Review

FIT2099: SEMESTER 2 2018

Another semester older…

- And what have we done?

We have:

- Learned a statically-typed OO language (Java)
- Learned principles of OO design
- Learned UML notation for design
- Used the above in a significant design and build task

Left to do

- For us:
  - Finish marking your assignments!

- For you
  - Study for exam

So…

- What's on the exam???

What's not on the exam!

- Trivia recall
- Writing large slabs of Java

What's examinable

- Everything you studied in this subject, including:
  - Lecture content
  - Readings
  - Lab exercises
  - Assignment-related material

Open book exam

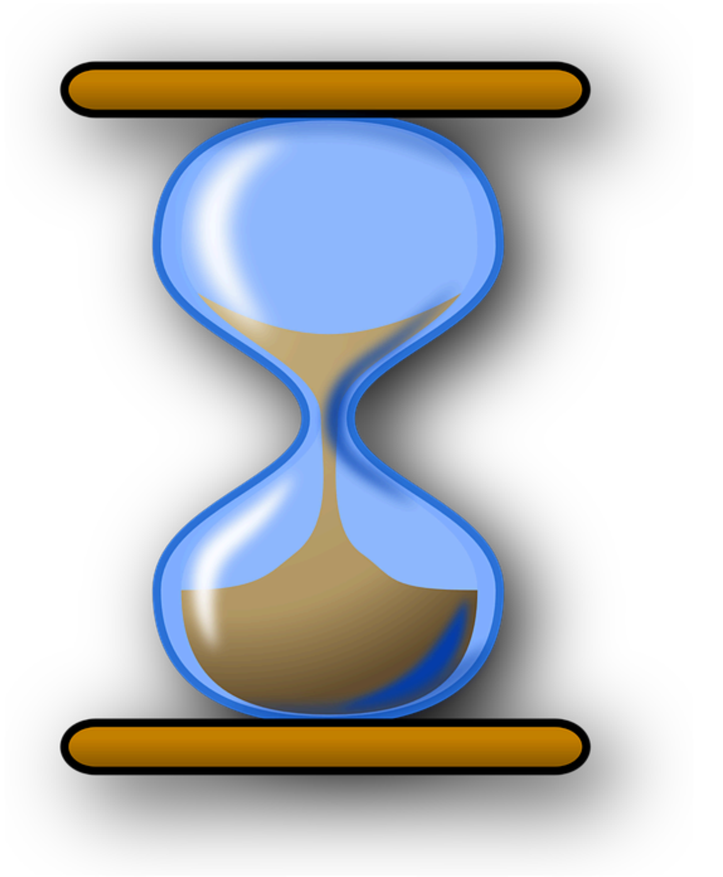
- Do take:
  - UML syntax references – class and sequence diagrams
  - Java reference (if you want) – consider a quick reference sheet
  - Lecture materials and your notes on them
  - Selected readings – or summaries thereof

Exam structure

- 2 Hours
- Open book
- Written answer (no multiple choice)

Open book exams

- Temptation: don't study
- End result…
  - Spend whole exam searching through stacks of paper rather than answering questions
- Either:
  - Remember it
  - Be able to find it QUICKLY



OO Design and implementation

- Things we might ask you to do:
  - Come up with a design based on requirements
    - Document that design
    - Justify that design
  - Critique an existing design/implementation
  - Analyze connections between design and implementation in Java

Exam strategies

Rote learning won't help

- Repeat – no trivia quiz questions
- You can bring ANYTHING YOU WANT (on paper) into the exam room
  - Handwritten or typed notes
  - Photocopied readings
  - Books
- Learning vocabulary will help
- We will test your ability to analyze, apply, explain, not regurgitate
  - So practice those skills

Read and answer the question

- Common mistakes:
  - Read half the question, assume the rest
  - Read a key phrase in the question, brain dump of everything you have memorized relating to that phrase
- We are not interested in your ability to recall/copy slabs of text

TMTOWTDI

- There's More Than One Way To Do It
  - Motto of the Perl community
- Also true of design
  - There will be *multiple* answers to design questions that are acceptable
- Marking schemes for exam design questions will be like miniature versions of assignment ones
  - EG a design would be marked on:
    - Use of correct notation
    - Implementability
    - Clarity
    - Adherence to design principles (DRY, minimizing dependencies, maximizing cohesion etc.)
    - Avoidance of code/design smells

Explain yourself

- Questions may ask you to explain your reasoning
- Don't ignore this!
  - Explanation is often as important (marks-wise) as the "right" answer
  - Sometimes, both answers can be "right", based on explanation
- For many of these questions, you're making an argument:
- *An argument is a connected series of statements intended to establish a proposition*
  http://www.montypython.net/scripts/argument.php
- Your "statements" should ideally be based on:
  - Facts
  - Principles as discussed in the course

Where to now?

- Software engineering -
  - Software that does what it's supposed to do
  - Delivered on time/budget
  - Keeps on delivering over life cycle
- Design is *one* part of that
  - There are other aspects
  - There is also much more to say on design

Related units

- FIT3077 Software Architecture
  - Direct follow on from FIT2099
  - "Design in the large" − not just classes and methods, but bigger components
- FIT2101 − Software Process and Management
  - Software is too big for one person to build
  - How do we work together on a bigger project?
- FIT2107 Software Quality and Testing
  - What is quality?
  - How do we evaluate software artefacts for quality?
- Various third and fourth-year project units
  - You will need good design for these if you want to succeed!

Further reading on design

- *Design Patterns: Elements of Reusable Object-Oriented Software*
- *Code Complete (2nd edition)*
- *Fundamentals of Object-Oriented Design in UML*
- *Refactoring: Improving the Design of Existing Code*
- *Agile Software Development: Principles, Patterns, and Practices*

How to improve your design skills

- Practice!
  - Project units
  - Contribute to open source projects
    - Try implementing an extension to an existing project