

STUDENT EXPLORER

Semester project 02/03
Prof. Bertrand Meyer
Supervisor: Karine Arnout



Dominik Wotruba
Grundstrasse 11
8048 Zürich

CONTENTS

1. PROJECT DESCRIPTION	2
OVERVIEW	2
SCOPE OF THE WORK	2
2. PROJECT MANAGEMENT	3
OBJECTIVES AND PRIORITIES	3
CRITERIA FOR SUCCESS	3
METHOD OF WORK	3
QUALITY MANAGEMENT	3
3. PLAN WITH MILESTONES	3
SPECIFICATION	4
DESIGN	4
IMPLEMENTATION	4
VALIDATION AND VERIFICATION	4
DOCUMENTATION	4
DEADLINE	4
SCHEDULE	5
4. THE STUDENT EXPLORER APPLICATION	5
MAIN WINDOW OF STUDENT EXPLORER	5
THE ADD STUDENT FORM	8
THE EDIT STUDENT FORM	9
DELETE STUDENT	11
THE MANAGE STUDENT GROUP FORM	11
THE MANAGE COURSES FORM	12
THE FILE MANAGEMENT	15
EXIT	16
HELP	16
5. THE IMPLEMENTATION	18
THE MODEL	17
THE VIEW	20
THE CONTROLLER	25
PERSISTENCY	26
6. TESTCASES	26
7. FUTURE EXTENSIONS	26
8. REFERENCES	26

1. PROJECT DESCRIPTION

Overview

The purpose of this project was to design and implement a graphical user interface (GUI). The resulting product called “Student Explorer” makes it possible to administrate groups of students: store their personal information (name, id, ...), keep track of the courses they are registered to and associate the corresponding grades with weights, and possible “testats”.

Scope of the work

Student Explorer provides the following features:

- Management of students with various criteria: Every student belongs to a student group (for example a semester) and takes part in zero or more courses, in which he could have been testified or have passed various exams with various outcome of different weight.
- Management of student groups: Ability to add, to remove or to change student groups is provided.
- Management of courses: Ability to add, to remove or to change courses is provided.
- Management of exam types: An exam type consists of a name (e.g. semester exam) and a weight (To the semester exam may be assigned weight n and to the normal exam may be assigned weight m). For each course there is the ability to add, to remove or to change exam types.
- Every student participates in zero or more courses. The range of courses corresponds to the courses defined in Student Explorer.
- Searching for students: Student Explorer provides the ability to search students according to specific criteria.
- Persistency: Student Explorer provides the ability to store all information into a file, including a support for XML.

2. PROJECT MANAGEMENT

Objectives and priorities

The main priority is to provide the basic functionality, namely management of student groups, student information, courses and grades.

Criteria for success

All specified functions of the GUI are implemented.

Method of work

Student Explorer has been developed using the cluster model of the software lifecycle [3]. Further, all important steps have been demonstrated to the supervisor.

Quality management

Documentation

- An API-style explanation of the implemented code on the basis of the documentation generated by EiffelStudio [2].
- A thesis report documenting and explaining the results.

Validation steps

- All major changes to the GUI and code have been delivered to the supervisor.
- Validation of the application using various test cases.

3. PLAN WITH MILESTONES

Specification

- To identify proper groups of students and courses, I analysed the organization of courses at ETH, especially the group of Prof. Bertrand Meyer.

Design

- First prototype of the GUI using EiffelBuild [2] to have a basis for discussion about a possible implementation.
- Definition of classes and of the relations between them. (inheritance, client relationship)

Implementation

- Implementation of all specified classes.

Validation and Verification

- Validation of the application by running various test cases.

Documentation

- Documentation of the classes and writing a thesis report documenting and explaining the results.

Deadline

Deadline is 20.03.2003

Schedule

Task	1.1-6.1	7.1-17.1	17.1-2.2	22.2-29.2	1.3-20.3
Analysis of students and courses at the ETH	█				
First prototype of GUI		█			
Start of GUI implementation		█			
Refined GUI with support for exam types			█		
Finishing the basic implementation			█		
Interruption of the project due to military service				█	
Implementation of XML-support and the find student form				█	
Testing				█	
Documentation (Semester thesis report)					█
Final Presentation					█

Table 1. Project schedule

4. THE STUDENT EXPLORER APPLICATION

Main window of Student Explorer

The main window of Student Explorer includes seven parts as shown on Figure 1 below: A title ①, a menu ②, a button list ③, a student tree ④, a labeled multi-column list ⑤, a text area ⑥ and a picture area ⑦.

The student tree ④, the labeled multi-column list ⑤, the text area ⑥ and the picture area ⑦ are resizable.

