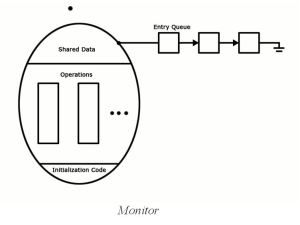
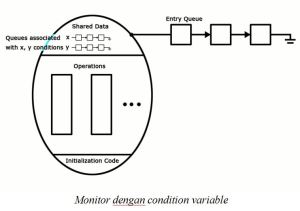
**Monitor**  
Monitor adalah suatu tipe abstrak yang memiliki tugas untuk mengatur aktivitas serta managemen resource oleh beberapa thread. Monitor pertama kali dikenalkan pada tahun 1970. Monitor terdiri atas data-data private dengan fungsi public yang dapat mengakses data tersebut. Method yang terdapat dalam suatu monitor sudah dibuat sedemikian rupa agar hanya ada satu method yang dapat bekerja dalam suatu waktu. Hal ini dilakukan untuk menjaga agar semua operasi dalam monitor bersifat mutual exclusion.

[](https://fatzandfriends.files.wordpress.com/2014/09/sisop3-1.jpg)

Monitor memiliki beberapa status, yaitu  
**1. Enter :** Kondisi dimana thread memasuki monitor  
**2. Acquire :** Kondisi dimana thread memasuki ruang kontrol  
**3. Owning :** Kondisi dimana thread menyelesaikan tugasnya dengan membagi resource yang berada di ruang kontrol  
**4. Release :** Kondisi dimana thread belum menyelesaikan tugasnya namun waktunya sudah habis, maka thread melepaskan kendali atas monitor  
**5. Waiting queue :** Tempat thread menunggu thread lain selesai  
**6. Release and exit :** Kondisi dimana thread sudah menyelesaikan tugasnya dan keluar dari monitor

Karena masalah sinkronisasi begitu rumit dan beragam, monitor menyediakan tipe data condition untuk programmer yang ingin menerapkan sinkronisasi yang sesuai untuk masalah yang dihadapinya. Condition memiliki operasi-operasi:  
1. Wait, sesuai namanya thread yang memanggil fungsi ini akan dihentikan kerjanya.  
2. Signal, jika suatu thread memanggil fungsi ini, satu (dari beberapa) thread yang sedang menunggu akan dibangunkan untuk bekerja kembali. Operasi ini hanya membangunkan tepat satu buah thread yang sedang menunggu. Jika tidak ada thread yang sedang menunggu, tidak akan terjadi apa-apa (bedakan dengan operasi buka pada semaphore).

[](https://fatzandfriends.files.wordpress.com/2014/09/sisop3-2.jpg)

Monitor juga dapat disebut sebagai mini-OS, karena konsep monitor sangat mirip dengan sebuah sistem operasi. Satu dapat mempertimbangkan inisialisasi sebagaimana data yang diinisialisasi ketika sistem boot up, data pribadi dan kode sebagai struktur data internal dan fungsi dari sebuah sistem operasi, dan prosedur monitor sebagai panggilan sistem. Program-program, tentu saja, benang yang membuat permintaan layanan. Oleh karena itu, monitor bisa dianggap sebagai mini-OS dengan layanan terbatas.

Kelebihan monitor  
1. Dapat mengatasi sinkronisasi tanpa perlu melibatkan programmer.  
2. Kompilator pada bahasa pemrograman yang telah mengimplementasikan monitor akan memastikan bahwa resource yang dapat diakses oleh beberapa thread dilindungi oleh monitor, sehingga prinsip mutual exclusion tetap terjaga.  
3. Kompilator bisa memeriksa kemungkinan adanya deadlock.  
4. Pengecekan error yang berhubungan dengan mutex dapat dilakukan secara terpusat hanya pada modul monitor saja sehingga lebih efisien.  
5. Ketika program monitor sudah benar tanpa error maka akses terhadap critical resource oleh berbagai macam proses akan selalu benar.

Kelemahan monitor  
1. Bila setiap signal csignal (fungsi sinkronisasi dalam monitor) selalu hilang maka proses yang ada dalam antrian juga akan di blok.  
2. Kurang efisien dalam mengembalikan proses yang sudah di blok.  
3. Memerlukan mekanisme penjadwalan proses yang benar-benar handal karena pengaturan proses dalam monitor yang sangat ketat.

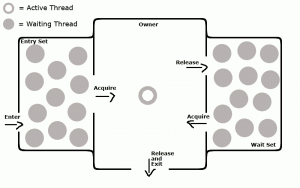
Bentuk dasar monitor:

**monitor example**  
**i : integer;**  
**c : condition;**

**Procedure producer(x);**  
**.**  
**.**  
**End;**

**Procedure consumer(x);**  
**.**  
**.**  
**End;**  
**End monitor;**

Contoh penggunaan monitor pada Java Virtual Machine (JVM), Java Virtual Machine memakai monitor untuk sinkronisasi thread.

[](https://fatzandfriends.files.wordpress.com/2014/09/sisop3-3.gif)

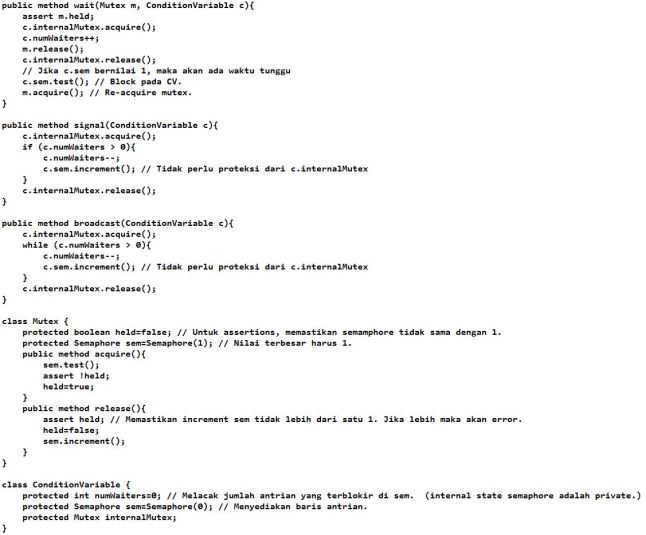
**Synchronized Statements**. Untuk membuat sebuah synchronized statement, gunakan keyword synchronized dengan ekspresi yang me-refer ke suatu object contoh:

***void reverseOrder()***  
***{***  
***synchronized(this)***  
***{***  
***/\*ekspresi method\*/***  
***}***  
***}***

**Synchronized Methods**. untuk membuat suatu synchronized methods, cukup dengan menambahkan keyword synchronized di depan nama method tersebut pada deklarasinya contoh:

**synchronized void reverseOrder()**  
**{**  
**/\*ekspresi method\*/**  
**}**

Implementasi monitor dengan semaphore:

[](https://fatzandfriends.files.wordpress.com/2014/09/sisop3.jpg)