Project Document Template

Programming III – Fall 2022
Course Project: Topic

Team Information

Team Name: koisumii

Team members: Rita Youssef 2124602, Rania Mustafa 2256255

Project Description

Build a quiz about inheritance and interfaces. It's inspired by moodle quizzes, it's only multiple choices.
The user has to choose one option out of four, they can submit their answer right away to check if it was right and if it's wrong it says incorrect and displays the right answer. Then they can go to the next the question and then finish the quiz, at the end they get their results. Their results is saved in a file.

Development Approach

Explain how did you prepare for the project. You can use the 5 steps of algorithmic thinking to you help build this section (you will need to elaborate on each step).

1. **Understanding the problem.**
   //user answer questions
   Program is expected to display to the user a question that will be readed from a file with options and the user has buttons with submission that checks the answer, goes to the next question or finish the quiz then at the end an output with their results.
2. **Formulating the problem.**
   After getting inspired by moodle quizzes, the input will be radio buttons to select their options and buttons to submit and the output a window with their score.
3. **Developing the application \ algorithm.**
   Create a MainWindow with contents, buttons, textBlocks and radioButtons
   Activate them and implement their functionality
   Read Questions from a file
   Add feedback on correct and incorrect answers when submitted
   // If the answer is correct, show "Correct!" message
   // If the answer is incorrect, show the correct answer
   // User proceeds to the next question
   When Quiz Is finished, show results with score and give option to save the results in a file and restart the quiz.
4. **Implementing the application \algorithm.**
   Using C# and WPF
5. **Testing.**
   Tested the MainWindow first before creating the QuizResult window, the proceeded with testing the QuizResult Window.

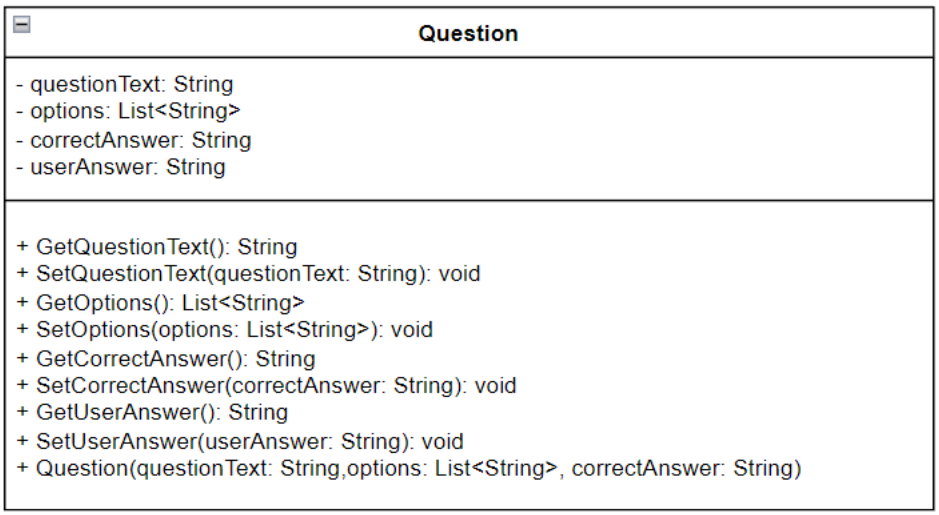
OOP Design

Question class, which is basically designated to model the structure and the behaviour of the questions in the quiz.

It contains four properties:

a. **QuestionText**: Gets and sets the text of the question and it ensures that the question text is not empty.
b. **Options**: Gets and sets a list of string options for the question and it allows modification of the options.
c. **CorrectAnswer**: Gets and sets the correct answer for the question and it ensures that the correct answer is a non-null value and is present in the options list.
d. **UserAnswer**: Gets and sets the user's selected answer for the question.

It contains a constructor that takes parameters for the question text, a list of options, and the correct answer and it initializes the properties.



Contributions

Rita	Rania
• Team Contract • Created the contents of the main window	• Modified the LoadQuestions method to check how many options are available for current question? then setting