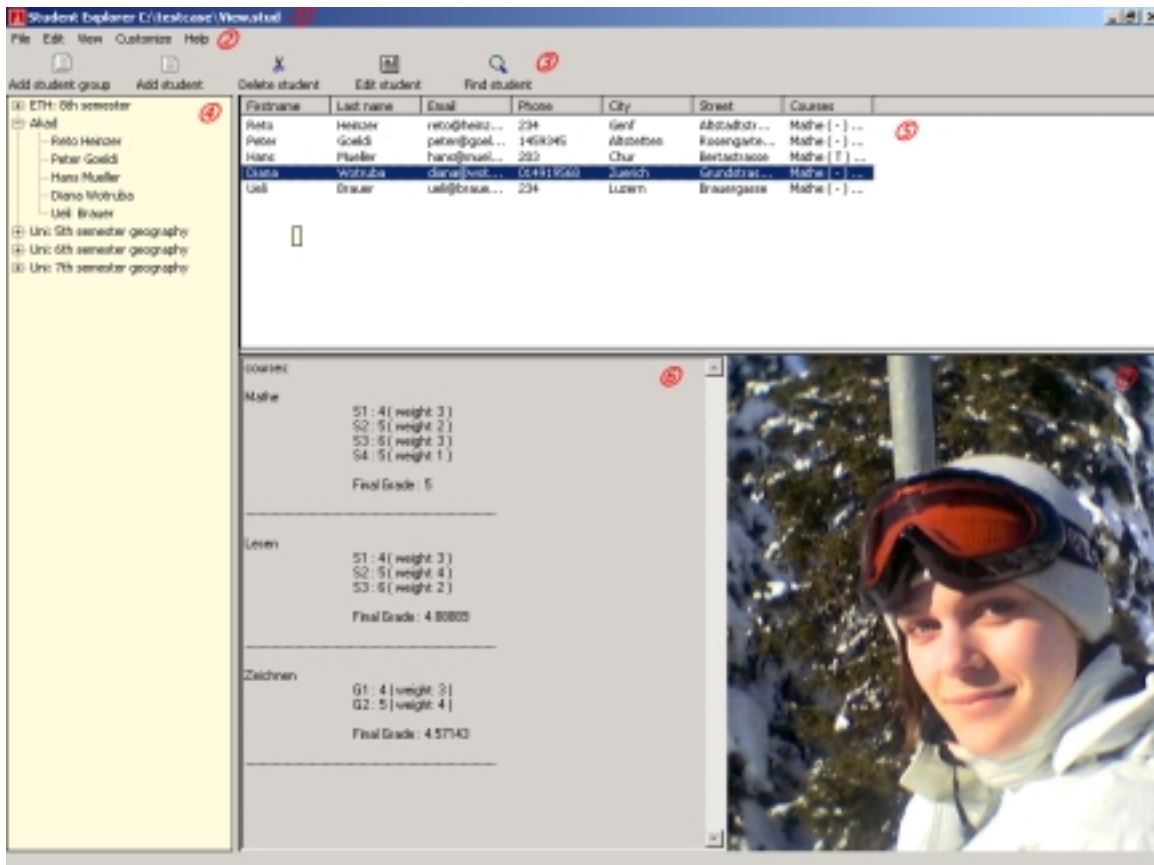


Figure 1. Main Window of Student Explorer

The title

The title ① shows, which file is currently open. A star (*) is displayed beside the title, indicating whether changes have been made to Student Explorer.

The menu

There are five different groups of menus ②: File, Edit, Customize, View and Help. The menu File includes all menu items for file management, menu items to create a new file and to exit Student Explorer.

Menu Edit has items to manage students and an item to search for students. In the menu Customize all items aim at managing student groups and courses.

The Help menu includes a menu item, which shows the about box and a help menu item, which displays a small tutorial about Student Explorer (how to manage students, courses and grades).

The button list

The button list ③ contains the most frequently used functions. From the left to the right buttons are called “Add student group”, “Add student”, “Delete student”, “Edit student” and “Find student”.

The “Add student group” can be displayed either by clicking the “Add student group” button or by selecting the “Manage student group” item in the menu “Customize”.

On condition that a student group exists a student can be added by pressing either the “Add student” button or the “Add student” item in the menu “Edit”.

The “Delete student” button is used to remove students. To remove a student an item must be selected in either the multi-column list or the student tree.

The “Edit student form” is displayed either by pressing the “Edit student” button or by selecting the “Edit student” item in the menu “Edit”. Before a student can be edited, the corresponding item has to be selected in either the labeled multi-column list ⑤ or the student tree ④.

The “Find student” form is displayed by clicking either the “Find student” button or the “Find student” item in the menu “Edit”.

All forms and functions introduced in this section will be explained in more details later.

The student-tree

The student tree ④ contains, on the one hand student groups and, on the other hand students, which are assigned to a particular student group. Therefore clicking any of the student groups, it displays all the students belonging to that particular student group. Clicking on a student in the student group, shows all data available for that student: it is displayed on the right in the labelled multi-column list ⑤ and text area ⑥. If a picture is available, it will be shown in the picture area ⑦.

The multi-column list

The following features are displayed in the multi-column ⑤ list: A unique number assigned to a particular student together with his/her first name, last name, email address, phone number, postal address and all courses in which the student is taking part. For every course a student is taking, there is either a minus sign “(-)” or a T-sign “(T)” assigned on addition to the course name in the labelled multi-column list ⑤. The T-sign means that the student has testified. The minus sign is shown if the student has not testified yet.

The text area

The text area ⑥ is divided into two sections: a course section and a comment section. Both sections are shown only if necessary.

All courses the particular student is taking part in can be found in the course section. A course consists of various exams and a final grade. For every exam the student is taking it displays the name, grade, and weight of the exam. The final grade is calculated for every course according to the grades and weights. It also displays some additional information for every course, such as “testats” (i.e. whether a student has testified or not).

The picture area

If there is a picture it will be shown in the picture area ⑦ as shown on Figure 2.

The add student form

After clicking the “Add student” button (a) or selecting the “Add student” item in the menu “Edit”, the “Add student” form will appear (see Figure 2 on next page).

Figure 2. The add student form

Compulsory and optional items

The “Add student” form contains two types of items. There are compulsory items (meaning they have to be filled in) and optional items. Compulsory items are: The student group (1), the student number (2), the first name (3), the last name (4), the email address (5), the phone number (6), the city (7) and the street (8). Optional items are: The comments (9), The courses (11), (since there are students, which don’t visit any course) and the exams types (16), (since there are courses, which don’t include exams).

The unique and permanent student number

For every student a unique student number has to be filled in the student number text field (2). Note: the student number is permanent. It can be deleted only by removing the student. If the student number filled in is not unique an error message is displayed after

pressing the “Add student” button (21). It is recommended to use the “ETH legi-number” as unique number, though any other unique number can be used.

Pictures

To import pictures press the “Import” button (23). A file dialog appears. The current version of Student Explorer only supports pictures whose file has the extension “.bmp”.

Course management

If course exist (have been created through the manage course form), it is possible to associate some with the student (i.e. the courses in which the student takes part by selecting the desired course in the course combo box and pressing the “Add course” button (12).

Removal of a specific course is possible by selecting the specific course in the course list (20) and pressing the “Remove button”.

Managing exam types

Once a course has been added, exam types for this course can be inserted. An exam type consists of an exam type name and a weight. It can be added by selecting a grade in the grade combo box (15) and by pressing the “add” button (17). Removal of exam types is possible by selecting the exam type in the exam type list (20) and pressing the “remove” button (22).

The course notebook

Pressing the desired notebook item (14) provides the ability to navigate between courses. For every course there is also the possibility to testify it by selecting the “testat” check box (19).

Once all compulsory features have been filled in, the student can be added by pressing the “add student” button (21).

Pressing the “clear” button (22) clears the whole form. All entries are cleared, including the course notebooks.

The edit student form

After pressing the “Edit student” button (a) or selecting the “Edit student” menu item in the menu “Edit”, the “Edit student” form appears with currently set values (see Figure 3).

