1. Classes and methods related to handling Sisu API

1.1 Overview

The following UML diagram illustrates implemented classes related to handling Sisu API, their features and relationships.

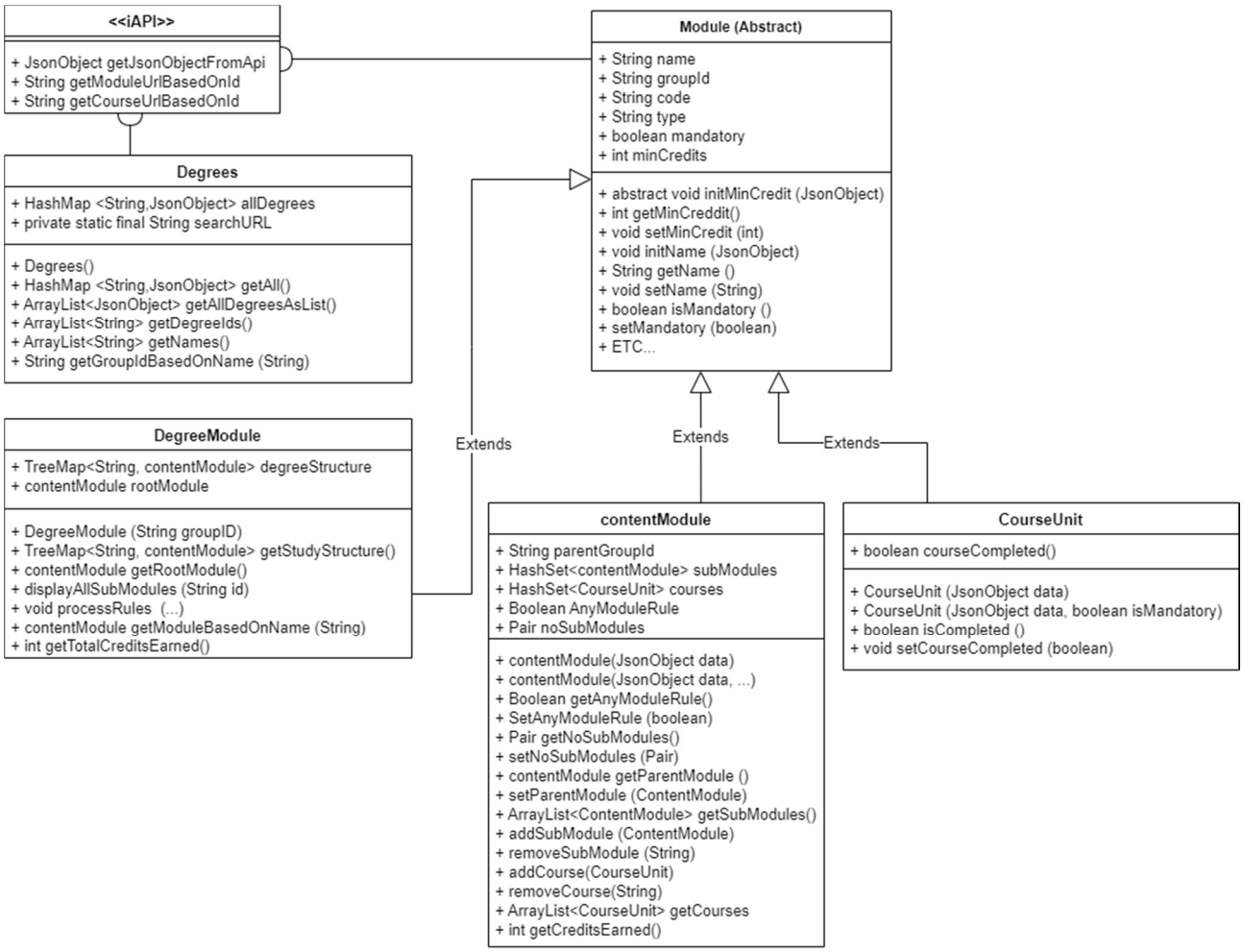

Figure 1. UML diagram.

1.2 Class Module.java

This abstract class implements iAPI. It handles several common features of modules and courses such as name, groupId and associated number of credits. In particular, it contains methods that are used to initialize these properties based Json objects that are retrieved from Sisu API.

1.3 Class Degrees.java

This class implements iAPI. It is used for retrieving Json objects of degreeModules based on their groupID. degreeModules are contained in the field hashMap <*String, JsonObject*> ***allDegrees***, where its key is *groupID* of the program and its value is JsonObject containing data for the degree program (retrieved from Sisu API). Degrees object can be created as follows (for example):

Degrees allStudyPrograms = new Degrees ();

For example, the following retrieves names of all degrees and displays them on console (in alphabetical order):

ArrayList<String> degreeNames = allStudyPrograms.getNames();

for (var name: degreeNames) { System.out.println(name) };

If the name of a degree program is known, the corresponding groupID can be found as follows:

String searchTest= "Teatteritaiteen maisteriohjelma";

String IdTest= allStudyPrograms.getGroupIdBasedOnName(searchTest);

Further, degreeModule (see next section) can now be created based on the retrieved groupID as follows:

DegreeModule DMtest= new DegreeModule(IdTest);

Finally, it should be noted that ***getTotalCredits*()** returns the total number of credits earned in the degree program.

