CSC301

Introduction to Software Engineering
What is CSC301 about?

The practice of making software products

- In a realistic environment
- With users in mind
- With a clear purpose and value
- On an on-going basis, in a traceable manner
- As part of a team
- While dealing with changing requirements
Course Goals

- Improve coding skills
- Introduce software development life-cycle
  - The tasks, tools, practices and conventions used by software professionals when creating and maintaining software products.
  - Much more than just coding!
- Get you to think as pragmatic professionals
  - Tooling as part of your workflow
  - Articulate goals, define success metrics and make data-driven decisions
  - Identify common problems/challenges and apply well-known, generic solutions
Topics

● **Software Tools**
  ○ Version control (Git)
  ○ Project management (GitHub)
  ○ Build and/or automation (Travis CI, Maven)
  ○ IDE and/or debugger (Eclipse)

● **A pragmatic approach to tooling**
  ○ Professionals use tools to be more efficient
  ○ And build custom tools (focusing on “bang for the buck”), when they are needed.
  ○ Goal: Maximize productivity
Topics

● Project management
  ○ Software processes
  ○ Focus on modern Agile techniques
  ○ Ex: Scrum, Kanban and Test-Driven Development

● A pragmatic approach to team organization
  ○ Collaboration comes with an overhead
  ○ Organize a team’s workflow ⇔ Reduce overhead
  ○ Goal: Minimize overhead (i.e. maximize productivity)
Topics

- **Product management**
  - Articulating *what* we’re building, *who* we’re building it for and *why* it is useful/valuable.
  - Standard planning tools/techniques such as personas, user stories, diagrams, mock-ups, etc.
  - Scoping and defining a Minimum Viable Product.

- **A pragmatic approach to product decisions**
  - Define success metrics → Experiment & collect user feedback → Make data-driven decision(s)
  - Do it frequently and incrementally improve your product
  - Goal: Maximize benefit/utility/value
Topics

- **Software design & Coding**
  - Best practices and common pitfalls
  - Design patterns such as Iterator, Adapter, DAO, Observer/Observable, Abstract Factory and Builder.
  - Code craftsmanship
  - Various useful topics in software engineering
    - Ex: Lambda expressions (aka callbacks), serialization and persistence, asynchronous programming, lazy-loading and caching, distributed applications, etc.
Logistics ...
Resources

- Course website: http://csc301-winter-2017.github.io
- Discussion Board: https://piazza.com/utoronto.ca/winter2017/csc301
- GitHub organization: https://github.com/csc301-winter-2017
Instructor

- Joey Freund
- Email: joey@cs.toronto.edu
  - If your question is of general interest to the class, please consider posting it on the discussion board, instead of sending an email.
- Office: BA5244
- Office hours: Monday 4-5
Lectures & Tutorials

● Day section, L0101
  ○ Lecture: Tuesday 12-2 @ BA1200
  ○ Tutorial: Thursday 1-2 @ BA1200

● Evening section, L5101
  ○ Lecture: Monday 6-8 @ BA1200
  ○ Tutorial: Monday 8-9 @ BA1200

NOTE: For the first two weeks, we’ll have lectures instead of tutorials.
Prerequisites

- **CSC209**
  - Implicit prerequisite, CSC207
  - Basic Object-Oriented programming in Java
  - Comfortable with Unix command line

- **CSC263/5**
  - Understand the difference between *data type* (interface) and *data structure* (implementation)
  - Basic data structures and types
  - Ex: Array, List, Queue, Stack, Map (aka dictionary), Tree, Graph
# Marking Scheme

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Individual Programming Assignments</td>
<td>30%</td>
</tr>
<tr>
<td>Term Test</td>
<td>25%</td>
</tr>
<tr>
<td>Team Project</td>
<td>45%</td>
</tr>
</tbody>
</table>

No Final exam!
Individual Assignments

- 4 Java coding assignments
- Auto-marked
- Focus:
  - Reading and writing object-oriented code
  - Hands on experience with professional tools
    - Ex: Git, GitHub, Travis CI and Maven.
  - Applying design patterns
- Meant for you to get 100%
  - The task is clearly specified (as JUnit tests)
  - You can submit as many times as you want
  - Travis CI is used for verifying your submission
Individual Assignments

- **Auto-marked assignments ⇒ Strict deadlines & No exceptions**
  - Auto-marker rolls back changes that were committed after the deadline
  - It is your responsibility to make sure your code compiles!
- **Start early and avoid last-minute, *unexpected* technical issues**
  - If your first commit is from the last 24 hours before the deadline, you are taking full responsibility for any unexpected issue that may occur.
  - Responsible professionals prepare for unexpected issues, and so should you.
Term Test

- Two topics:
  - Git/GitHub
  - Applying design patterns to solve common engineering problem
- Tests your ability to communicate (i.e. read/write) using code
  - Focus is on software design, not algorithms
- Based on the individual assignments
  - Therefore, if you don’t understand something about the assignment, you should ask. During office hours, on the discussion board or in class.
- Meant to be fairly challenging
  - There are no easy questions
Team Project

- ~8 weeks long
- 5-7 students per team
- One TA per team, acting as a “mentor”.
- Focus:
  - Identifying users and need
  - Defining a product
  - Building a prototype/MVP
  - Organization a team
  - Working in a traceable manner
  - Presenting your work
Team Project

- **25% - Three deliverables**
  - Concise deliverables presenting your work
  - Meant to be useful, not to add extra work
  - Evaluated by the TA’s

- **10% - Final demo**
  - During the last week of the term
  - Evaluated by the instructor

- **10% - Consistent individual contribution**
  - Commit history & graphs on GitHub
  - Participation in tutorials throughout the term
  - You are expected to contribute valuable work, at the very least, twice a week.
Cheating

- Don’t cheat!
- Feel free to discuss ideas with others, but don’t take notes or share code with others.
- When in doubt, ask your instructor or TA.
Please keep in mind that CSC301 is a hands-on course!

In other words - A lot of fun, but also a lot of work.