Goals

• Requirements for condition synchronization
• The monitor model
• How to implement the monitor model in Java
Condition Synchronization

Mutual exclusion is insufficient to guarantee that operations only happen when the system is in a proper “state” for the operation to be done without loss of safety

A condition on the state of the system must be associated with the operation in such a way that the operation is only attempted when the condition is true
Condition Synchronization

Examples:

• Producer-Consumer problem
  – produce only when buffer is not full
  – consume only when buffer is not empty

• Readers-Writer problem
  – readers may read if the only other operation in progress is another read operation
  – writers may write if there is no other operation in progress of either kind

Evaluating the condition requires that the state of the system not change during the condition’s evaluation ==> mutual exclusion during the evaluation
Hoare’s Monitor Model

acquire
lock
release

monitor
procedure/method

return

call
Hoare’s Monitor Model

- **call**
- **lock**
- **acquire**
- **release**
- **condition queue**
- **C.wait()**
Hoare’s Monitor Model

Lock

call

C.wait( )

return

C.signal( )

C-queue

urgent-queue

notify
Hoare’s Monitor Model

monitor BoundedCounter
begin
  condition belowMax, aboveMin;
  integer value;
  constant MAX=100, MIN=0;
  void value() begin return value; end
  void inc()
  begin
    if (value == MAX) belowMax.wait();
    value = value + 1;
    if (value == MIN+1) aboveMin.signal();
    return;
  end
  void dec()
  begin
    if(value == MIN) aboveMin.wait();
    value = value - 1;
    if(value == MAX-1) belowMax.signal();
    return;
  end
  begin
    value = MIN;
  end
end