COMP 2450 – Software Development 1

Course Description

Calendar entry

Real-world software development involves many different parts of Computer Science: software design, validation and verification, user interface design, computer security, and information management. This course introduces students to the elements of software development. May not be held with COMP 2150 or COMP 2160. COMP 1006 is recommended. Prerequisite: COMP 1020 or COMP 1021.

General Course Description

Introductory Computer Science courses help you think like and communicate with computers: you learn about what computers actually do and you learn how to reframe solutions to problems so that a machine can do the work for you.

Real-world software development goes beyond basic problem solving and programming, always involving many different parts of Computer Science. We will discuss how to design and structure large software applications, validating and verifying data, information management, computer security, and user interface design.

We'll also look at the social implications of all of these different parts of software development, how what you do as a Computer Scientists affects you, your community, and the broader world.

This is a project-based course where we will begin to build larger software applications that require planning, design, and testing.

Detailed Prerequisites

Before entering this course, a student should be able to:

- Use classes and objects effectively to design and implement structured representation of implementation (COMP 1020).
- Write code that deals with large sets of data using files stored on disk (COMP 1020).
- Formulate recursive solutions to simple problems and write simple recursive methods (COMP 1020).
- Use built-in data types (e.g., Java's `ArrayList`) to work with a collection of data.
- Implement simple searching and sorting algorithms (COMP 1020).
- Write code that handles failures gracefully, including through basic error checking and the use of language constructs like exceptions (COMP 1020).
- Use version control software (COMP 1006).
- Use debugging strategies and tools to independently debug software (COMP 1006).

Course Goals

By the end of this course students will:

- Design and implement larger, interactive, text-based applications.
- Verify and validate that code meets a set of well-defined expectations through manual and basic automated testing.
- Model data and its relationships, persist data in a structured format.
- Describe and build secure applications through appropriate programming practices.
- Analyze the impact that a program's design choices may have on users and other people and factors within the system's ecosystem.

Learning Outcomes

Design and implement large applications

Students should be able to:

- Divide code from a complex project into higher-level modules, for separate development and compilation.
- Use an automated build tool to build a complex project.
- Describe how they would continue to build and improve the software project that they've built.

Software verification and validation

Students should be able to:

- Explain the purpose of testing code.
- Create test data (general case, edge case) for an abstract data type.
- Implement a test harness that automates the testing of an abstract data type.
- Define the pre* and post-conditions for a routine and apply them in a program.
- Define the invariants that encapsulate the valid states of an abstract data type.
- Use preconditions, postconditions, and invariants to validate the run-time behaviour of an abstract data type being used in an application.

Data modelling

Students should be able to:

- Design, diagram, and implement a domain model using appropriate modelling techniques (e.g., class diagrams).
- Persist and load data and its relationships using a structured text-based format.

Programming practices for secure applications

Students should be able to:

- Identify and use different strategies for error handling, including exceptions, return values, and global states.
- Implement input validation and error handling in an application.

Designing software for humans

Students should be able to:

- Sketch the flow of interaction in an interactive system before implementing it.
- Describe and implement appropriate methods for handling and reporting errors to different kinds of users.
- Explain why some methods for handling and reporting errors to users are inappropriate.