# Software Design

CSE 3311

**Gunnar Gotshalks**

CSB 2032
www.cse.yorku.ca/~gunnar
gunnar@cs.yorku.ca

## Course information

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

# www.cse.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.cse.yorku.ca/course/3311

- Timetable
  - Due dates for reports and tests
- Workload
  - 5 reports (5%, 5%, 6%, 6% & 8%) for 30%
  - 2 class tests 15% each for 30%
  - final examination for 40%

# www.cse.yorku.ca/course/3311

- Grading scheme

  To pass the the course requires

  ≥ 2.0 gpa over 3 reports

  **AND**

  ≥ 2.0 gpa over class tests and final exam

- To verify your course grade record use the following prism command

  **courseInfo 3311 [2009-10 W]**

# https://forum.cse.yorku.ca

- Used by the instructor
  - For announcements about the course
- Used by students
  - To discuss the course material and specific problems you have in reports
  - **NOT** for posting solutions for reports
- Login using your CSE account

# What this Course is About

- Building software systems and components
  - small to medium systems
- Object oriented design
- Design by contract for quality software
- Documenting and testing software
- Applying object-oriented programming
- Evaluating design decisions according to quality factors
- Practice ... practice .... practice ...

# On Software Engineering

- Software engineering is a pure intellectual activity
  - Output is documentation
  - Software code can be considered as a special 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?
- ***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 who knows? Getting better)
- Eiffel has been used successfully on many 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 repetition
- Attend classes
  - All material is in textbook or slides
- For each class
  - Read relevant material before class
  - Do suggested exercises before class
  - Same day after class re-read, think, and expand notes

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

# How to succeed

```
success_in_3311 is
    require
        Some_courage -- mental or 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
```