Concurrent Object-Oriented Programming

Prof. Dr. Bertrand Meyer

Lecture 1: Welcome and overview
Practical Details

• Schedule
  • course: Tuesday 10-12, RZ F21
  • exercise: Tuesday 13-14, RZ F21

• Course page
  • Check it at least once a week: [http://se.inf.ethz.ch/teaching/2009-S/0268/](http://se.inf.ethz.ch/teaching/2009-S/0268/)

• Course mailing list (professor, assistants, students)
  • concourse@se.inf.ethz.ch

• Assistant
  • Benjamin Morandi: [http://se.inf.ethz.ch/people/morandi](http://se.inf.ethz.ch/people/morandi)
  • Later on in the semester, we will be joined by Dr. Sebastian Nanz
Grading

• Exam 50%
  • Will be held at the end of the semester (not in the semester break).
  • Exam date (to be confirmed): lecture time on last day of course, 26.5

• Project 50%
  • An application to be built using SCOOP
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.
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.
- 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.
Two sides of the same coin

**SCOOP part**
- The “object lesson”
- High-level support for concurrency
- Concurrency solution integrated with an OO programming language, i.e., Eiffel
- Starts from object-oriented programming as a given, adds concurrency

**“Classic” part**
- Survey of classic and modern approaches
- Explains historical evolution
- Illustrates problems and solutions e.g., Java