SOEN 6461 Fall 2018 — Assignment 2 Marking

Mark each section 0, 1, 2 marks; no fractions.

Problem Description

This should contain

- Task Description
  Develop an activity/process/method to check the consistency of a schedule with the requirements of the undergraduate program,
- Assumptions
  The problem description should make it clear which rules of the program are being checked as part of consistency.
- Inputs and Outputs
  Inputs: a StudentClassSchedule for a given student and a givem semester, along with required information on student record, program description, course description, and classes with times and locations
  Outputs: a boolean result indicating consistency, or not
- Assumptions on Hardware and Software Context
  There are none specified, other than OO for the design.
- Indication of Non-Functional Qualities of Interest
  Besides correctness, the students were asked to consider resource usage for memory and cpu steps.

For two marks, must include task clearly described, clear list of which rules are checked, something on inputs/outputs, and something about qualities required.

Give zero marks, if the task is not clearly described. It is a description for Assignment 2 and not for the whole system.

Design Description

Main things to make clear in the document is

- the overall process for checking consistency
- which class is responsible for doing the overall task of checking consistency
- how checking work is assigned amongst different classes, as delegated substeps of the overall process

Ideally, they should encapsulate rules into classes (see Strategy pattern), so that all types of rules share a common abstract interface. [This is to make it easy to add new kinds of rules into the consistency checking. Internally, the implementation of each rule will differ, but their external look via the interface should be the same.]

There should be a UML class diagram to highlight the main players in the consistency checking process (but it should not be complete in details, only highlighting key methods and attributes of the required classes).

There should be a UML sequence diagram (or similar) to illustrate the execution of the overall consistency checking process, and how work is assigned to other objects/classes.

It should be clear how each type of rule is checked.

But none of the descriptions should lead to unwieldy pseudocode.

There is no absolute requirement for the use of UML diagrams. But if they include one, then it must have a number, title, be clear and readable. A diagram does not replace the need for text descriptions of the above items.

There is no requirement that they use the Strategy pattern.

Do not worry about exceptions.

For two marks, they must include good descriptions of how checking is done by the overall process and by each type of rule. The rules checked must match the assumptions in Problem Description section. They must check prerequisites, co-requisites, and time conflicts. Also, any diagram must be clear, and consistent with the text.

Give zero marks, if they do not describe overall process of consistency checking clearly.

Major Design Decisions

This should contain a short list of decisions with a description

- What was the issue
- What was decided
- What was the impact of the decision
- Maybe alternatives

Possible major decisions are:

- how Strategy pattern is used for each rule that is checked
- distribution of responsibility across the classes for the overall process of checking consistency
- how a rule is represented in the design
- and more

For two marks, they need to talk about at least two decisions and make clear the issue, their choice of solution, and the impact of their choice. No need to talk about alternatives.

Give zero marks, if they do not mention any of the three listed decisions above.

Design Review

This should contain

- Brief Description of their review steps
  Scenario-driven review
  Which qualities: correctness, data usage, computation time
  Some details of concrete scenario, system state, and inputs
- Justification of correctness
  Logical reasoning showing that the checking process leads to the correct result for every schedule; which requires reasoning that checking each rule is correct for all schedules.
  This is straightforward, as it only requires that they recongnize that the overall process should check each rule in each class in the schedule; and that they show that checking each rule individually is correct.
- Formulas for data and computation
  This should be done in terms of the number C of classes in the schedule; the number R of Rules that are checked; the memory m(Class) used to store one Class object; the memory m(Rule) used to store one Rule object; and the steps c(Class, Rule) required to check one Class object against one Rule object.
  They need to consider that different types of rules will have different storage and different computation steps, hence use the max across m and c.
  Basically C*m(Class)+R*[max of m(Rule)] and C*R*[max of c(Class, Rule)]

For two marks, they need some specifics on the scenarios, reasoning about correctness, and formulas for memory and time usage.

Give zero marks, if they simply say "scenario-driven" review, without details.

Glossary

This should contain

- List of Dictionary Entries
  Short definition, like a dictionary, of important terms in design. Should have schedule, class, course (with a clear distinction between course and class). Might have student, semester, time, location, etc.
  It is okay to include terms for methods (and classes) related to consistency checking, eg Rule or Rule::check()
  Not terms from OO or SE. So not class in OO sense, or object, state, design, etc

For two marks, need at least 3 terms with good to-the-point definitions, and no OO/SE terms.

If only OO/SE terms, then give zero marks.

Marks Spreadsheet

For me, create a spreadsheet which gives the five marks against each students name and ID. Have a column with the total out of 10.

Also indicate number of days late in a column, and page length in a column.

Do not count days late until after 9am following the deadline; do not count Declaration Form in the pages, and do not count a "title page" if they have a separate title page (ie has no design information).