Department of Computer Science and Software Engineering
Concordia University

SOEN 6461 Software Design Methodologies
Design Assignments Fall 2018

Design Assignments

The assignments require you to practice creating a design, communicating your design, and analysing your design to demonstrate that the design works, and that the design has the properties — for example, performance, security, usability, etc — required for the task.

You do not program your designs, so your program can not serve as a way to check your design. You must be able to communicate and check your design independently of a program.

Design work requires you to be able to express your ideas about a design to your fellow designers. This includes the key features of the design, why these features are good, or necessary, and what are the consequences of using this design feature (both positive consequences and negative consequences).

These design conversations can happen in formal review meetings, or as informal chats around the water cooler.

These design conversations happen no matter which methodology you are using for software development.

Due Dates

Assignment 1: *Design for StudentClassSchedule class.*
Due 23:59 Wednesday 10 October 2018

Assignment 2: *Consistency checking of schedule.*
Due 23:59 Sunday 4 November 2018

Assignment 3: *Heuristics to create a schedule.*
Due 23:59 Sunday 2 December 2018

Advice

Scope the problem so it is do-able. Decide what information the program needs. Decide which qualities are critical to success of the design. Hand-execute a few simple examples of the problem. How do you know the design works? How do you know the design has desired qualities, such as resource efficiency?

Assignments 2018
— Undergraduate Student Class Schedule Builder

The undergraduate students in the Bachelor of Computer Science program at Concordia University select a schedule of course sections for each semester in order to fulfil the requirements of their program.

Your job is to design a system to record their schedule, to check their schedule for consistency, and to guide them in building a schedule.

Getting started: The assignments are organized so that you focus on different aspects of the information used by the system, and the different tasks of the system.

Assignment One is about how to represent a weekly class schedule for the undergraduate student, as an object of class StudentClassSchedule.

Assignment Two is about checking a schedule for consistency with the requirements of the degree program (in terms of pre-requisites and co-requisites), checking that the student has done the required pre-requisites and satisfies the C-rule, and that there are no time conflicts between the selected sections for lectures, tutorials, and labs.

Assignment Three is about developing heuristics to automatically build a schedule for an undergraduate student, or at least to guide a student in building their schedule.

Information: In general there is the following information available in the Concordia University undergraduate calendar:

- the requirements of the degree program (see below);
- the rules of the degree program (eg see the C-rule, below);
- a list of course descriptions, including pre-requisite and co-requisite requirments.

The general structure of an undergraduate degree in Computer Science is:

- the CS core consisting of 33 credits, the courses: COMP228, COMP232, COMP233, COMP248, COMP249, COMP335, COMP346, COMP348, COMP352, COMP354;
- a complementary core of 6 credits consisting of the courses: ENCS282, ENCS393;
- Mathematics electives consisting of 6 credits, chosen from the courses: MAST218, MAST219, MAST234, MAST235, MAST332, MAST334, MATH251, MATH252, MATH339, MATH392;
- CS electives consisting of 30 credits, chosen from other COMP and SOEN courses;
- General electives consisting of 15 credits, chosen from outside of the ENCS Faculty, for example, from the Humanities, Social Sciences, Sciences, and Business.

Note that some courses are 4 credit courses, or 3.5 credit courses, while most are 3 credit courses.

The C-rule requires a student to achieve a grade of C- or better in a 200-level course before they can use that course as a pre-requisite.

Each department in Concordia University will coordinate to offer a collection of course sections for a semester.

- Not every course is offered every semester. Many are offered only once a year.
- Many courses will have multiple sections per semester, offered at different times.
- Sections may be lectures (LEC), tutorials (TUT), or labs (LAB). Assume that when tutorials and labs are offered, then they are a mandatory accompaniment to the lecture.

Each section has a capacity limit (but you may ignore this issue).

The information in the student record of a student:

- student ID
- name
- address
- email
- phone
- degree program
- list of exemptions for courses
- list of transfer credits for courses
- list of courses taken each semester, with a result: DNE, DISC, FNS, F, D-, D, D+, C-, C, C+, B-, B, B+, A-, A, A+ (Note that some courses may be repeated.)

Typically, a student will take 12 or 15 credits a semester, but is free to take anywhere from zero to 18 credits per semester.

Assignment One

The Problem

Design a class called StudentClassSchedule for the class schedule of a student. It represents the information to describe the weekly class schedule of an undergraduate student for one semester. It has the basic operations required.

Guidance

What assumptions are you making?

What data structures are used in the class?

How would you format the information when printing an object of StudentClassSchedule?

