Better SISU

By Aleksi Hasu, Leo Mandara, Jyri Hakala

## User manual

## Extra features

Thorough testing of logic classes. All logic-centred classes were implemented using test-driven development. An interface was agreed on for each class, and tests were implemented first to make sure the class to be implemented worked according to what was agreed.

Student settings window. At the start of the program a window is shown for entering a student’s information.

Program saving. A planned degree can be saved by pressing the “SAVE” button in the bottom right corner of the “DEGREE STRUCTURE” tab. The saved degree can be accessed the next time SISU is opened by choosing the applicable student’s name from the drop-down menu on the login screen.

## Class diagram

To be added

## Responsibilities, pre and post conditions

The Better SISU implementation stores data in three different module classes stemming from one abstract base class, and one course unit class. Due to the varying nature of data stored, the responsibility of these classes is just to store any data given to them. These classes are as follows:

Module

Stores common data for Modules such as their name, id, groupId, and sub modules and sub course units.

GroupingModule

Adds no functionality to module as such but distinguishes grouping modules from other module types.

DegreeProgramme

Adds maximum credits to module.

StudyModule

Adds completion status, module requirements, and organizers to module.

CourseUnit

Stores all information about a course unit, including id, groupId, name, code, and minimum and maximum credits.

Reading data from a source into modules and course units is done by the classes ModuleReader and CourseUnitReader. Both rely on a class implementing the JsonStringFetcher interface to get their respective data. Two different classes implementing JsonStringFetcher were used, one for reading data from the remote SISU and the other for reading data from a local JSON directory.

JsonStringFetcher

Defines a method for reading Module data and a method for reading CourseUnit data from a source. The classes implementing JsonStringFetcher are LocalJsonFetcher and UrlJsonFetcher. UrlJsonFetcher assumes that a module or course unit is available for every given groupId. LocalJsonFetcher takes into account that not all files may be present.

ModuleReader

Reads data from JsonStringFetcher into modules. Assumes that all data given is in the same format as in the SISU remote data.

CourseUnitReader

Reads data from JsonStringFetcher into course units. Assumes that all data given is in the same format as in the SISU remote data.

Student

Stores a student’s basic information and their chosen DegreeProgramme.

StudentWriter

Writes JSON files representing Students to local storage. Common use is calling write(Student) for a single student, but other methods are included to ensure usability.

StudentReader

Reads JSON files, as saved by StudentWriter, from local storage into Student objects. Common use is calling readAll() to fetch all saved students.

ModuleAdapter

Is used by StudentWriter and StudentReader to differentiate types of Module when writing/reading JSON files. Adds fields “type” and “properties” to each, where type is self-explanatory, and properties holds the actual attributes of the class.

## Division of work

The division of work was agreed on in weekly meetings. The different Module and CourseUnit classes were implemented as a group in order to get the group up to speed. After the basic data classes, the team mostly worked on their own responsibilities. Aleksi’s responsibility was the user interface, Leo’s responsibility was reading data from the Kori API into the defined data structure, and Jyri’s responsibility was classes for storing and reading student data. Though everyone had their own part, the group gave feedback on each other’s completed modules. Larger git merges were also discussed in the weekly meetings, so that everyone knew what would be merged into main.

## Bugs and missing features

* When marking courses as complete in the “DEGREE STRUCTURE” tab, the changes are only reflected in the “STUDENT INFORMATION” tab on program restart.
* UI was only tested manually