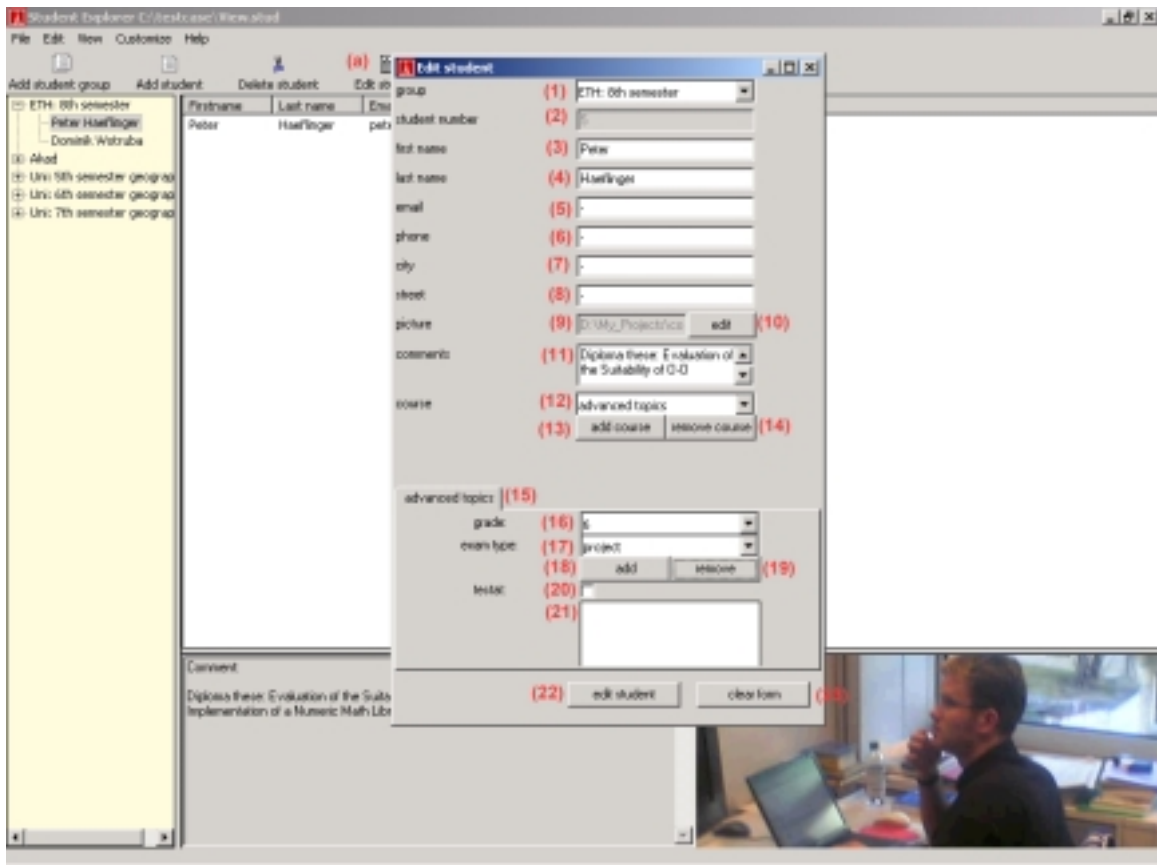


Figure 3. The edit student form

The group combo box

The group combo box (1) shows the actual group the student belongs to. Selecting a different group will place the student into that group.

The unique and permanent student number

The student number (2) is unique and permanent. Removal is possible only by removing the student. Note: to change a student number the student has to be deleted and reinserted with the desired new student number.

Compulsory and optional items

All other text fields (first name (3), last name (4), email (5), phone (6), city (7), street (8), comments) can be edited. Note: all fields except comments (11) are compulsory, which means they have to be filled in.

Edit courses

To edit a course belonging to a student the desired notebook item (15) has to be selected. Choosing the desired notebook item enables the management of exam types. There is another option to add and remove exam types as mentioned in previous section about the “Add student” form. Furthermore a checkbox is provided to testify courses (20).

Once all desired items have been edited, the “edit student” button (22) can be clicked.

Note the state of a student changes only if the “edit student” button (22) gets clicked.

Delete student

To delete a student, the corresponding line must be selected first. There are two options to select a student: Either in the multi-column list (1) or in the student tree.

Once a student has been chosen, it can be removed by pressing the “Delete” button (a).

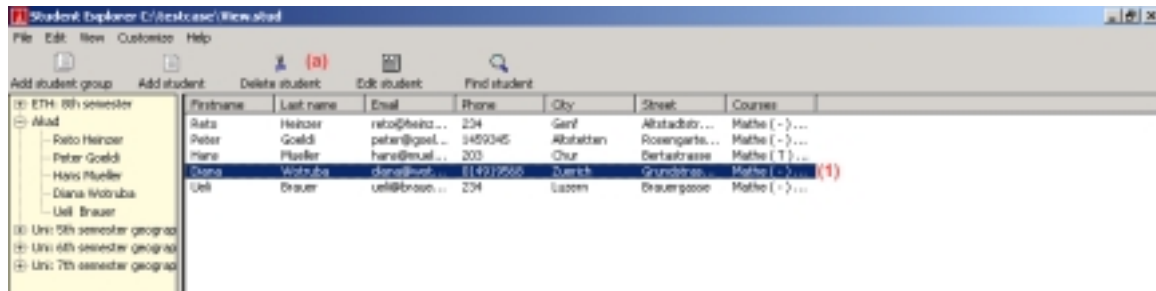


Figure 4. Deleting a student

The manage student group form

Once either the “Add student group” button (a) or the “Manage student group” menu item in the menu “Customize” has been selected, the “Manage student group” form appears. (see Figure 5)

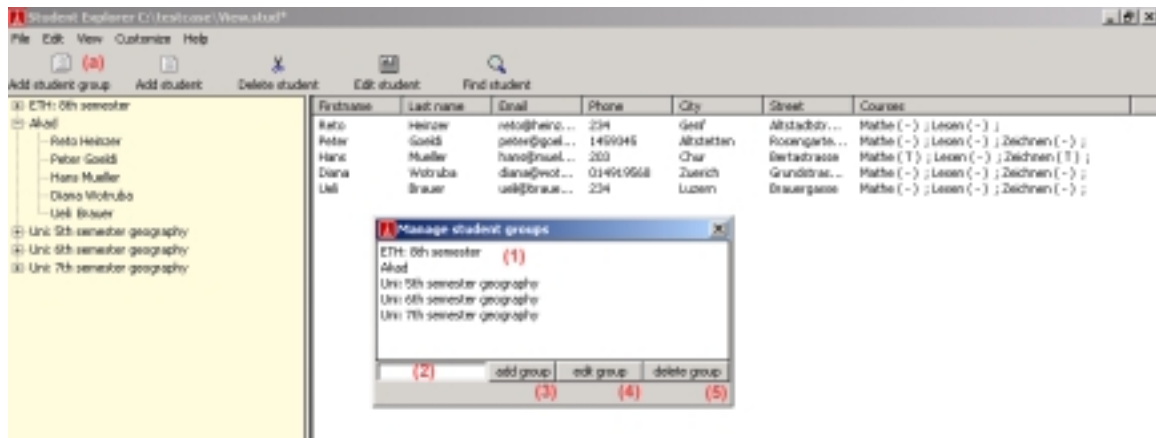


Figure 5. The manage student group form

The role of the “Manage student group” is to add, remove and edit student groups.

Add student group

To add a student group, the student group name has to be filled in the group text field (2) below the student group list (1). Pressing the “Add” button (3) will insert the student group with the name just entered (2).

Edit student group

To edit a student, a student group has to be selected in the student group list (1). The student group name appears in the text field (2) and then can be changed. Pressing the edit button (4) changes the student group globally, meaning that it will change the student group name for all students belonging to that student group.

Delete student group

To delete a student group, the corresponding line must be selected in the list (1). Pressing the delete button (5), deletes the desired student group, together with all the students belonging to it.

