Software Design

EECS 3311

Gunnar Gotshalks

LAS 2032
www.eecs.yorku.ca/~gunnar
gunnar@cse.yorku.ca

Course information

www.eecs.yorku.ca/course/3311
https://forum.eecs.yorku.ca

www.eecs.yorku.ca/course/3311

- Textbook
  - *Object-Oriented Software Construction*
  - Betrand Meyer
  - Prentice Hall, 1997, ISBN 0-13-629155-4
- Class schedule
  - Approximate timing for topics, links to slides used in class
- Resources
  - Supplementary notes & tool use

www.eecs.yorku.ca/course/3311

- Timetable
  - Due dates for reports and exams
- Workload
  - 5 reports (4%, 5%, 6%, 7% & 8%) for 30%
  - 2 in-class exams 15% each for 30%
  - final exam for 40%

www.eecs.yorku.ca/course/3311

- Grading scheme

  To pass the the course requires

  $\geq$ 2.0 gpa over the reports

  AND

  $\geq$ 2.0 gpa over in-class exams and final exam

www.eecs.yorku.ca/course/3311

- Verifying your course grade record
  - Follow link *Your grade record* on the *Grades* page for the course
  - Use your EECS account

- Alternate
  - Log into Prism
  - Use the following command

```
courseInfo 3311 [2014-15 F]
```

https://forum.eecs.yorku.ca

- Used by the instructor
  - For announcements about the course
    - Report specifications
    - Notification that course work has been graded
- Used by students
  - To discuss the course material and general problems your with reports
  - **NOT** for posting solutions for reports
- Login using your EECS account

What this Course is About

- Building software systems and components
  - small to medium systems
- Object oriented design and implementation
  - Design patterns
  - Multiple Inheritance
- Design by contract for quality software
- Documenting and describing software
- Evaluating design decisions according to quality factors
- Practice ... practice .... practice ...

On Software Engineering

- Software engineering is a pure intellectual activity
  - Output is documentation
  - Program text is a form of electronic documentation
- Difference with other engineering disciplines
  - Software has no physical characteristic
    - no mass, no heat produced
  - Software implements highly complex functions in a flexible way, making it an essential part of other systems

What this Course in Not Directly About

- Requirements analysis: figuring out what a customer wants
- Teaching algorithms, data structures, syntax
- Teaching programming
  - expect that you know how to program
- Teaching a programming language
  - use a language to explain and apply the concepts
- Just getting programs to work
  - a program that executes is one small piece of the solution.
- **WARNING**: design is challenging
  - there is no right or wrong way to do it

Why Eiffel? – 1

- Why not C++? Java? Smalltalk? Objective-C?
- ***This isn't a language course!*** You're here to learn about design
- Want a language that supports software engineering and production of quality software
- Want a language that has an integrated development method

Why Eiffel? – 2

- Want an industrial-strength language (Java? Getting better)
- Eiffel is used successfully on large projects
- People who have learned Eiffel and OO have no trouble picking up
  - C++, Java, other design methods (Booch, OMT, UML, Objectory, Fusion)
- Designers experienced with Eiffel and its methods are generally more experienced, more competent, and more versatile than others

Study Strategy

- Don't fall behind
  - Learning is work and self-testing
- Attend classes
  - Not all material is in textbook or slides
- For each class
  - Read relevant material before class
  - Do suggested exercises before class
  - Within 24hr after class re-read, think, and expand notes

If you do not reflect on and use the material within 24 hours you forget 50%, and within 48 hours you forget 80%.

How to succeed

```
success_in_3311
    require
        some_courage -- mental and moral strength to venture
    do
        prepare_for_classes
        attend_classes
        critically_review_notes
        plan_build_debug_software
    ensure
        enjoyment_and_mastery_of_the_material
    end
```