1.4 Class DegreeModule.java

The class extends Module class, and it has the following non-inherited field:

- HashSet<contentModule> ***degreeStructure*** : data structure containing the degree program.

This class is used for handling modules and courses that are contained within a degree program. A degree program is stored to the object ***degreeStructure***, which contains contentModule (see below) objects that are included in the degreeProgram module. New DegreeModule object can be created based on the groupID of a degree program as follows:

String test2= "uta-tohjelma-1705"; // TKT, kandi

DegreeModule DMtest2 = new DegreeModule(test2);

The degree structure can be retrieved as follows:

HashSet<contentModule> degreeStruct1 = DMtest2.getDegreeStructure();

More examples on the methods in DegreeModule will be given later.

1.5 Class contentModule.java

The class extends Module class. It contains and handles data on modules that have type StudyModule or GroupingModule. The class has following non-inherited fields:

- String ***parentGroupId*** : groupID of the parent module (or *null* in case of root module).
- ArrayList<contentModule> ***subModules*** : submodules (childrens) of the module.
- ArrayList<CourseUnit> ***courses*** : courses contained in the module.
- Boolean ***AnyModuleRule*** : can submodules be freely added/ removed?
- Pair ***noSubModules*** : minimum (.min) and maximum (.max) number of submodules. Default values are 0 and 99.

Practically all methods in the class are related to accessing and manipulating these fields. Note that in this case, minCredits is initialized from JSONobject key “targetCredits” (retrieved from Sisu Api). Also, the method ***getCreditsEarned***() return the sum of credits earned from completing courses that are under the module.

1.6 Class CourseUnit.java

The class extends Modules class. It contains and handles data on modules that have type CourseUnit. The class has the following non-inherited field:

- Boolean ***IsCourseCompleted*** : is course completed?

It has also the corresponding method:

- void ***setCourseCompleted*** (Boolean): set the completion status of the course (true: course is completed, false: course is not completed.

Note that in this case, minCredits is initialized from JSONobject key “credits” (retrieved from Sisu Api). Also, the field “type” for CourseUnits is always *null*.

2. Graphical user interface (GUI)

Graphical user interface is divided into four different files: App.java, FirstWindow.java, SecondWindow.java and SeparatorApp.java. App.java runs UI, FirstWindow.java creates first view for UI, SecondWindow.java creates the second view for UI and SeparatorApp.java generates separator line to UI view for the second window.

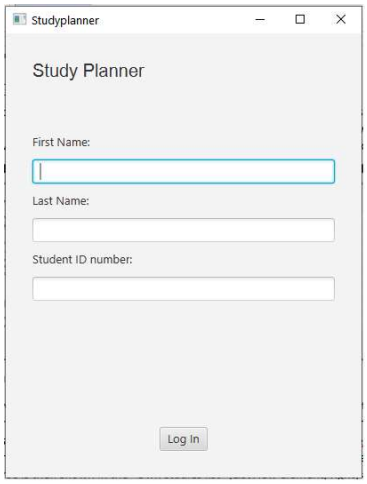

Figure 2. First window.

When the program launches, the window shown in Figure 2 shows up. After filling out the fields and pressing “Log In” button, the following window shows up:

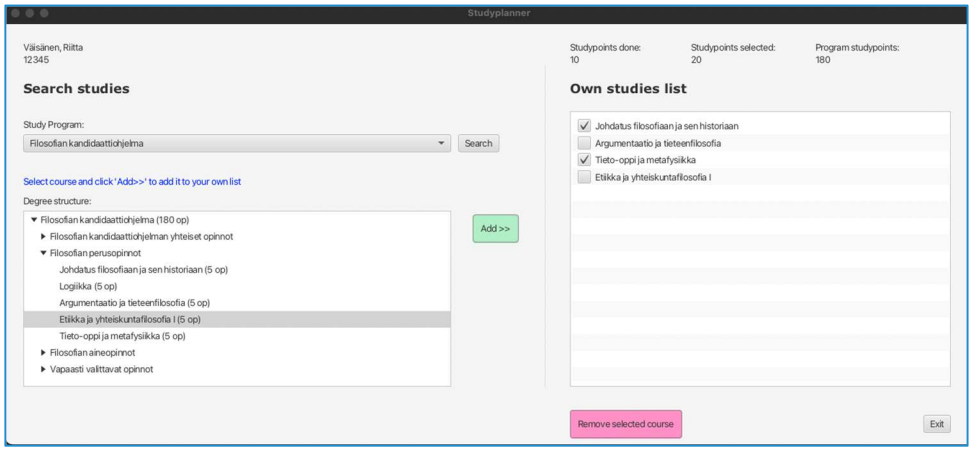

Figure 3. Main (second) window.

Figure 3 shows the main window where Study Program is chosen, and available and selected courses are shown (see section 5 below for details on how to use the UI). The study program is chosen from the drop down menu (top left), the degree structure is shown hierarchically in TreeView element (bottom left). The courses are shown in the "Own studies list" (ListView element, right). Studypoints on right top of the window are updated accordingly after relevant user actions (top right). Also, the user information is shown (top left).

