CISP 1020

Group 1

Hanadi Mohamed

3/31/2018

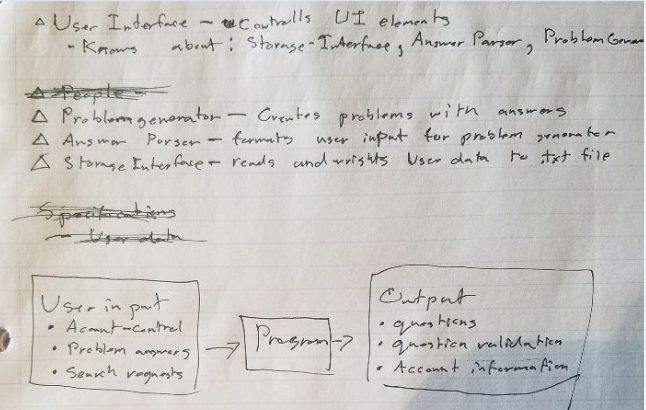
Term Project Report

Requirements specification:

We need to develop a program that:

* Features a user login that takes in a unique username as input.
* Features a new user dialog that allows new users to create an account that comes with role-specific features, whether student, teacher, or parent.
* Each role will have its own menu in the application.
* From the teacher menu, teachers will be able to search, sort, and list his/her students’ by name, score, or grade-level and have access to those students’ names, scores, and grade-levels. The teacher will also need the ability to specify the number of students he/she wants to supervise.
* From the student menu, students will be able to take grade-level specific quizzes with dynamically generated questions, including subtraction, addition, multiplication, and division, with feedback at the end. The student will also need the ability to specify his/her grade level to determine the difficulty of the problems generated.
* From the parent menu, parents will be able to access their students’ scores.
* The program will need to write the quiz results from the students to a text file to be read later.

Analysis process:



Design process:

Discovering classes:

After a noun search we came up with:

User, Login, UserName, Input, Dialog, Account, Role, Feature, Student, Teacher, Parent, Menu, Application, Name, Score, GradeLevel, Number, Quiz, Question, Subtraction, Addition, Multiplication, Division, Feedback, End, Abililty, and Difficulty.

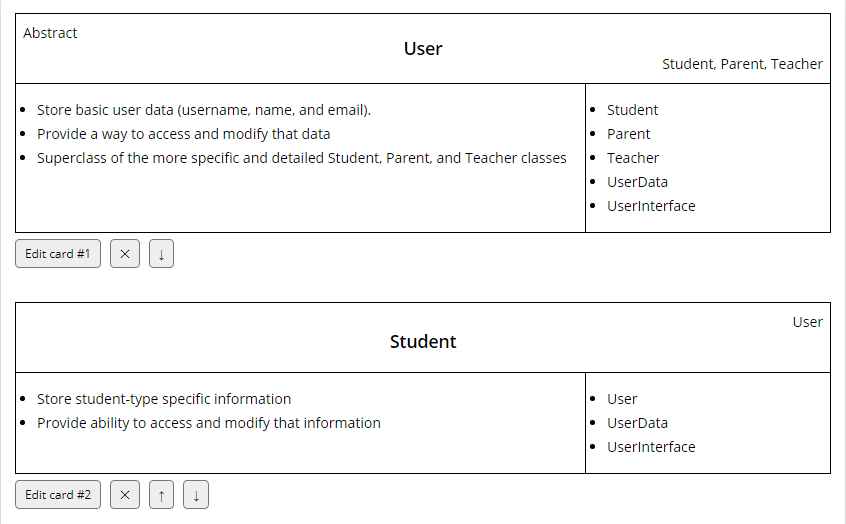
After deliberation we reduced this list to:

