Welcome to

02152

Concurrent Systems

Fall Semester 2008

Hans Henrik Løvengreen
Robin Sharp

Informatics and Mathematical Modelling

Concurrent Systems Aims

After the course you should

- Understand *concepts* and *notions* of concurrency and networking
- Know *abstract models* of concurrency and principles of *verification*
- Be well versed in synchronization and communication *mechanisms*
- Know about underlying *implementation principles*
- Be aware of concurrency *pitfalls* and principles for avoiding them
- Know how to *test* concurrent programs
- Be skilled in writing *multi-threaded Java* programs
- Be skillled in use of TCP/IP through Java *socket programming*
- Know about various internet application *protocols*
- Know about network *middleware*
- Know a number of concurrency *SW-architectures*

Concurrent Systems Prerequisites

Required

- Good command of *sequential Java*
- Good knowledge af *algorithms* and *data structures*
- Discrete mathematics, including *predicate logic* ($\forall$, $\Rightarrow$)
- Basic knowledge of *C-programming*

Useful

- Basic knowledge of *machine architecture*
- Basic knowledge of *program representation*
- Knowledge of *databases*
- Knowledge of *automata* (state machines)
- Knowledge of *program semantics*

Concurrent Systems Material

- G. Andrews: *Foundations of Multithreaded, Parallel, and Distributed Programming*. Textbook
- H.H. Løvengreen: *Basic Concurrency Theory*. Note (**50 Kr.**)
- R. Sharp: *The Poor Man's Guide to Computer Networks and their Applications*. Note
- Other notes and auxiliary material will be available online:

www.imm.dtu.dk/courses/02152

Concurrent Systems Evaluation Fall 2008

Mandatory Assignments ($\sim$ 50 %)

- Test ability to apply concepts and notions in practice
-

Assignment no.	Set	Due
1	Wednesday, Sep 23	Thursday, Oct 23, 12.45
2	Monday, Nov 10	Thursday, Dec 4, 12.45

- Carried out in *groups of 2-3 students*
- Must be documented by *reports*
- *Hard deadlines*

Written Exam ($\sim$ 50 %)

- Test understanding of concepts and notions
- Wednesday December 10
- 4 hours, all (non-electronic) aids allowed

Concurrent Programming

Why?

1. Because the real world is *parallel*:

   many users, many devices $\Rightarrow$ many tasks to do

   By reflecting this parallelism within programs we may get:
   - better *structure*
   - better *response* times
   - higher *performance* (on multi-processors)
2. Because we have *parallel machines architectures*:

   network connected computers, multi-processor computers $\Rightarrow$ many hands to keep busy

   Programs must be *(re-)structured* to exploit these architectures

Caveat

- Many pitfalls: Race conditions, deadlocks, starvation

Concurrent Programming

How?

- *Languages* for expressing concurrency
- *Models* for reasoning about concurrent behaviour
- *Mechanisms* for solving sychronization issues
- Techniques for *proving* and *testing* concurrent programs
- Principles for good *design* of concurrent systems