

# Process Synchronization – Monitor

## ❖ Monitor 필요성 및 구현

- Semaphore를 사용하면, 실수하는 경우가 많아진다.
- 따라서, Monitor 객체를 만들어서 공유변수에 접근할 때 Monitor의 Method를 통해 공유변수에 접근
- 쉽게 Class를 생각하면 된다.

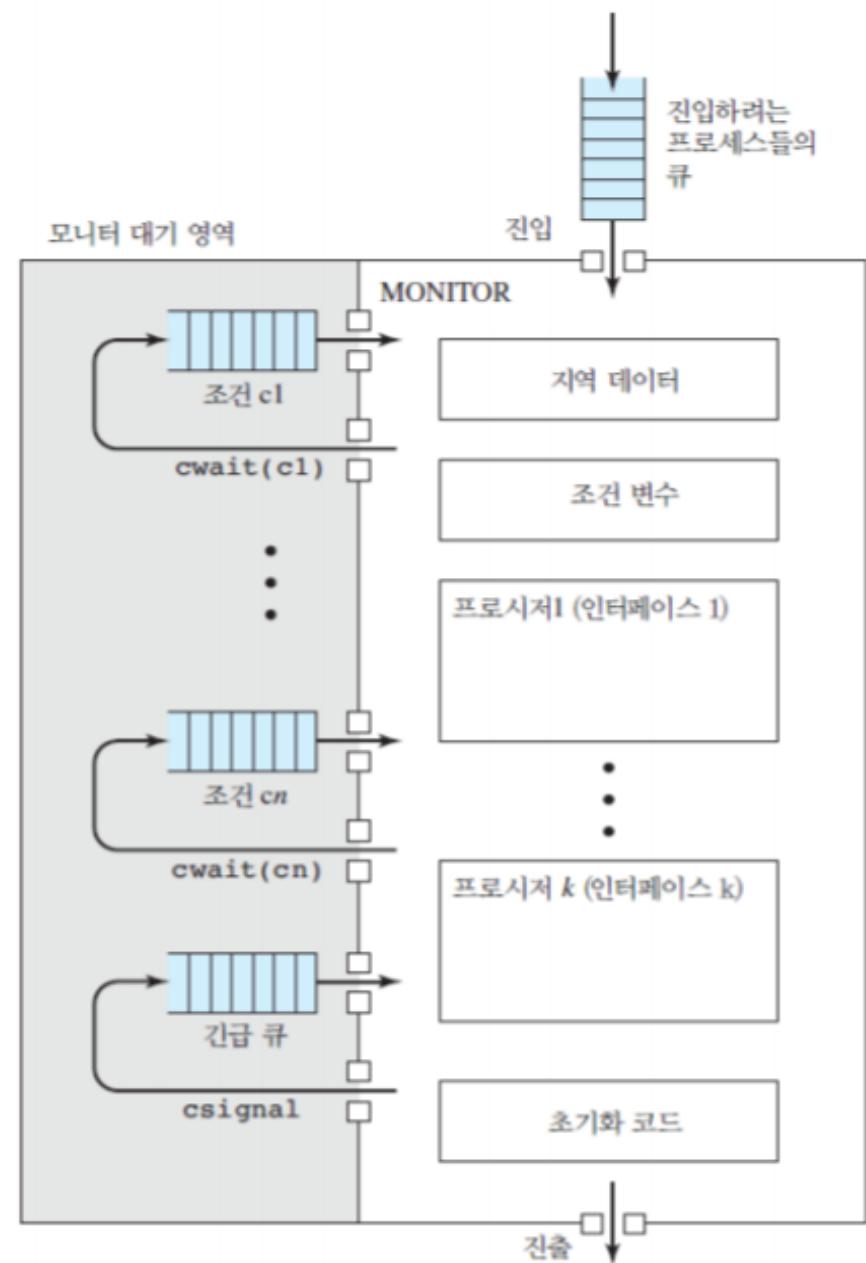


그림 5.15 모니터 구조

# Process Synchronization – Monitor

```
P(monitor_lock);  
    Body of the Function;  
if (sig_lock_cnt > 0) {  
    V(sig_lock);  
} else {  
    V(monitor_lock);  
}
```

조건 변수의 Waiting Queue 진입  
Signal 방의 Thread를 깨운다.  
없으면, Main 방 문을 열어 둔다.

```
/* x.signal */  
if (x_cnt > 0) {  
    sig_lock_cnt++;  
    V(x_lock);  
    P(sig_lock);  
    sig_lock_cnt--;  
}
```

monitor ResourceManager  
boolean busy;  
condition x; 조건 변수

```
procedure acquire()  
begin  
    if (busy) then x.wait();  
    busy = true;  
end
```

```
procedure release()  
begin  
    busy = false;  
    x.signal();
```

busy = false;  
end monitor

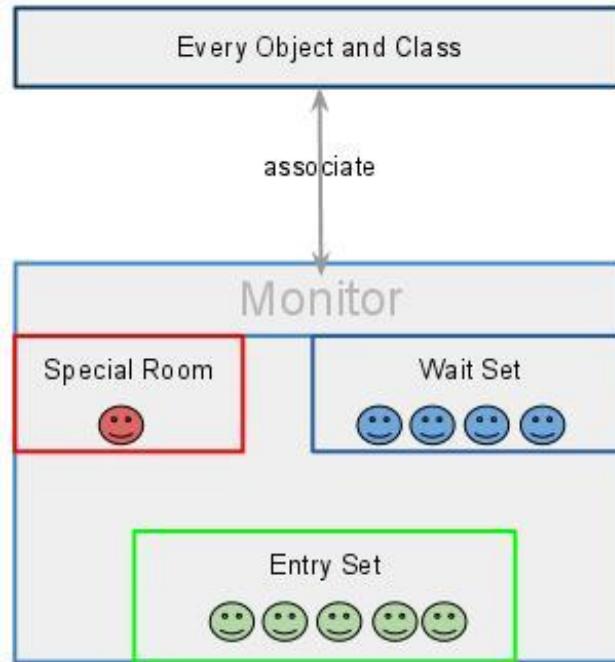
```
/* x.wait */  
x_cnt++;  
if (sig_lock_cnt > 0) {  
    V(sig_lock);  
} else {  
    V(monitor_lock);  
}  
P(x_lock);  
x_cnt--;
```

조건 변수의 Waiting Queue 진입  
Signal 방의 Thread를 깨운다.  
없으면, Main 방 문을 열어 둔다.

조건 변수의 Waiting Queue안에 있는 Thread 깨운다.  
본인은 Signal 대기실에 들어가 있다.

# Java Synchronization

☞ Java에서는 Monitor를 어떻게 구현 되어있는 것이지?



♂ Java의 “Synchronized” 가 어떻게 monitor를 사용하는 거지?

## 내 착각

Monitor 객체 내부에 Method가 존재해야 한다고 생각했다.

그러면 java의 “Synchronized”가 있는 객체가 monitor가 되는 건  
가?

## 내 결론

우선 Java에서 Synchronized 영역에 진입할 때 Lock을 획득하고 Critical Section에 벗어날 때, Lock을 반납한다.

Object의 Header에는 Monitor 객체를 참조하는 pointer가 존재한다.

이 Monitor 객체의 Lock과 Conditional Variable 역할로만 사용해도 충분히 Monior 객체로서 역할 수행이 가능하다!!