The manage courses form

To manage courses the “Manage course” item has to be selected in the menu “Customize” (a). Then, the “Manage course” form is displayed (Figure 6).

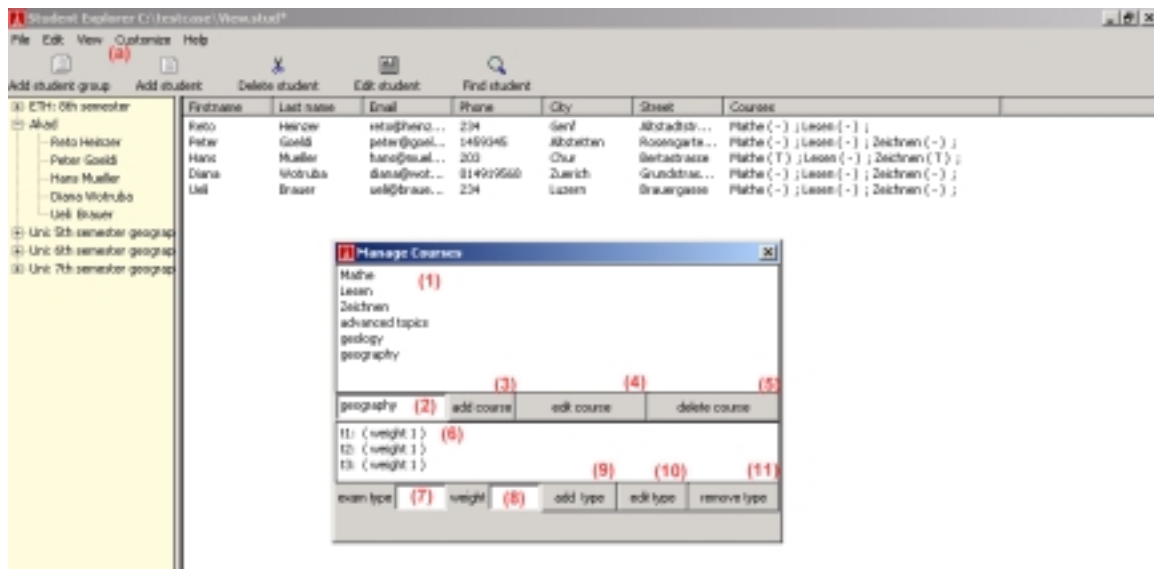


Figure 6. The manage courses form.

Add a course

To insert a new course the course name text field (2) below the course list (1) has to be filled in. Pressing the “Add course” button (3) validates the entry and adds the course to the list of existing courses.

Edit a course

To edit a course, selection of a course in the course list (1) is required. After choosing the desired course, its name is displayed in the course name text field (2). The name of the course can be changed. Pressing the edit button (4) will rename the course with the newly specified name. Note: Editing a course affects all the students participating that course.

Delete a course

To remove a course, selection of the desired course in the course list (1) is required. Pressing the “Delete course” button (5) deletes the chosen course. Note: if you delete a student then all students participating in that course are affected.

Add an exam

Insertion of exams requires that at least one course exists, because the exams are associated with a course. To add an exam a desired course has to be chosen in the course list (1) and both fields: exam type (7) and weight (8) have to be filled in. Note that the exam type has to be unique, but only for the specific course. It is possible to have two courses with the same exam type. Pressing the “add type” button (9) inserts the exam to the chosen course.

Note: weight must be an integer value (i.e. 0, 1, 2, 3). The weights have the following semantics: An exam of weight n will count n/m times as much as an exam with weight m . Exams with type zero have no effect on the final grade.

Edit an exam

There are two ways to edit an exam type: either editing the name of the exam type or changing the weight. The exam type name has to be unique for a specific course and therefore can never be changed to an existing exam type name. To edit an exam, an exam-type has to be chosen in the exam type list (6) and then the exam type name or value has to be changed in the corresponding text fields (7, 8). Clicking the “remove type” button (11) will delete that particular exam type (see next).

Delete an exam

To delete an exam, an exam type has to be chosen in the exam type list (6). Pressing the “remove type” button (12) deletes that particular exam type.

The find student form

Pressing either the “Find student button” (a) or the “Find student” item in the menu “Edit” displays the find student form (see Figure 7). This form contains a combo box called criteria (1) where the search criteria can be selected, a search text field (2), a search button (3), a labelled multi-column list (4) (where the results are displayed), a text area (5) and a picture area (6) for detailed information.

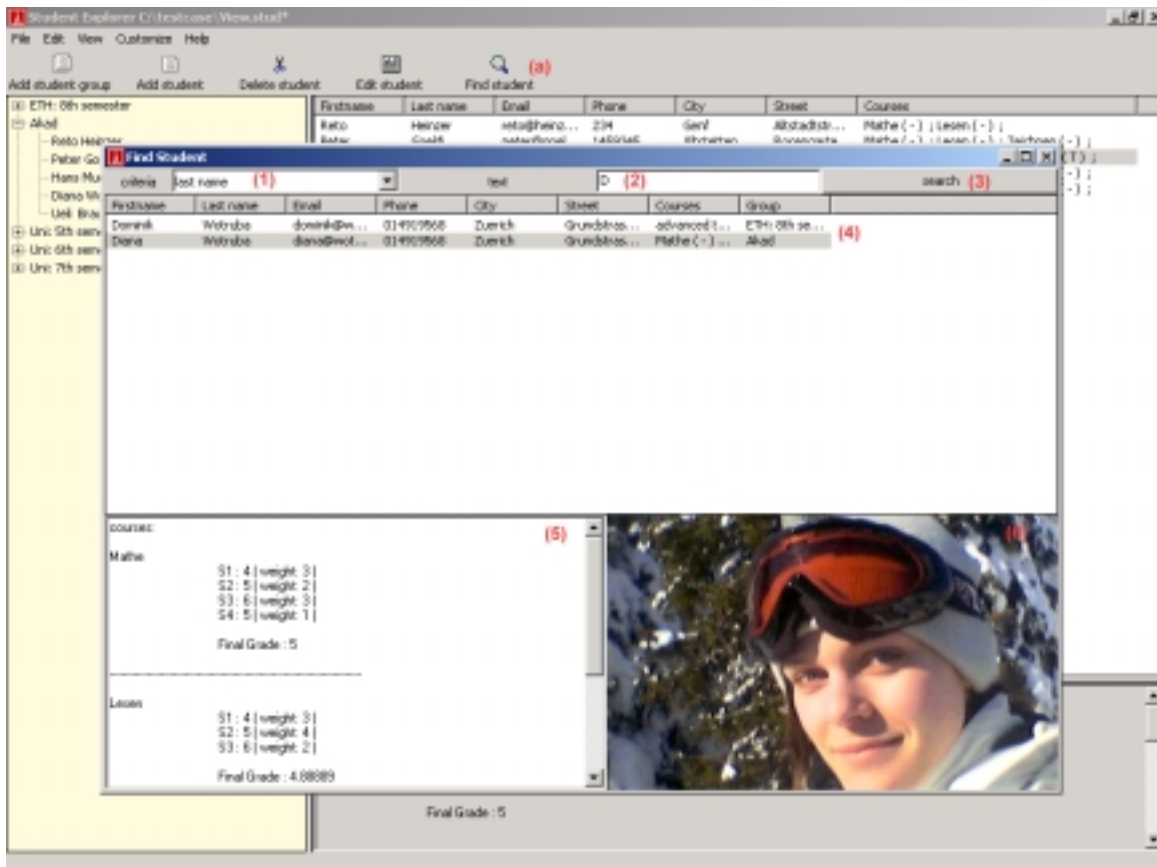


Figure 7. The find student form

Search criteria

Student Explorer currently supports following search criteria:

