UNIVERSITÉ Concordia UNIVERSITY GINA CODY SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

SOEN 6441
Advanced Programming Practices
Computer Science and Software Engineering
Fall 2023

Course Instructor:
Amin Ranj Bar
amin.ranjbar@concordia.ca

Office Hours:
Thursdays, 17:30-18:30, by appointment only
Location: ER 1175

Tutorials: Please see your class schedule for details

Labs: Please see your class schedule for details

Course Calendar Description:
Problems of writing and managing code. Managing code complexity and quality through a programming process. Self-documenting code, and documentation generation. Software configuration management. Best practices for writing unit tests to control code quality. Advanced practices such as multithreading concurrency, code reuse, and fault tolerance. A project. Laboratory: two hours per week.

Prerequisite Knowledge: none
Co-requisites: none

Specific Knowledge and Skills Needed for this Course:
Although this course does not officially have course prerequisites, it is taken for granted that the students are already proficient programmers that master the object-oriented programming paradigm. The lectures, exercises, examinations, and completion of the project will necessitate pre-existing programming skills. This course is not meant to teach you how to program, but rather how to extend your programming skills and apply them in a team project.

This is an advanced graduate programming course. You must be very familiar with object-oriented programming; in particular, you should have good knowledge of "classical" Java programming and be comfortable working with software IDEs, such as Eclipse, as well as version control systems, such as Subversion or Git.

Rationale:
Most students coming out of introductory programming courses know how to write simple programs written in a specific programming language. Other more advanced courses show them how to use the more advanced features of the language, thus reaching for a complete understanding of the syntactical constructs of a specific programming language. Industrial programming requires a lot more diversified skills than simple mastery of language syntax. Industrial programmers have

to know how manage the complexity of their coding activities, install and use libraries, create reusable, documented, fault free and fault tolerant code. This course aims at broadening the knowledge of the students to these concepts, techniques and tools that are complementary to what is taught in standard programming courses.

Course Objectives:
Improving the practical programming skills of students by emphasizing real-life aspects of programming that are not dealt with in regular introductory and advanced programming courses. Practical mastery of techniques and tools for the writing of superior quality code and complementary programming artifacts such as inline documentation, design patterns, and automated testing infrastructure. The goal of this course is to become familiar with modern industry-grade programming techniques, as they are used for example in commercial web sites. Features such as scalability, robustness, fault-tolerance, real-time data handling and responsiveness require a number of advanced programming techniques that are typically not taught as part of an introductory programming course. Here, we will study such techniques based on the new features introduced with Java 8+.

Course materials
Raoul-Gabriel Urma, Mario Fusco, and Alan Mycroft, Modern Java in Action: Lambdas, streams, functional and reactive programming, September 2018, ISBN 9781617293566, Manning Publications.
https://www.manning.com/books/modern-java-in-action

Walter Savitch. Absolute Java. Addison Wesley. Sixth Edition, 2015.

Additional research papers will be posted as recommended/required reading. Great thanks to professor Paquet for providing his lecture notes.
Note: Check the website (Moodle) for important notices that you must read, lecture notes, projects, etc.

Grading Scheme

- Project (3 deliveries): 10% + 20% + 20%
- Midterm Exam: 20%
- Final Exam: 30%

To pass the course, you should pass each component of the course (project and the exams). You must have at least 50% total mark, as well as in the combined examinations mark in order to pass the course.
1. Project: The project is to be tackled by teams of 5 or 6 members. It is divided into 3 practical assignments related to the project. Each assignment includes the delivery of an operational subset (i.e. and increment or build) of the final project. The project consists of a large program whose development involves most of the topics discussed in the lectures. Each build is graded independently of the other builds, following a grading scheme given prior to the due date. Each build will be presented orally in a practical demonstration in the course laboratory. Each phase has both a group and an individual contribution part. **Note that the attendance to the project demo is**

mandatory; if you do not participate in both group and individual project demonstrations, you will not receive marks for the project.

2. Exams: The exams are closed-book whose goal is to individually test the comprehension of the material taught in class. The final examination covers the materials from the entire semester, including lectures, research papers, and the project and will be conducted on a date determined by the university. Passing the exams is necessary for passing the course. There is no calculator allowed in the exam. In addition, students must not have a cell phone or other electronic device on their person or anywhere in their work area during the exams. As a general rule for exams, you need to show all your work (just the final result is not enough). Any rough work on question paper will not be considered.

Tentative Course Schedule

(Week 1)	Introduction. Software Process & DevOps.
(Week 2)	Extreme Programming and Agile development
(Week 3)	Revision Control Systems and Integration
(Week 4)	Testing Fundamentals & JUnit testing framework
(Week 5)	Program Architecture and MVC
(Week 6)	Midterm.
(Week 7)	Build Tools. Documentation Generation.
(Week 8)	Design Patterns.
(Week 9)	Fault tolerant programming.
(Week 10)	Parallel Programming Dependency Injection.
(Week 11)	Generic programming.
(Week 12)	Project Final build.

Lab Details

The laboratory instructor(s) will be there to help you on the project, most particularly on the use of the tools and libraries to be used for the implementation. The three project deliveries will be held in the laboratory.

Details on assessment tools:

There are no pre-set cutoff points for the final grades; the cutoff points will be decided based on an assessment of difficulty level, class performance, fairness, and instructor's wisdom from teaching and grading the course in the past. That is, there is no definite rule for translation of number grades to letter grades.

Policy: You may ask for a make-up exam or later submission of assignments ONLY under a university-approved condition, such as sickness with a doctor's note. Other events such as a business travel are not excused. You should make the request for a mark-up exam or a later submission of an assignment before the date of the corresponding exam and the deadline for the corresponding assignment, respectively. In particular for exams, you need to inform me at least one week in advance, unless it is an emergency. Note that, your request is not accepted, until you receive an explicit confirmation from me. Do keep the confirmation as a proof. No exception.

Health and Safety Guidelines

All health and safety rules specific to this course can be found in the lab manual. General health and safety instructions and available health and safety trainings can be found at:

Safety Programs - Concordia University (https://www.concordia.ca/campus-life/safety/general-safety.html)

On Campus Resources
Please visit Student services at Concordia University for the services available Gina Cody School students.

Disclaimer
In the event of extraordinary circumstances beyond the University's control, the content and/or evaluation scheme in this course is subject to change.