## Monitor:

- Is known as synchronization between threads. AKA threads which safe.
- a **monitor** is a synchronization construct that allows threads to have both mutual exclusion and the ability to wait (block) for a certain condition to become true. Monitors also have a mechanism for signalling other threads that their condition has been met
- However, a thread can actually suspend itself inside a monitor and then wait for an event to occur. If this happens, then another thread is given the opportunity to enter the monitor. The thread that was suspended will eventually be notified that the event it was waiting for has now occurred, which means it can wake up and reacquire the lock.

## Mutexes:

- AKA mutual exclusion are very similar to binary semaphores which have a value of 0 or 1.
- a **mutex** is a program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously
- For example, when the operating system's lock library is used and a thread tries to acquire an already acquired lock, the operating system could suspend the thread using a context switch and swap it out with another thread that is ready to be run, or could put that processor into a low power state if there is no other thread that can be run
- Many forms of mutual exclusion have side-effects. For example, classic semaphores permit deadlocks, in which one process gets a semaphore, another process gets a second semaphore, and then both wait forever for the other semaphore to be released.

References:


- https://en.wikipedia.org/wiki/Monitor_(synchronization)
- https://en.wikipedia.org/wiki/Mutual_exclusion