- Student number: To find the student with the desired number
- First name: To find the student with the desired first name
- Last name: To find the student with the desired last name
- Email: To find the student with the desired email
- Phone: To find the student with the desired phone number
- City: To find the student with the desired city
- Street: To find the student with the desired street
- Group: To find all students which belong to a desired group
- Course: To find all students which attend the demanded course

How to find a student

The desired search criteria (1) have to be selected and the search text has to be entered in the text field (2).

There is also support for abbreviations. For example if there is a need to search for somebody whose first name is “Hanspeter” the search string can be “Hans”.

Pressing the “Search” button (3) starts the requested search; the results are displayed in the labelled multi-column list (4). The content resembles the multi-column list in the main window, except that there is an additional row for student groups.

Pressing on a list item (4) displays additional information in the text area (5) and the picture area. (6).

The file management

There are two ways to store files in Student Explorer. Pressing the “Export” menu item in the menu “File” displays the form to export to XML. Choosing the “Save” menu item in the same menu displays the form to save to “.stud” format (see Figure 8). Selecting the “Import” item opens the form to import from XML. Pressing the “Open” item displays the form to retrieve a file from the “.stud” format.

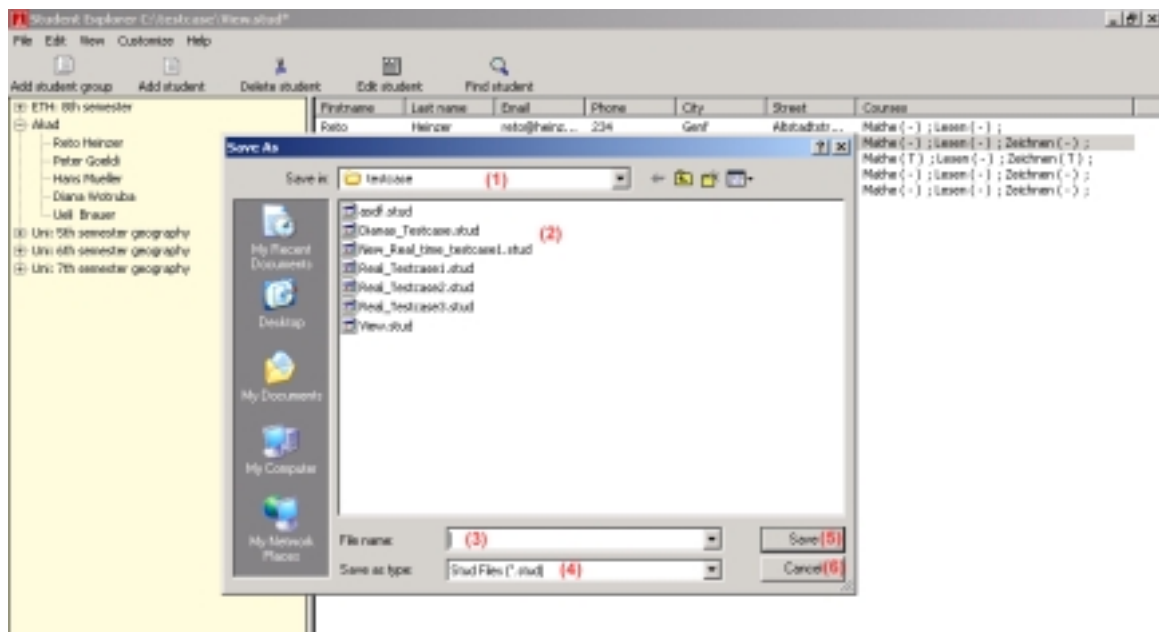


Figure 8. The Save dialog

Advantage and disadvantage of the .stud format

The first possibility is to store students in the “.stud” format. Storage in this format has the advantage that it provides fast retrieval, but has the disadvantage of a proprietary file format, which can change from version to version.

Advantage and disadvantage of XML format

The second possibility, exporting to XML has the disadvantage that it is slower in retrieval. The advantage is that the XML format is human readable and might stay unchanged or change only slightly in future versions. Furthermore it can be edited or expanded using a simple editor. Interesting is also the ability of using style sheets. Using them it will be possible to transform the XML documents or parts of the XML documents into HTML or PDF documents.

Start with a new document

To start with a new form press the new menu item.

Exit

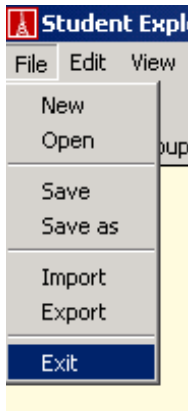


Figure 9. The Exit menu item

To quit Student Explorer, press the “Exit” item in the “File” menu (Figure 9). If changes have been made to the document, the “Save student” form will appear before Student Explorer terminates, otherwise Student Explorer will just quit. Note: if you press “Cancel” in the “Save student” form the program will terminate immediately without saving your changes.

Help

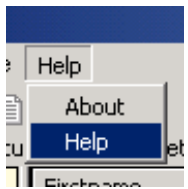


Figure 10. The help menu item

Pressing the “Help” menu item in the “Help” menu (Figure 10) shows a short instruction to Student Explorer.

4. THE IMPLEMENTATION

The classes implementing Student Explorer can be grouped into four categories:

- The model, including all classes, for the student groups, students, courses and grades.
- The view, which contains all classes for the GUI.
- The controller, which is the connector between the model and the view.
- The persistency group, which handles persistence.

Every group and corresponding classes will be thoroughly discussed in the following sections.

The model

The model includes a student root, student groups, students, courses, grades and grade types, represented by the classes `STUDENT_ROOT`, `STUDENT_GROUP`, `STUDENT`, `COURSE`, `GRADE`, `GRADE_TYPE` and `MODEL`, which will be explained in detail in this section.

The student root

The student root is represented by the class `STUDENT_ROOT`, which includes a `LINKED_LIST [STUDENT_GROUP]`. The student root is a list of student groups of type `STUDENT_GROUP`.

