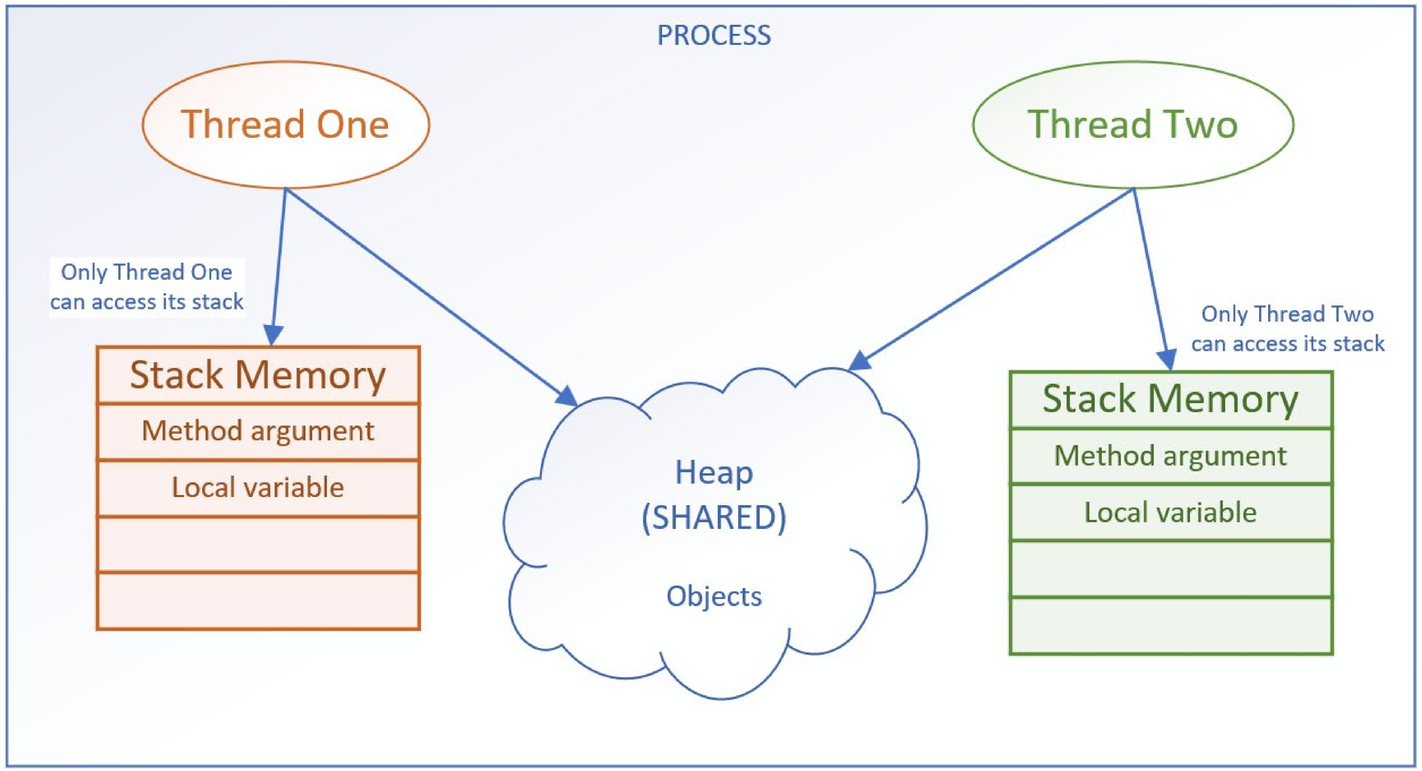
This is where objects and their data reside.



Threads accessing memory

This shared memory space allows all threads, to read and modify the same objects.

When one thread changes an object on the heap, these changes are visible to other threads.



Time Slicing

Time slicing is also known as time-sharing or time division.

It's a technique used in multitasking operating systems, to allow multiple threads or processes to share a single CPU for execution.

Available CPU time is sliced into small time intervals, which are divvied out to the threads.

Each thread gets that interval, to attempt to make some progress, on the tasks it has to do.

Whether it completes its task or not, in that time slice, doesn't matter to the thread management system.

When the time is up, it has to yield to another thread, and wait until its turn again.

Unfortunately, when your threads are sharing heap memory, things can change during that wait.

The Java Memory Model (JMM)

The Java Memory Model, is a **specification** that defines some rules and behaviors for threads, to help control and manage shared access to data, and operations.

- **Atomicity of Operations**. Few operations are truly atomic.
- **Synchronization** is the process of controlling threads' access to shared resources.