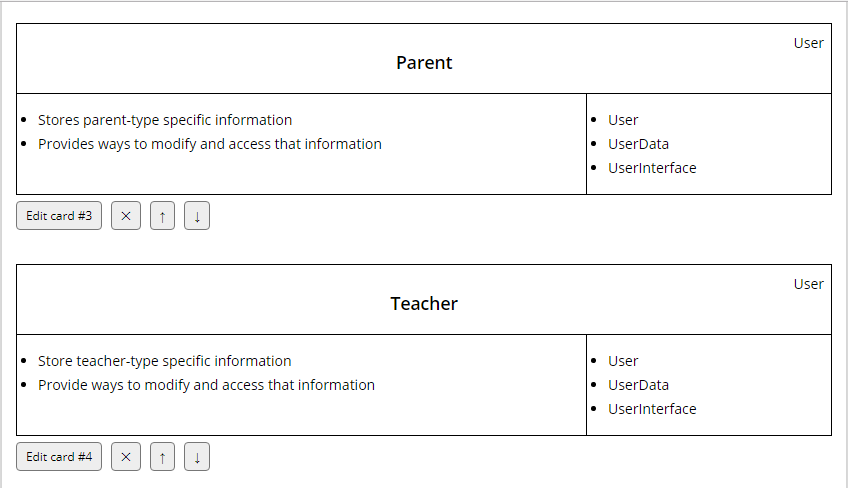
User, Teacher, Parent, and Student.

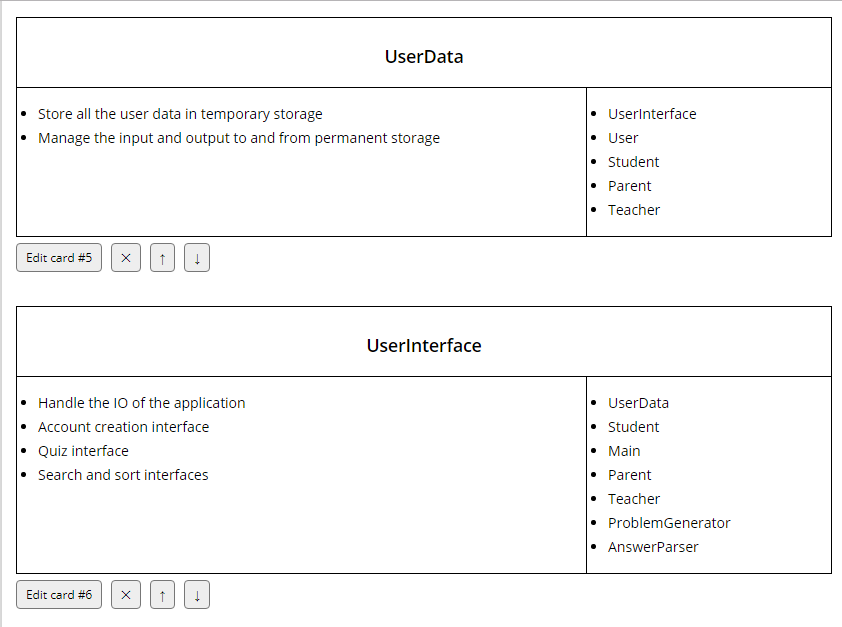
We added some additional (utility) classes:

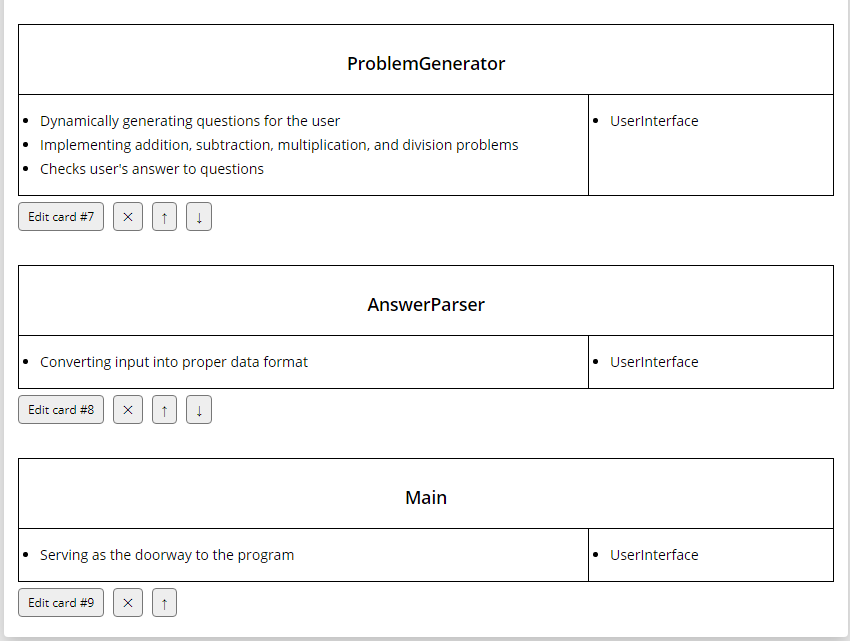
AnswerParser, ProblemGenerator, UserData, UserInterface, and lastly Main.