Student groups

The student group is represented by the class `STUDENT_GROUP`, which includes a `LINKED_LIST [STUDENT]`. The student group is a list of students of type `STUDENT`. Student root, student groups and students can also be seen as a tree, in which the student root forms the root, the student groups are the intermediate layer, and students are the leaves.

Students

As its name suggests, the class `STUDENT` represents a student. Every student has a unique student number, a first name, a last name, a birth date, an email address and a location (country, city, street), participates in zero or more courses, can be reached by a telephone number or a mobile number, belongs to a semester and can even have a photo. There can also be supplementary information, stored as comments.

The following features can access all student data as discussed above: “semester”, “number”, “firstname”, “lastname”, “birthdate”, “email”, “country”, “city”, “street”, “course_list”, “phonenum1”, “phonenum2”, “comment”, “photo”.

All of them are used by the application with exceptions of “country” and “phonenum2”, but both might be used in future versions. Furthermore there are features to compare the set values. They are used by the class `FIND_STUDENT`, which

tests, for example, whether the name, which has been set, resembles the name specified in the search text field of this class.

Courses and grades

Furthermore every student participates in zero or more courses. This fact is modelled by the “course_list” reference, see picture below (Figure 11). Every course has zero or more grades represented by the “grade_list” and every grade has a type, represented by the class GRADE_TYPE.

Uniqueness of grade types.

To guarantee the uniqueness of grade types for a specific course, every course has a “grade_type_list”. Every “grade_type”, which is used in a course, is registered in the “grade_type_list” to ensure that only grade types are inserted, which have not been registered so far.

Uniqueness of courses and student numbers

The model includes a course list, a student number list and a student root. The role of the course list and the student number list is to guarantee uniqueness. Every course has a unique course name and every student has a unique student number. Before a student or a course can be inserted, it has to be checked first if the student or the course are not already registered.

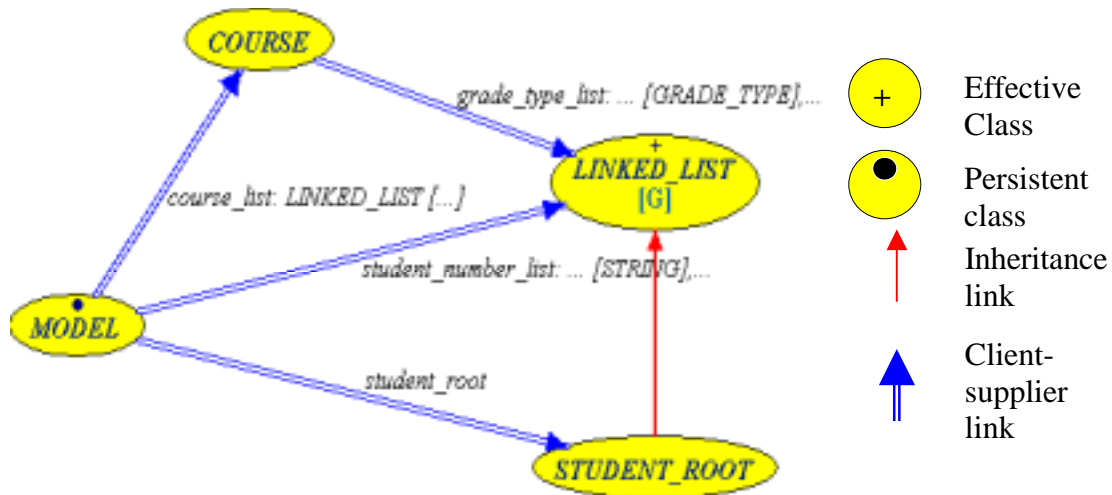


Figure 11 Class diagram of the “Model” cluster

A closer look at courses

Besides the features discussed above, a course can be testified using the feature “testat”, which has the feature “final_grade” and a list of grades called “grade_list”.

A closer look at grades

A grade consists of a value and a grade type. The value is represented by the feature “value”, the grade type by the feature “type”.

A closer look at grade types

A grade type consists of a name and a weight. The value is represented by the feature “value” and the grade type name by the feature “name”.

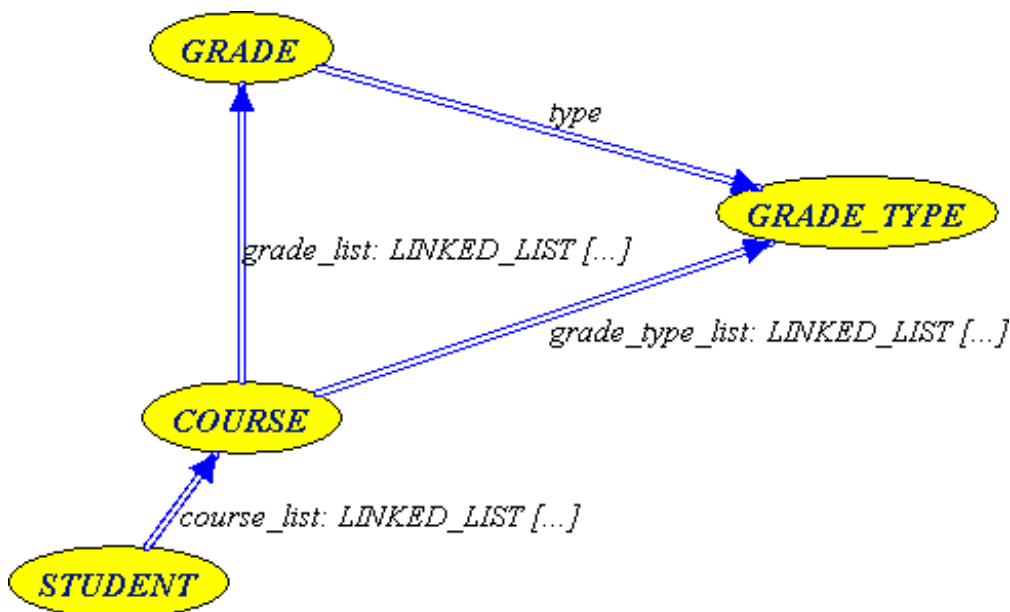


Figure 12. A closer look at the “Model” cluster (courses, course grades and types)

The class MODEL

The class MODEL provides features to add and remove student groups, students, courses and student numbers. As seen in the diagram below (Figure 13), the model inherits the class STORABLE, which enables the model to be serialized. Furthermore it has a reference, called “student_root”, which makes possible to access all student groups, students and their attached courses with all exams and grades. This entry point is needed for different queries. An example of such a query can be found in the class MANAGE_COURSE in which the grade type for a desired course is changed using the feature “edit_type”.