Note: You must document the design of your class as part of your submission.

Assignment Two

The Problem

Determine whether a schedule is consistent with the requirements of the program, including time conflicts.

Guidance

What assumptions are you making?

What other information do you need in order to confirm consistency? What is the API that you would design to access each piece of required information?

What design patterns are relevant here?

Did you check the C-Rule?

Did you check mandatory tutorials and labs?

Concordia has two campuses — SGW and Loyola — and offers classes on both campuses. Did your check for time conflicts to allow for travel time between campuses?

Note: You must document the design of your process for consistency checking as part of your submission.

Assignment Three

The Problem

Design a class StudentClassScheduleBuilder that guides a student in the creation of their schedule.

Guidance

Use heuristics.

What assumptions are you making?

Be explicit about each heuristic, how it is represented, how it is applied, when it is applied.

Hand execute — step by step — several examples of a student creating a schedule with the help of your class.

Note: You must document the design of your overall architecture, and your heuristic process for guiding the building of the schedule as part of your submission.

Your Submission

There are three assignments were you must create, evaluate, and document a design.

The deliverable is a design document of about five (5) pages as a pdf file, submitted to the ENCS electronic submission system under `assignment-1, assignment-2, assignment-3` respectively. The first line of your document should identify the course, the assignment number, your name, and your student number; do not have a separate title page. Use a 12pt font or larger throughout the document.

The Design Document

The document is meant to be short and concise. The three design documents are practice at communicating. They are also practice in identifying which of your design decisions are obvious decisions to other designers, and which design decisions are important, critical to the success of the design, and not obvious. You must communicate the important information. There is no need to document the obvious design decisions.

Your design document should have subsections:

1. a brief description of the problem (up to one half (0.5) page);
2. a concise description of the design (from two (2) to four (4) pages), where you focus on presenting the most important parts of the design (whatever they may be);
3. a brief description of the major design decisions (one half (0.5) to one (1) page);
4. a brief description on how you reviewed the design (one half (0.5) to one (1) page), and which qualities of the design were considered during the review;
5. a glossary of important things (one half (0.5) to one (1) page in length), such as classes, objects, methods, algorithms, and data structures.

Write your document for a reader at the level of a CS or SOEN graduate like yourself. Use UML diagrams where and when they are appropriate for communicating information about the design, but do not use them if they are not needed.

Guidance

Focus on clarity of communication of the design and of the justification for the design being correct and having the desired qualities. Avoid flaws in organization of material such as missing information.

Section 1 should clearly state the problem being solved in the design, including all inputs, all outputs, and all assumptions. For example, it should be clear whether the system is looking for one solution or all solutions to the problem (in this case a schedule); and whether you, the designer, assume there is a solution for the given input.

Even so, a good design should be robust, and not assume that there is a solution.

Section 2 should make clear

- what are the parts of the design (i.e. the classes)
- what is their role/purpose/responsibility and
- their class API, where each method has a complete signature, and contract.

There should be

- a black-box description of the class interface (API) in terms of contracts (or some very clear description); and
- it should be explained how the parts work together to solve the puzzle, such as a clear sequence diagram with text, or clear pseudocode for the "main algorithm".

Pseudocode should be properly indented as code, and mention which objects are called.

Nobody should discuss the user interface. UI design is not a topic of the course.

Avoid diagrams that are unclear, i.e. too crowded, font too small, poor choice of names, etc.

Section 3 should have more than one design decision, and they should not be trivial decisions. You need to say why it was "major", its consequences, and ideally what alternatives you considered.

Section 4 should discuss

- correctness
- resource usage for space and time, and
- scaleability.

The marker will look for clarity of the statements, and use of formulas. The marker will look for clarity of the justification, and expect links or references back to the design description. That is, clear use of method and class names, etc.

Show formulas for estimates of time or memory resource usage.

Give reasoning.

Section 5 should not include common CS terms like class, object, state, method, algorithm. It should refer to concepts in the problem and in the design only. It does not need to be exhaustively complete, but it should not simply list three definitions.

Marking Scheme

For each design document, the marking scheme will assign a mark out of 2 for each of the subsections listed above, for a total mark out of 10:

- 0 marks for a missing subsection, or very unclear section, or where the section content is not appropriate for the section (that is, it is mis-placed);
- 1 mark for an informative subsection that does not have too many errors;
- 2 marks for a clear, informative, concise, almost error-free subsection.

Length Penalty: There is a penalty of 10% of the assignment mark for each page of your submission over 7 pages.

Late Penalty: There is a penalty of 10% of the assignment mark for each day that your assignment is late.