CRC Cards:

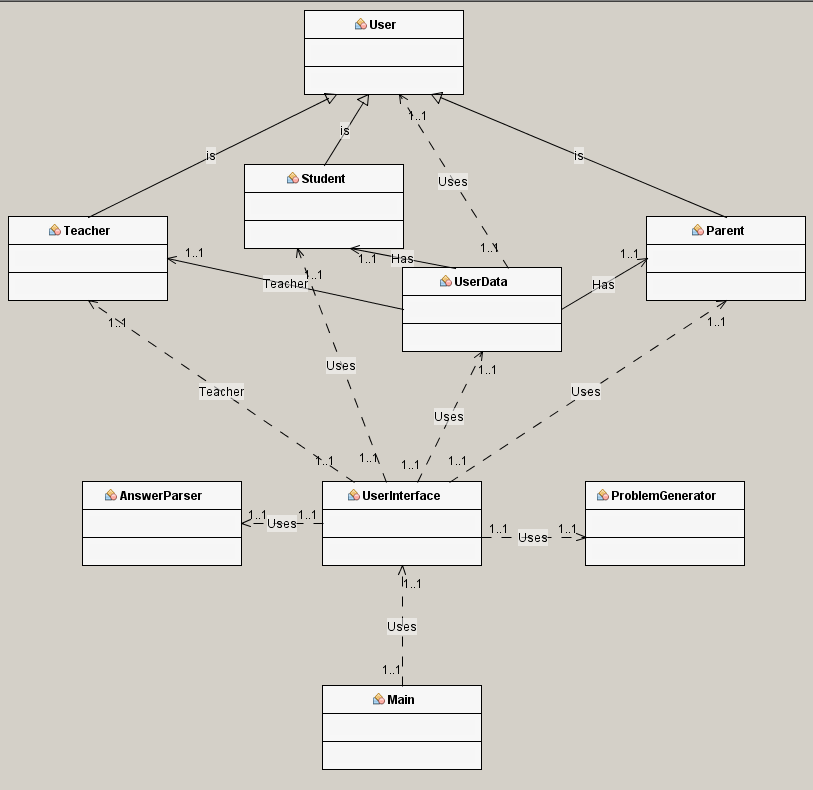








UML Diagram:



Implementation process:

From here we started by dividing up the labor. The AnswerParser class went to Anthony, the UserData class went to Johnathan, the ProblemGenerator class went to Maxmilian, and lastly, the UserInterface class and the User class, along with its subclasses, the Teacher, Parent, and Student classes, went to Stephen. Most of these roles and responsibilities did change and evolve as the project went on. These changes included the UserInterface class being co-written by Maxmilian and Stephen and the UserData class being co-written by Johnathan and Stephen.

The roles of classes changed as well. Originally, the UserData class was only going to manage the input and output from the permanent storage file only: There was going to be no temporary storage. We later changed our mind when we saw that the temporary storage was going to be easier to work with directly than the permanent storage; so, the UserData class naturally took on that new task. The UserInterface is perhaps the most dramatic example of change in this project, as we had to do a complete overhaul on that particular class, headed up by Maxmilian, to transfer functionality from what was originally going to be scores of dialog classes into dialog methods located inside of UserInterface. Our reasoning was that the dialog classes, needing to be able to transfer data to and from the UserInterface, created an enormous volume of dependencies and that we would be better off without them. Overall, the change streamlined the code, making it easier to manage and read.

Inheritance was yet another thing we had to consider, as previously mentioned we originally were going to keep all the data in the file strictly; but yet, another factor that changed our minds was when we had to remove the dialog classes along with their class hierarchy. We could not think of anything else besides the User class hierarchy to demonstrate inheritance after removing the dialog classes, it worked well though, because we already had built all of the User classes.

Testing process:

Rather late in the process we decided to do our testing with group 4, fortunately fitting them in just in time.