Exercise Session 4

The Dawn of Oblig 1

Oleks Shturmov

<olekss@uio.no> / <oleks@oleks.info>

University of Oslo

IN[59]570: Distributed Objects

February 18, 2019

The source code for these slides is maintained here:

https://github.com/emerald/in5570v19/tree/master/exercise-sessions/04

Agenda

1. Elements of Programming in This Course
2. Discussion: Why Would You Write a Report?
3. What Should Be Clear from Your Report
4. What Is Less Clear Without Yourt Report
5. Conclusion: What To Include In Your Report

Elements of Programming in This Course

A refresher from Exercise Session 1:

1. Writing **programs**
   - Submit readable, preferably working code
   - Test your code, and tell us how to reproduce your test results
   - Refactor your code once it works, and before you submit
2. Writing programs in a **text-based** programming language
   - Use indentation to indicate program structure
   - Use adequate naming
   - Organize code into methods and classes
   - ~~Organize code into files and directories~~ (Maybe later)
   - Apply other common elements of (text-based) programming style
3. Writing programs for **distributed** execution
   - Program fragments execute concurrently on (distant) nodes
   - Program fragments coordinate to get common tasks done
   - Nodes are unreliable (the software/hardware beneath you may fail)
   - Node-to-node communication is unreliable

Why Would You Write a Report?

Discussion

What Should Be Clear from Your Report

(Without looking at your code!)

1. In-how-far you have solved a given task
2. In-how-far you have tested your solution
3. How to compile and run your code (if possible)
4. How to reproduce your test results

What Is Less Clear Without Your Report

1. Your code, especially the nitty-gritty details
2. The reasoning behind your design decisions
3. What to expect when we compile and run your code

Conclusion: What To Include In Your Report

For each task, you should:

- Give an overview of your submission
  - What is located where?
- Give a high-level overview of your solution
  - Use diagrams, pseudo-code, prose, etc.
  - Explain limitations, shortcomings, or additions, if any
- Justify your design decisions
  - Why this way, and not in some other way?
- Explain the non-trivial parts of your implementation
  - For example, if you implemented a special data structure
- Explain how to reproduce your test results
  - How do we compile your code?
  - How do we run your tests?
  - What should we expect to see in case of success, in case of failure?