Criterion B: Design

As mentioned in criterion A, the idea behind the program is to create a user-friendly grading application. After consulting with the client and forming the prototype, I decided that the program was going to have a central form with the main menu, from where the client would be able to perform basic CRUD operations (such as creating an object, updating or deleting it) or print the files she chose. For this to happen, the program will start by loading any files saved and loading their content on the memory. After that, the user will keep working on the memory, and when the program is closed, the data will be stored back into the files and saved on the hard disk.

In order to achieve this, sets, and more specifically HashSets were used. This way, double values are forbidden, which is crucial for the program, since it is based on a unique identification for every class instance, and they are sorted automatically. In addition, simple text files were used, since it takes less time for them to be parsed and saved and creating a backup is easy. Also, the number of students is so small, that a SQL database would be too much; the burden of maintenance and the fact that there is no need for transactions since the program is a single user app also adds to that. Finally, I chose delimiter text files, because there is no need for implementing comma separated files since they are too complicated for my needs, and because creating files with a standard word length is also too much for this case.

Furthermore, in order to make the program more user-friendly, the CRUD operations and the forms used follow a similar pattern; this uniformity means that training the client will take less time, and she will be able to use her experience to solve any problems that come up instead of having to find a different solution each time. In an attempt to be pragmatic, I also decided that there will be no undo for all destructive actions, but instead confirmation questions will appear every time the user chooses such an action. This way, the user will be able to “play around” in order to see what each operation does without accidentally destroying something important. Finally, when it comes to the printing commands, I reduced the problem of printing to an already solved one, by taking advantage the printing capabilities of browsers.

TABLES FOR DATA TYPES AND PROPOSED FORMS

➢ TABLE 1: STUDENTS

FIELD NAME	DATA TYPE	DESCRIPTION
studentID	integer	unique identification – primary key
name	string	student's first name
surname	string	student's last name
schoolClass	object	school class he belongs to

➢ TABLE 2: SUBJECTS

FIELD NAME	DATA TYPE	DESCRIPTION
subjectID	integer	unique identification – primary key
name	string	name of the subject

➢ TABLE 3: SCHOOL CLASSES

FIELD NAME	DATA TYPE	DESCRIPTION
schoolClassID	integer	unique identification – primary key
name	string	name of the class

➢ TABLE 4: GRADES

FIELD NAME	DATA TYPE	DESCRIPTION
student	object	student to whom the grade belongs to
subject	object	the grade is for which subject?
trimester	object	the grade is for which trimester?
mark	integer	how much did the student score?

➢ TABLE 5: TRIMESTERS

FIELD NAME	DATA TYPE	DESCRIPTION
trimesterID	integer	unique identification – primary key
name	string	name of the trimester

➢ TABLE 6: PROPOSED FORMS

FIELD NAME	DESCRIPTION
MainForm	all main operations are conducted from here

StudentForm	appears when user want to create or update a student
SubjectForm	appears when user want to create or update a subject
SchoolClassForm	appears when user want to create or update a school class
GradeForm	shows existing grades for a specific class, for a specific subject and for a specific trimester

Testing plan

Testing Scenarios	Nature of Test	Example
GUI forms work as expected	Tables, drop down lists, scroll pane work, clickable buttons etc.	Use the forms to ensure that they are functioning
CRUD operations work	Data in the files are stored correctly after being created, altered, deleted	Object will be created, updated and deleted, and then the appropriate files will be checked to check results
Confirmation questions work	Confirm cancellation, warn when data connected to other classes is to be deleted	CRUD operations will be performed and cancelled, students with existing grades will be deleted
Data validation	Data validation works when it is needed	Grade above/ below 20 will be inserted, along with characters and/or symbols
Correct averages	Averages are calculated correctly	Create students with four grades, calculate averages manually and check result
Correct alphabetical order	Records printed on the browser are in ascending alphabetical order	Print multiple students and check their order
Print only when data are available	In case there are not data saved, printing will not take place	Start the program with empty files and try to print

Evidence and results of the testing can be found in the Development section.

word count: 408 (because of extensive writing in the beginning)