Practical information

K. V. S. Prasad

TDA384/DIT391 Principles of Concurrent Programming
Chalmers Univ. and Univ. of Gothenburg

20 January 2020

Canvas room and course website

From us to you

Make sure to regularly check the Canvas room and course website:

Canvas announcements, discussion forum.

CTH https://chalmers.instructure.com/courses/8738

GU https://canvas.gu.se/courses/8738

Website lectures, labs, exams, ...
www.cse.chalmers.se/edu/year/2019/course/TDA384_LP3/

These are the primary sources of information about the course.

From you to us

Use the Canvas discussion forum for questions and discussions of general interest to the course: The forum URL is linked from the course website.

!!! Do not share solutions to labs on Canvas (or anywhere else) !!!

Teachers

- K. V. S. Prasad (course-in-charge, examiner, main lecturer)
- Nir Piterman (lecturer)
- Abhiroop Sarkar (TA)
- Carlos Tomé Cortiñas (TA)
- Ivan Oleynikov (TA)
- Andreas Lööw (TA)

The TAs supervise the lab sessions and grade your labs. They also run the website and the Fire system, and occasionally lecture.

Please *do not mail us* if you can help it. Ask during the lectures and labs, or use Canvas.

If you have questions...

1. ask them during the lectures and lab sessions,
2. post them in the discussion forum ,
3. send an email to `pcp-teachers@lists.chalmers.se`,
4. book an appointment with the lecturers or TAs (by email).

Protip: options 1 & 2 are quicker than options 3 & 4.

Student Representatives

	email	Name
TKDAT	emelieblade@hotmail.com	Emelie Blade Andersen
TKDAT	Risne99@gmail.com	William Risne
TKDAT	sjulius@student.chalmers.se	Julius Schumacher
TKDAT	adam.thornblom@gmail.com	Adam Thörnblom
TKDAT	Shabir.walid00@gmail.com	Shabir Walid

Table: From CTH

Still waiting for GU admin to send us their lst of names.
We usually meet the stureps after weeks 2, 4 and 6. Mail them your questions, suggestions, say what you like, what you like less, etc.

Main learning goals

By the end of the course you should be able to

- understand the problems common to concurrent and parallel systems,
- demonstrate techniques and patterns to reason about and write correct and efficient concurrent programs,
- apply those techniques and patterns in modern programming languages.

Overview of the course

- Introduction to concurrency.
- *Part 1.* Classic, shared-memory concurrency in Java:
  - java threads,
  - locks, semaphores, and monitors.
- *Part 2.* Message-passing concurrency:
  - Erlang and the actor model.
- *Part 3.* Parallelizing computations:
  - fork/join parallelism,
  - lock-free programming.

Labs

There will be three labs – one for each part of the course.

1. Trainspotting (Java)
2. CCHAT (Erlang)
3. A-mazed (Java)

Descriptions of the labs, deadlines, and rules are on the website.

- Register your group (2 students) in Fire https://pcp-lp1-19.fire.cse.chalmers.se/.
- Make sure to check the lab/room schedule on the website.

!!! Do not share solutions to labs on Canvas (or anywhere else) !!!

Slides and reading material

Lecture slides are on the website, as are actual lecture notes and tutorials (running text, not bullet points).

Books:

- Ben-Ari: *Principles of concurrent and distributed programming*, 2nd edition. (a few copies should be available from the library)
- Hébert: *Learn you some Erlang for great good* (free online),
- Herlihy & Shavit: *The art of multiprocessor programming* (should be available online from the library)

Computing resources

- Install Java and Erlang/OTP on your computers.
- Try out the examples presented in class; the complete examples will be available on the website.
- Lab 1 (Trainspotting) requires a simulator, which runs on the lab computers (Unix/Linux workstations).
- See the course website for instructions and resources on how to
  - use the lab computers, and
  - set up Java & Erlang/OTP on your own computers.

Programming Languages

- Labs: Java for labs 1 and 3, Erlang for lab 2.
- Exam
  - pseudo-Java (full Java in Appendix if needed)
  - Erlang
- Lectures
  - pseudo-Java (fullish Java occasionally)
  - Erlang
  - Occasionally, we widen horizon beyond exam
    - These lectures or parts of lectures are optional, ignore if you wish.
    - We might use ad-hoc notation, or Promela. You only need to read these, not write.
    - Promela = concise modelling language for concurrency. Allows models beyond Java and Erlang. Runs on simulator with assertion checking.

Ben-Ari's text book uses pseudo code, and supports Java and Promela.

What is Erlang?

- Java, C, Python, etc. are all *imperative*:
  - program = sequence of *commands*, which change the *state*.
  - *Assignment* is the only command. I/O is a special kind of assignment.
    - ⋆ *Control flow* (`if`, `case`, `loop`, etc.) says which assignment is next.
- Erlang
  - *has no assignment*
  - Each *process* can send *messages* to other processes, and receive messages from its inbox, where messages wait till they are read.
  - Messages from a process are typically functions of its state and inputs.
  - These functions are computed *functionally*
- Haskell is a *functional language* you may have seen
  - Erlang is one too (if you ignore the messages), but it has no static types, so expect more errors
  - **If you have never programmed functionally**
    - ⋆ **GET STARTED NOW WITH ERLANG TUTORIALS**

Tutorial on Erlang and functional programming in week 3.