Study Program drop-down box is created with ComboBox element. Degree structure is created with TreeView including VBox elements. Own studies list is a ListView which CheckBoxes. ComboBox, TreeView, ListView and button elements have event handlers.

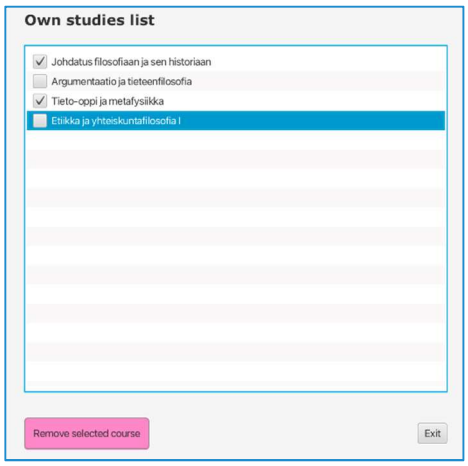

Figure 4. ListView element showing chosen studies.

Figure 4 shows situation where user wants to delete course from the list (course "Etiikka ja yhteiskuntafilosofia I"). Whole item has to be selected to delete it.

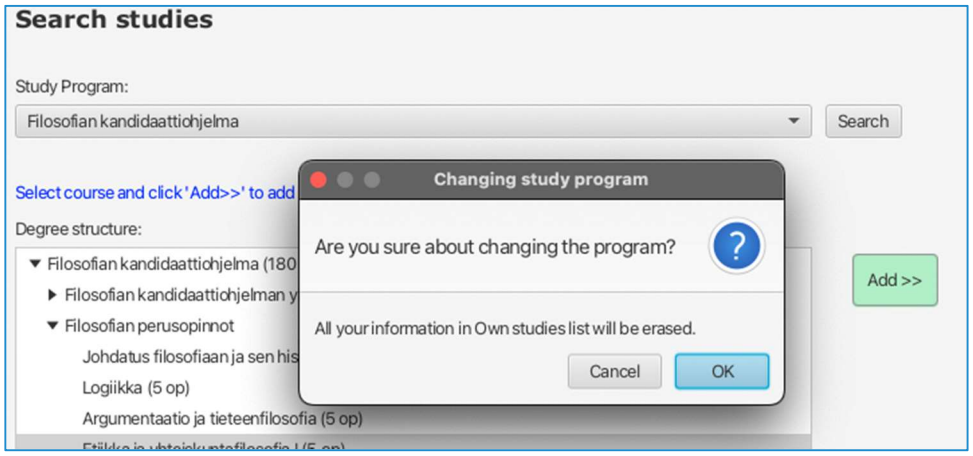

Figure 5. State of UI when the user is changing Study Program and alert pops up.

3. Unit testing

Unit testing was made for the classes Module, Degrees, DegreeModule, contentModule, and CourseUnit. There are 11 tests total, and they test basic operations of the classes and some exceptions that may be thrown.

Degrees.java is for example tested with testFindID(), which tests finding the groupdID of the degree based on its name.

DegreeModule is for example tested with testGetDegreeName(), which tests getting the name of the degree based on its groupID. It also tests Module through inheritance.

ContentModule is for example tested with testGetCourseName(), which tests getting the name of the specific degree's course based on their groupIDs. It also tests Module through inheritance and CourseUnit.

There are exception tests as well that test some cases when exceptions are thrown.

Tests can be run in the Maven project by clicking Sisu (project name) --> Test --> information of the tests', and results of the tests are shown in the console output.

4. Division of work between group members

Group members:

- Tatu Anttila
- Sanna Nykänen
- Miia Raerinne

Tatu Anttila was responsible for implementing classes and methods described in Section 1, and for some of the GUI implementation described in Section 2 (TreeView element). Sanna Nykänen was mainly responsible for designing and implementing GUI, especially SecondWindow.java. Miia Raerinne was responsible for developing unit tests and also for some parts of the GUI (first window). Total workload was divided practically evenly among the group members.

5. Brief user manual and program functionalities

START : First window

1. Type your information in to text fields. Text fields can't be empty.
2. Press log in to get to the next view.

Second window

1. Select **Study Program** from drop-down menu.
2. Click **Search** button.
3. Select course from **Degree Structure** list.
4. Add course to your own list by clicking on **Add >>** button.
5. You can check the box from **Own studies list** to indicate that course is competed.
6. Study points update regarding your choices. Program shows completed study points, study points of all courses in your **Own Studies list** and total study points of selected study program.
7. You can uncheck boxes or remove the whole course from your list by clicking **Remove selected course** button.
8. You can change **Study Program** from the drop-down menu.
9. Exit via **Exit** button or closing the window.

Program functionalities

1. Retrieves study programs from API.
2. Generates list of studies related to selected study program. Study program studies are retrieved from Sisu API.

3. User can add courses to their own list.
4. User can check courses to be completed
5. User can erase studies from their own list.
6. User can change study program.
7. User gets feedback from selected and completed courses.