McGill School of Computer Science

# COMP 303 Software Development - Winter 2016

[Basic Information] [Description] [Reference Material] [Evaluation] [Lectures]

## Basic Information

| | |
|---|---|
| **Instructor:** | Martin Robillard |
| **Email:** | For your own protection, I only respond to student inquiries that originate from official McGill email accounts |
| **Time and place:** | Mondays and Wednesdays, 16:05-17:25, LEA 26 |
| **Office hours:** | See on myCourses |

## Description

Principles, mechanisms, techniques, and tools for software development, with a focus on software design.

### Sample Course Topics

*Principles:* Single responsibility, separation of concerns, encapsulation, substitutablity, interface segregation.
*Mechanism:* Exception-handling, serialization support, concurrency and synchronization, reflection.
*Techniques:* Design patterns, design by contract, unit testing, refactoring.
*Tools:* Integrated software development environment, automatic testing tools, coverage analyzers, static checkers.

### Prerequisites

The official prerequisites (COMP 206 and COMP 250) are the absolute minimum program requirements. This course targets students who have reached a basic level of Java programming fluency. This means that students registering to take COMP 303 should be able to, with a minimum of hesitation, write Java programs to solve small and well-defined problems, use a revision control system to organize their work, and use a debugger to trace through execution and inspect run-time values. COMP 302 and COMP 251 are recommend background.

### Learning Outcomes

The learning outcomes for this course are organized along the four main conceptual axes of the course: *principles*, *mechanisms*, *techniques*, and *tools*. The following table lists the expected learning outcomes for the course (the table reads by row, from left to right).

After this course, you should be able to...

| | **Principles** | **Mechanisms** | **Techniques** | **Tools** |
|---|---|---|---|---|
| **Name, using the proper terminology** | The important first principles of object-oriented software development | The common programming-language based mechanisms used to build OO software applications | Common software development techniques | A number of software development tools |
| **Describe and explain** | The purpose of each principle and how it can be applied | How each mechanism works | How to apply each technique and when it should be applied | The theory underlying each tool, and the technique(s) the tool supports |

| | | | | |
|---|---|---|---|---|
| **Apply** | Each principle | Each mechanism | Each technique | Each tool |
| **Evaluate** | Whether the application of a principle is appropriate to a given situation | The technical consequences of a solution involving the mechanism | The cost and benefits of using the technique in a given situation | The suitability of different tools for a given task |
| **Create** | A complete object-oriented application based on the first principles of object-oriented software development, the structured use of programming language mechanisms, the application of software development techniques, and the use of software engineering tools | | | |

## Reference Material

Required Textbook: Horstmann, Cay. Object-Oriented Design and Patterns, 2nd Edition. Wiley, 2005. Available at the Paragraphe Bookstore.

Complementary Resource: Martin, Robert. Clean Code: A Handbook of Agile Software Craftmanship. Prentice-Hall, 2008. Available on-line from the McGill domain.

Since this is a software design course, some of the required reading will be of source code, which will be drawn from the following two projects:

Sample Project: Solitaire.

Diagramming tool (and sample project): JetUML.

## Course Work and Evaluation

You will have the opportunity to apply the course material in a series of **individual assignments** that will culminate in the creation of an interactive card game. The assignments are self-directed but **necessary** to build the skills and knowledge that will allow you to pass the lab tests and exams.

| | |
|---|---|
| **Lab tests** | 20% |
| **Midterm exam 1** | 20%/5%/0% |
| **Midterm exam 2** | 20%/5%/0% |
| **Final exam** | 40%/60%/75% |

**Important Notes:**

1. The midterm exams will count for 20% if their score is higher than or equal to that of the final exam.
2. If the grade on one or both midterms is lower than that of the final exam, one midterm can count for 0% and the other one can optionally be worth 5%. The difference is made up by the final exam.
3. Missing a midterm exam results in a grade of 0%. However, if you miss both midterm exams, you can present original, idependently-verifiable documentation justifying your absence to *both* midterm exams. Valid causes are limited to *unforseeable and unpreventable* circumstances.
4. The following material is subject to evaluation: Material presented in class, all material in the mandatory reading, operational knowledge of the tools covered in the course, anything completed to meet the learning objectives of the assignments. The midterm will cover all lectures up to and including the lecture immediately before it. The final will cover everything.

**Official Academic Integrity Statement** McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offenses under the Code of Student Conduct and Disciplinary Procedures (see www.mcgill.ca/students/srr for more information). Note that we reserve the right to run plagiarism detection software on all software submitted as assignments.

**Language Policy** In accord with McGill University's Charter of Students' Rights, students in this course have the right to submit in English or in French any written work that is to be graded.

## Lectures

General philosophy of the role of lectures, and related policies:

1. Lectures **complement** the course material with demonstrations, examples, discusssions, and class activities: they do **not replace** the mandatory readings and assignments.
2. The required reading must be done in advance to be able to follow the lecture.
3. The lectures will **not** be Power Pointless. I will occasionally use slides to provide visual support for the material. The slides should not be expected to consitute a self-contained study document. I must emphatically warn anyone against attempting to pass the course by memorizing the slides.
4. Although attendance is not monitored, I strongly recommend attending lectures. Lectures will feature walkthroughs of design and implementation tasks which you are encouraged to follow along on your own laptop.
5. If you decide to attend lectures, you are asked to refrain from using your computing devices in a distracting or disruptive manner. This includes watching videos, playing games, etc.
6. **No audio or video recording of any kind is allowed in class**, with the one exception that you can take pictures of the projection screen (only) if you can do so in a non-distuptive manner. Note, however, that practically all lecture material will be posted after class.

These policies are subject to revision at any point during the term.

## Overview of the Schedule

The following is the expected progression of topics. The detailed course schedule is available on myCourses.

| Lecture Topics |
|---|
| Introduction |
| Class design |
| Polymorphism |
| Design by contract |
| Introduction to Unit Testing |
| Object identity and life-cycle |
| Composite and Decorator design patterns |
| Modeling Object State |
| Observer Design Pattern |
| Inheritance-based reuse |
| Abstract Classes; Template Method Design Pattern |
| Error Handling |
| Software Quality Checking |
| Introduction to GUI design |
| GUI Design and Implementation |
| Threads |
| Synchronization |
| Generic Types |
| Visitor Design Pattern |