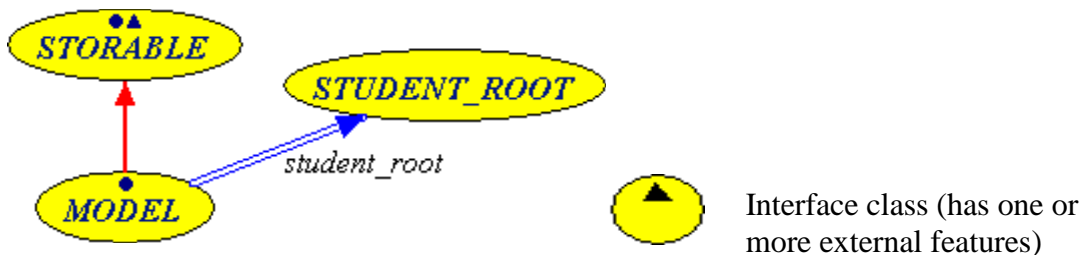


Figure 13. The class model (and related classes)

Managing student groups

Student groups are inserted to or removed from the “student root”. There is also a feature to move a student from one group to another, called “move_student_to_new_group”. The role of the feature “wipe_out” is to clear the whole model.

XML-Export

All classes discussed in this section except the class MODEL provide features to serialize the class to XML. These features have the name “emitx”

The view

There are two kinds of classes representing the forms used by Student Explorer. One category includes classes generated by EiffelBuild [2], second category correspond to the corresponding implementation classes. Besides there are classes like COURSE_NOTE_BOOK_ITEM, COURSE_LIST_ITEM, GRADE_TYPE_LIST_ITEM, STUDENT_GROUP_TREE_ITEM, STUDENT_MULTI_COLUMN_LIST_ROW, STUDENT_TREE_ITEM representing parts of the forms.

Class names ending with IMP

The class name of a generated class ends with “IMP”. Generated classes are the following: ADD_STUDENT_IMP, EDIT_STUDENT_IMP, MANAGE_COURSE_IMP, MANAGE_STUDENT_GROUP_IMP, FIND_STUDENT_IMP and MAIN_WINDOW_IMP. They can be edited and regenerated opening the corresponding “.bpr” file using EiffelBuild [2]. Rebuilding them has to be performed in another directory than the current one where the Student Explorer project resides, because the class implementing the “IMP” class has to be overwritten.

Functionality behind graphical representation

The core classes are those of the second category, since they provide the functionality behind the graphical representation. They are called: ADD_STUDENT, EDIT_STUDENT, MANAGE_STUDENT, MANAGE_STUDENT_GROUP and FIND_STUDENT and MAIN_WINDOW.

Feature groups in common

These classes have some features in common. Except the FIND_STUDENT class, they all have a feature clause “Update” with a feature “update”, which gets called whenever the form is updated. Since the find student form does not have to be updated, there is no such feature.

All classes except MAIN_WINDOW have a feature clause “Basic operations”, containing public features exported to other classes. For example in class ADD_STUDENT, this clause contains a reference to the controller, since only the controller provides the ability to add students to the model.

All classes except the FIND_STUDENT class have a feature clause “Element change”, containing features that have the ability to change the state of Student Explorer, for example by adding or editing items. Removal of various items is handled by the feature clause “Removal”.

All classes except FIND_STUDENT and MAIN_WINDOW have a feature clause “Status report”, containing queries for status retrieval (the status being calculated from the registered information).

All classes provide a clause “Implementation” with various features, which are not visible from the outside (information hiding).

The class ADD_STUDENT

This class manages the insertion of new students and their corresponding courses. Its features are grouped into several categories: “Initialisation”, “Update”, “Access”, “Basic operations”, “Element change”, “Removal”, “Status report” and “Implementation”.

The groups with feature clause “Initialisation”, “Update” and “Basic operations” contain the same kind of features as the classes managing other forms discussed in previous sections.

The feature group “Element change”

In the group with feature clause “Element change”, there are features to insert a new student, to insert a new course for a specific student and to add exams to a specific course. They are represented by the features called “add_student”, “add_course” and “add_grade_and_type”. These features are used as agents, called whenever the “Add student” button (respectively the “Add course” button) is pressed in the notebook item.

The feature group “Removal”

In the group “Removal” there are features to remove courses and exam types from a specific course. Both are used as agents, called whenever the “Remove course” button (respectively the “Remove” button which enables removing exam types) is pressed in the course notebook.

The feature group “Status report”

The group “Status report” has already been discussed in a previous section.

The feature group “Implementation”

In the group “Implementation” there are features to import pictures. The feature “import_picture” is used as an agent, which is called when the “Import button” is pressed.

The class EDIT_STUDENT

This class manages the form to edit students and their courses. It contains the same feature clause as the class ADD_STUDENT. The important groups are “Element change” and “Removal”.

The feature group “Element change”

In the group “Element change” there are features called “edit_student”, “add_grade_and_type” and “add_course”. As their names suggest, their role is to edit the form corresponding to a student, as well as to add new exams and courses.

The feature “edit_student” is used as an agent, called whenever the “Edit student” button is pressed. The feature “add_grade_and_type” is used as an agent called whenever the “Add” button is clicked. To insert exam types into the notebook, the item must have been pressed. The agent built from feature “add_course” is called, whenever the “add course” button gets pressed.

The feature group “Removal”

In the group “Removal” there are features used as agents to remove either an exam or the course. The feature “delete_course” is used to remove a chosen course from the student’s course list. This is made possible when the “Remove course” button has been pressed. The role of the “remove_grade_and_type” is to remove the exam type from the chosen course. This is made possible when the “Delete” button has been pressed.

The class MANAGE_STUDENT_GROUP

In this class the important feature groups are “Element change” and “Removal”.

The feature group “Element change”

In the feature group “Element change”, there are features to add and edit students. Both features are used as agents. The “add_group” feature is called whenever the “Add student group” button is pressed. The role of this feature is to insert the group to the model.

Note: Since the student group name is unique, it is only inserted when it has not previously been registered.

The “edit_group” feature is called whenever the “Edit student group” button is pressed and renames the group.

The feature group “Removal”

In the feature group “Removal” one can find the “delete_group” feature, which removes a student group from the model.

The class MANAGE_COURSE

This class resembles the class MANAGE_STUDENT_GROUP. The features “add_group”, “edit_group”, “delete_group” have exactly the same role as in the class STUDENT_GROUP, except that this class manages student groups instead of individual students.

Managing course types

There are additional features to manage types. The features “add_type”, “edit_type”, “remove_type” are used as agents. The “add_type” agent is called whenever the “add type” button gets pressed. This agent inserts a type only, if the type name with the same value doesn’t exist. The “edit_type” agent is called whenever the “Edit type” button gets pressed. This agent changes the name and weight only if the chosen name and weight differ from the registered exam type. The “delete_type” agent is called whenever the “delete type” button gets clicked.

Updating the type list

In the feature group implementation, there is a feature called “update_type_list”. Its role is to update the type list according to the course, which has been selected.

The class FIND_STUDENT

The only feature of importance is “search”. Its role is to traverse all student groups and display them in the multi-column list, if they are corresponding to the search criteria. Every student represented by the class STUDENT has features of the form “has_x (v: like x): BOOLEAN”, which returns true if “x” resembles “v”.

