Lecture 1:
Course overview and introduction
Practical details

Schedule:

Course: Tuesdays 14-16, RZ F21
Exercises: Tuesdays 13-14, RZ F21

Course page (check it at least once a week):

http://se.inf.ethz.ch/teaching/ss2005/0268/index.html

Assistant:

Piotr Nienaltowski
Grading

Exam 35%
Will be held at the end of the semester
(not in the semester break)
Exam date (to be confirmed): lecture time on 28 June

Project 65%
Application to be built using SCOOP
The purpose of this course

- To give you a practical grasp of the excitement and difficulties of building modern concurrent applications
- To expose you to newer forms of concurrency such as multithreading and Web services
- To study how the object-oriented paradigm transposes to concurrent settings, and how it can help address concurrency issues
- To introduce you to the main concurrent O-O approaches and give you an idea of their strength and weaknesses
- To study in depth one particular approach: SCOOP
- To enable you to get a concrete grasp of the issues and solutions through a course project
Object technology has interesting potential applications to concurrency, distribution, real-time and Web Services. In practice, a number of obstacles have prevented O-O techniques from repeating in the concurrent world the success they have now achieved in the sequential world.

This course explores the connections between the object-oriented and concurrent programming paradigms, discussing the problems that arise in the process of attempting to merge them.

It reviews the main existing approaches to concurrent O-O computation, including both widely used libraries for multi-threading in Java and .NET and more theoretical frameworks, with a particular emphasis on the SCOOP model. It also provides some of the formal background for discussing the correctness of concurrent O-O applications.
What you should do by next lecture

Read SCOOP introduction at

http://se.inf.ethz.ch/people/nienaltowski/papers/scoop_easy_draft.pdf

and start reading the concurrency chapter at


(or in *Object-Oriented Software Construction, 2nd edition*, available from Polybuchhandlung)
It’s not just sequential any more

Multithreading

Internet-based applications

Distribution

Pervasive computing

Web services
Concurrency & distribution

Everyone wants to do it

Many are doing it

Those who are doing it are not doing it very well
Sequential programming

Used to be messy

Still hard but:

- Structured programming
- Data abstraction & object technology
- Design by Contract
- Genericity, multiple inheritance
- Architectural techniques

Switch from operational reasoning to logical deduction (e.g. invariants)
Concurrent, multithreaded, distributed...

Used to be messy

Still messy

- Examples: threading models in most popular approaches
- Development level: ca. 1968
- Only understandable through operational reasoning
Removing the impedance mismatch

O-O: high-level abstraction mechanisms

Concurrency: semaphores, locks, suspend, mutual exclusion, sharing...
End of lecture 1