

# ITU-VOOP-E2001

## Trådprogrammering – 2

Lars Thorup

[lars.thorup@acm.org](mailto:lars.thorup@acm.org), Nov 04, 2001

1

## Dagens overblik

- Opsamling
- Kommunikation mellem tråde
- Deadlock
- Til næste gang

2

## Opsamling

- Tråde
- Synkronisering
- Netværkslyttetråd

3

## Inter-thread kommunikation

- Starting and stopping another thread
- Giving information to another thread
  - wait & notify
  - send & receive
  - worker-thread
  - command queue
  - join

4

## Wait & Notify – 1

- Blocking current thread by calling
  - <monitor>.wait()
- Waking up another thread by calling
  - <monitor>.notify()
- Thread must hold lock when calling wait or notify
- Example in WaitNotifyTest.java

5

## Wait & Notify – example

- Thread 1:
  - Holding lock,
  - calling wait(),
  - giving up lock,
  - blocks....
  - 
  - 
  - is waken up,
  - gets lock back,
  - returns from wait()
- Thread 2:
  - 
  - 
  - Holding lock
  - calling notify(),
  - returning from notify(),
  - doing something else...,
  - eventually unlocking,
  - wake up other thread
  - 
  -

6

## send & receive

- Blocking current thread with Socket.receive()
- Waking up another thread with Socket.send()
- Not dependent on a monitor
- Can transfer information in the data being sent
- Works also between different programs

7

## Worker-thread

- Provide all information before thread starts
- Example: print job

8

## Command Queues

- What I call the Agent Pattern
- An agent has its own life (= its own thread)
- You can stimulate it and get responses
  - asynchronously
  - Sending a stimulus does not wait for a response
- See Agent.java for an example

9

## Command Queues

- Uses Command Pattern internally
- Uses Observer Pattern for responses
- Could use a Thread pool for additional parallelism

10

## join

- Wait until another thread is finished
- Call `Thread(Control).join()`

11

## Testing multithreaded code

- Test concurrent behaviour
  - Starting things
  - Testing that they have not finished yet
  - Waiting until they must have completed
  - Testing that they have completed

12

## Deadlock – eksempel

- Synchronize på to objekter i hver sin rækkefølge
  - Thread1            Thread2
  - synchronize(a) {    synchronize(b) {
  - synchronize(b) {    synchronize(a) {

13

## Deadlock – issues

- Not normally detected by the runtime system
- Often not detectable because it only shows under certain timing constraints
- Avoid by design

14

## Deadlock – solution

- Hierarchical locking
- Either statically order locks in hierarchy
  - Any attempt to lock non-hierarchically should then throw an exception

15

## Deadlock – solution

- Hierarchical locking
- Or dynamically detect non-hierarchy in lock access pattern
  - When locking C and already holding lock on A and B, draw edge from A→C and B→C
  - Does not allow A, B, C and A, C, B even though the cycle is protected by A

16

## Testing multithreaded code

- Test for no deadlock
  - Running the same testsuites
    - repeatedly, many iterations
    - with different pauses
  - Running automated tests
    - deadlock would hopefully show up time and again

17

## Til næste gang

- Implementer lyttråde og netværkskommunikation i tegnekonferencesystemet
  - så klient og server kan køre separat
- Læs til næste gang:
  - Artiklerne
    - Performance Features and Tools
    - Performance Analysis
- Forslag til emne sidste forelæsning

18