The class MAIN_WINDOW

As the class name suggests, MAIN_WINDOW represents the main window, which appears when starting after the application. In the group “Initialization” there is a feature “user_initialisation”, which is called when the main window gets instantiated. This feature instantiate all forms, (i.e. the add student form, the edit student form, the manage student form, the manage student group form, the find student form) and all dialogs, (i.e. the about dialog and the help dialog). Furthermore the model, the controller and the render class, which are explained in this section later, are instantiated here.

The main window title

The feature group “Element change” contains the feature “set_changes”. It is called whenever the state of Student Explorer changes. The boolean value “changes” is used to handle the “*” beside the main window title.

File handling

In the “Implementation” group there are features to handle files, such as importing, exporting, opening and storing files. They are called: “new”, “open”, “save”, “save_as”, “store”, “store_as”, “load”, “import”, “my_import”, “export”, “export_as”, “my_export”.

Agents in main window

Most of the other features are used in agents. The agent “exit” is called whenever the “Exit” menu item gets selected or main window gets cancelled. The agent “find_student” causes the find student form to appear. The agent “add_student” causes the add student form to appear. The agent “add_student_group” opens the “add student group” form. The agent “edit student” is responsible for the edit student form. The agent “delete student” is used to delete students. The agent “all_students” shows all students in the labeled multi-column list. The agent “about” opens the about dialog. The agent “help” displays the help dialog.

The class COURSE_NOTE_BOOK_ITEM

This class represents a notebook item, which is shown inside the add student form and the edit student form. It has a feature group called “Basic operations” containing features to extend a grade and remove it from a chosen course. An important implementation detail is that both buttons “add” and “delete” are publicly accessible. There are accessed from classes `ADD_STUDENT` and `EDIT_STUDENT` to register the agent “add_grade_and_type” to the “add_button”, (respectively to register the agent “remove_grade_and_type” to the “delete” button).

Dialogs

There are two dialogs. `ABOUT_DIALOG` displays the version of Student Explorer and the name of its author. The `HELP_DIALOG` displays a small introduction. The next version of Student Explorer could extend it, with an improved help dialog, including an index and a search function.

List items

Since there are many lists in the program many lists, (for example the multi-column list, a course list, a grade list, a student group list), which must have at least one reference to different types, there is a class `LIST_ITEM [G]`, where G is of a generic reference type. There are two classes, which inherit `LIST_ITEM [G]`: the class `COURSE_LIST_ITEM`, which has reference to the class `COURSE_NOTE_BOOK_ITEM`, and the class `GRADE_TYPE_LIST_ITEM`, which has reference to the course. Both classes could be joined to form just one class `LIST_ITEM [G, H]`.

Tree items

There are two items representing the visual tree: `STUDENT_GROUP_TREE_ITEM` and `STUDENT_TREE_ITEM`. As the class names suggest `STUDENT_GROUP_TREE_ITEM` represents the student group and `STUDENT_TREE_ITEM` represents the graphical representation of the student leaves.

The class STUDENT_MULTI_COLUMN_LIST_ROW

The multi-column list row is represented by the class STUDENT_MULTI_COLUMN_LIST_ROW.

The class RENDER

This class includes all features, which are responsible for rendering, such as depicting the multi-column list row or displaying a tree item. If it is required that the student tree item only show the first name instead of first name and last name for example, this is the class, which has to be adapted.

The class INTERFACE_NAMES

This class contains string values including the messages for error handling. This class would facilitate translating of the program into different languages.

The controller

The controller, represented by the class CONTROLLER, is the connector between the model classes and the view classes. As shown on the diagram below (see Figure 14) controller can access the model directly. Besides the controller contains references to the view (classes like ADD_STUDENT, EDIT_STUDENT, MAIN_WINDOW) in order to update the view if necessary. The backward references are used for manipulations of the model. If for example it is required to add a student in the add student form, the following call has to be performed: “controller.extend_student (a_studentgroup, a_student)”. The feature in the controller calls “model.extend_student (a_studentgroup, a_student)” and updates the add student form. This indirection enables storing only data in the model.

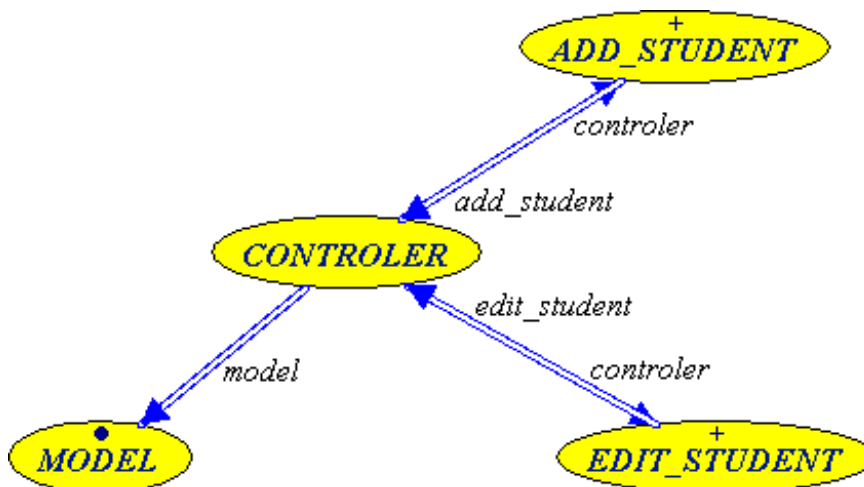


Figure 14. Class diagram of the “Controller” cluster

Persistency

Student Explorer supports two forms of persistency. First, it supports serialization of the model, by using the class STORABLE (compare with Figure 13), which is inherited by the model. Second it supports storing data in a XML file to retrieve data. A XML-File is generated by traversing all student groups, students, courses and grades.

From a programmer's point of view it is very easy with the first approach, since the class STORABLE inherited by the class MODEL implements the feature "retrieved", which can read the serialized objects from a file.

Using the second approach requires changes in the DEF_XML_FILTER class, (more precisely in the features), writing in the XML-File.

6. TESTCASES

Student Explorer has been tested with three test cases using the XML files "testcase1.xml", "testcase2.xml", "testcase3.xml", produced with Student Explorer. "Testcase1.xml" models the course "Compiler Construction", "testcase2.xml" models a "Psychiatry Course" and "testcase3.xml" contains an example, modelling the group of Prof. Bertrand Meyer.

7. FUTURE EXTENSIONS

An interesting feature would be to support reading student data from the ETH-database. Thanks to the unique student number it would be possible to retrieve the entire stored data of the particular student.

Another interesting feature would be to support multi-user with privileges.

8. REFERENCES

- [1] Chair of Software Engineering: *Semester-/Diplomarbeiten*; Online at: , consulted in October 2002.
- [2] EiffelSoftware. Documentation about EiffelStudio and EiffelBuild retrieved February 2003 from <http://www.eiffel.com>
- [3] Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.