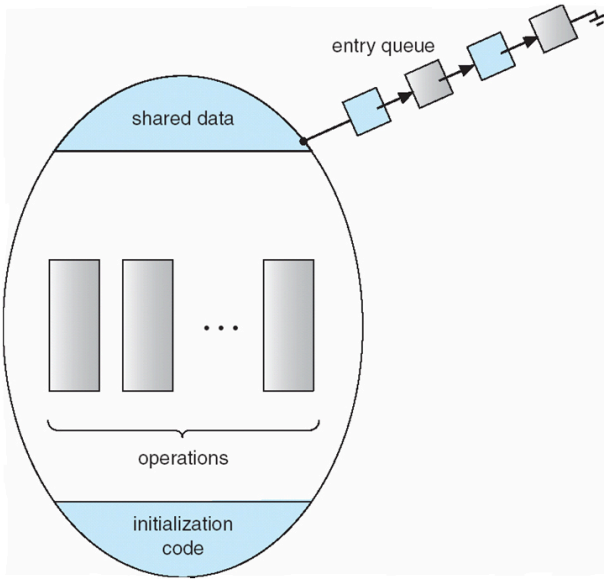
Monitors



Producer/Consumer with Monitors

```
class Queue {

  private ...; // queue data

  public synchronized void add(Object item) {
    put item on queue;
  }

  public synchronized Object remove() {
    remove and return item;
  }

}
```

Producer/Consumer with Monitors

```
class Queue {

  private ...; // queue data

  public synchronized void add(Object item) {
    put item on queue;
    this.notify(); // wake up waiting thread (aka Signal)
  }

  public synchronized Object remove() {
    while queue is empty {
      this.wait(); // give up lock and sleep
    }
    remove and return item;
  }

}
```


C++ Style Monitors

```
class Queue {

  public:
    add();
    remove();

  private:
    Lock lock;
    ConditionalVariable cv;
    Queue data;

}
```

```
Queue::add(item) {
  lock->acquire();

  put item on queue;
  cv->signal();

  lock->release();
}
```

```
Queue::remove()
  lock->acquire();

  while queue is empty {
    cv->wait(lock); // release lock & sleep
  }
  remove item from queue;

  lock->release();
  return item;
}
```

Semaphore using a Monitor

```
Semaphore(int initialVal) {

    int counter = initialVal;
    Lock lock;
    ConditionVariable notZero;

}
```

```
Semaphore::Wait() {
    lock.acquire();

    while (counter == 0) {
        notZero.wait();
    }
    counter--;

    lock.release();
}
```

```
Semaphore::Signal() {
    lock.acquire();

    counter++;
    notZero.signal();

    lock.release();
}
```


Monitor using Semaphores

```
class Monitor {

  public:
    void cvWait();
    void cvSignal();
    <monitor methods>;

  private:
    <shared data>;

    Semaphore lock = 1;
    Semaphore suspend = 0;
    Semaphore cv = 0;

    int numSuspended = 0;
    int numWaiting = 0;

}
```

```
Monitor::cvWait() {
  numWaiting++;
  if (numSuspended > 0)
    suspend.signal();
  else
    lock.signal();
  cv.wait();
  numWaiting--;
}
```

```
Monitor::cvSignal() {
  if (numWaiting > 0) {
    numSuspended++;
    cv.signal();
    suspend.wait();
    numSuspended--;
  }
}
```

```
Monitor::someMethod() {
  lock.wait();
  // .. do something ..
  // may use cvWait or cvSignal
  if (numSuspended > 0)
    suspend.signal();
  else
    lock.signal();
}
```