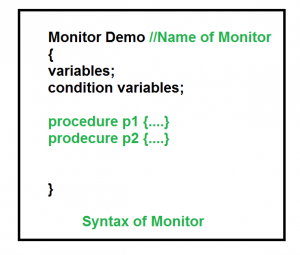
# Monitors in Process Synchronization:

The monitor is one of the ways to achieve Process synchronization. The monitor is supported by programming languages to achieve mutual exclusion between processes. For example Java Synchronized methods. Java provides wait() and notify() constructs.

1. It is the collection of condition variables and procedures combined together in a special kind of module or a package.
2. The processes running outside the monitor can’t access the internal variable of the monitor but can call procedures of the monitor.
3. Only one process at a time can execute code inside monitors.

**Syntax:**



**Condition Variables:**   
Two different operations are performed on the condition variables of the monitor.

Wait.

signal.

let say we have 2 condition variables  
**condition x, y; // Declaring variable**

**Wait operation**  
x.wait() : Process performing wait operation on any condition variable are suspended. The suspended processes are placed in block queue of that condition variable.

**Note:** Each condition variable has its unique block queue.

**Signal operation**  
x.signal(): When a process performs signal operation on condition variable, one of the blocked processes is given chance.

If (x block queue empty)

// ignore signal

else

// Resume a process from block queue.

**Advantages of Monitor:**  
Monitors have the advantage of making parallel programming easier and less error prone than using techniques such as semaphore.

**Disadvantages of Monitor:**  
Monitors have to be implemented as part of the programming language . The compiler must generate code for them. This gives the compiler the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes. Some languages that do support monitors are Java,C#,Visual Basic,Ada and concurrent Euclid.