MONITORS

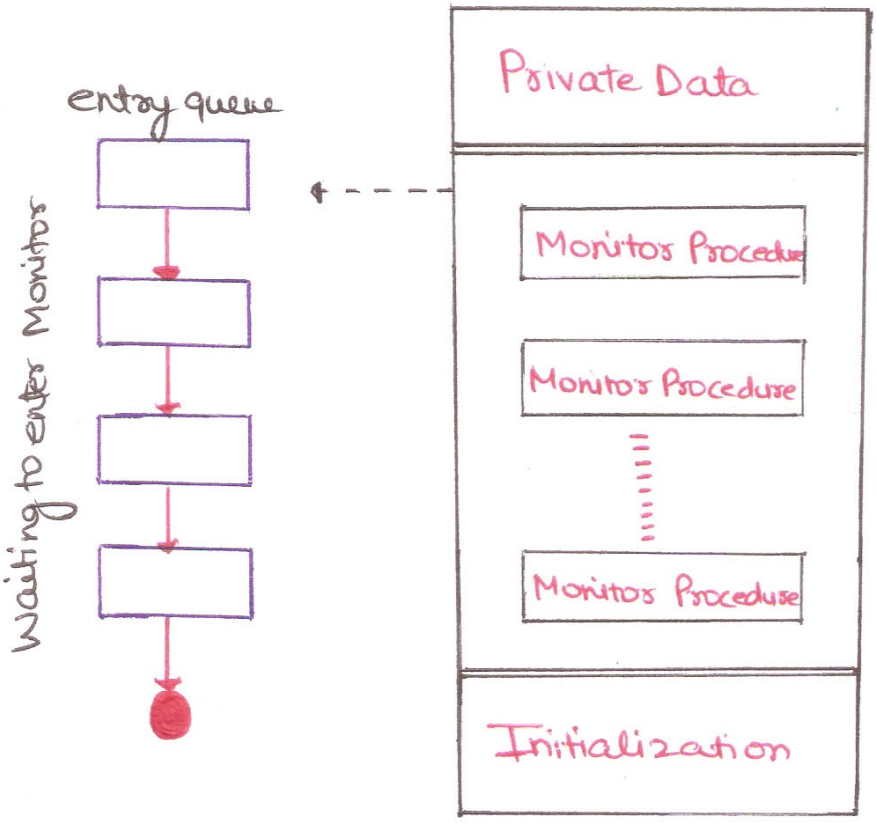
A monitor is a programming Language Construct that Controls access to shared data. OR

Monitor is a module that encapsulates

- 'Shared Data Structure'
- 'procedures' that operate on the shared data structure
- Synchronization between Concurrent procedure invocations

A monitor is same as a class type: like objects of a class are created, the variables of monitor types are defined.

Critical Regions are written as procedures and encapsulated together in a Single Module.



```
type <monitor type> = monitor
  ... data declaration
  ....
  monitor entry <name and its para>
{
  ....
}
```

The Synchronization among the processes is provided through the monitor entry procedures.

** Wait and Signal introduced in Semaphores are also implemented in monitors through the use of 'Condition' Variables.

Condition Variables are different from normal Variables because each of them has an associated queue.

A process calling wait on a particular Condition is put into the queue associated with that Condition Variable.

It means that the process is waiting to enter a CS guarded by the monitor.

A process calling the Signal causes the waiting process in the queue to enter the monitor.

These all variables are declared as:-

Condition <name of Variable>

wait < Condition Variable >

Signal< Condition Variable>

To force a process to exit immediately after the Signal operation, Signal-and-exit monitor is used.

If a process needs to be inside the monitor for some more time after signaling, then Signal-and-Continue monitor is used.

It means that the signaling has been done by the process, but it still maintains a lock on the Semaphore.

Java programming language implements the Signal-and-Continue monitor.