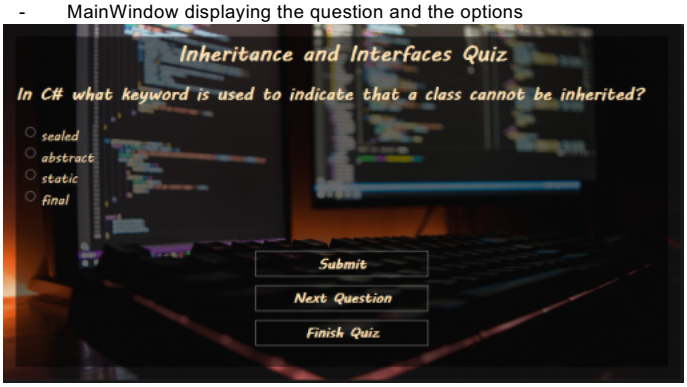
• Created A LoadQuiz Button with implemented method to activate it. • Implemented the MainWindow.xaml.cs: 1. A method to Initialize the UI all hidden. 2. A method to activate the LoadQuiz button. 3. A method to clear the feedback for the previous question. 4. A method that loads the questions that sets the text of the Question to the text of current question and clearing previous selections, no pre-selections for next questions. 5. A method SubmitButton_Clicked (object sender, RoutedEventArgs e) that shows if the when the user submits their answer, it's correct or incorrect and shows a message of the following and if incorrect shows the correct answer. 6. A method NextQuestionButton_Clicked(object sender, RoutedEventArgs e) that saves the user's answer and loads the new question. 7. A method GetSelectedAnswer() that gets the selected answer from the radio buttons. • Created the Question class (description in the OOP Design part) and implemented it. • The project document template.	content for each RadioButton. • Modified the method NextQuestionButton_Clicked(object sender, RoutedEventArgs e) finishes the quiz if the user chooses so and shows their final score • Created the method Btn_GotoQuizResultsWindowClicked(object sender, RoutedEventArgs e) that handles the click event for the button to go to the Quiz Results window. • Implemented the SaveUserAnswer method that saves the user's answer for the current question. • Added a save to file method. • Created a button to restart the quiz. • Created a method that will read a list of questions and their answers. • Created the contents of the Results window. • Implemented the QuizResultsWindow.xaml.cs: • A method ShowResults() that displays the quiz results on the UI, ensures every question has an answer to avoid exceptions, updates each question with the user's answer, sets the ItemsSource of the ListView to the list of questions and displays the final score in a TextBlock. • A constructor that initializes fields with data provided from the quiz • UML Diagram • Modified Team Contract

The work was fairly divided.

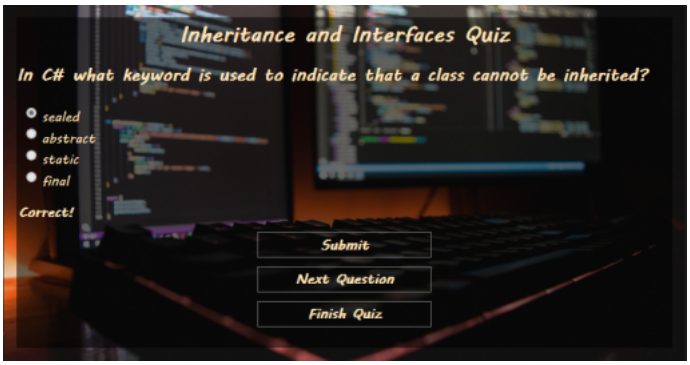
App Snapshots

- Click Button to Load the Quiz

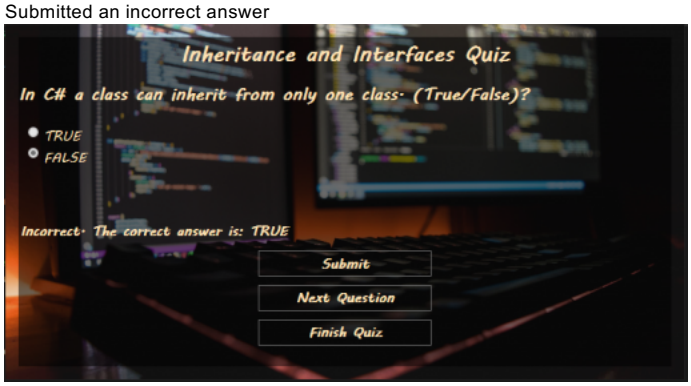
- MainWindow displaying the question and the options



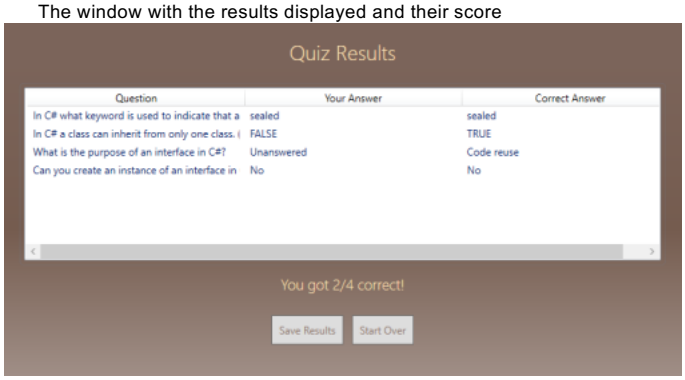
- Submits the answer to check if correct



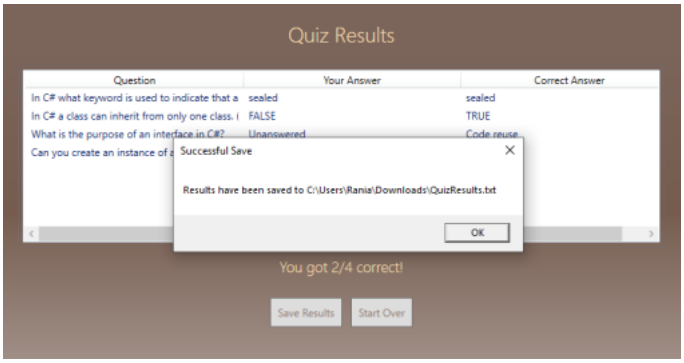
- Submitted an incorrect answer



- The window with the results displayed and their score



- Save Your answers in a file



QuizResults - Notepad

File Edit Format View Help

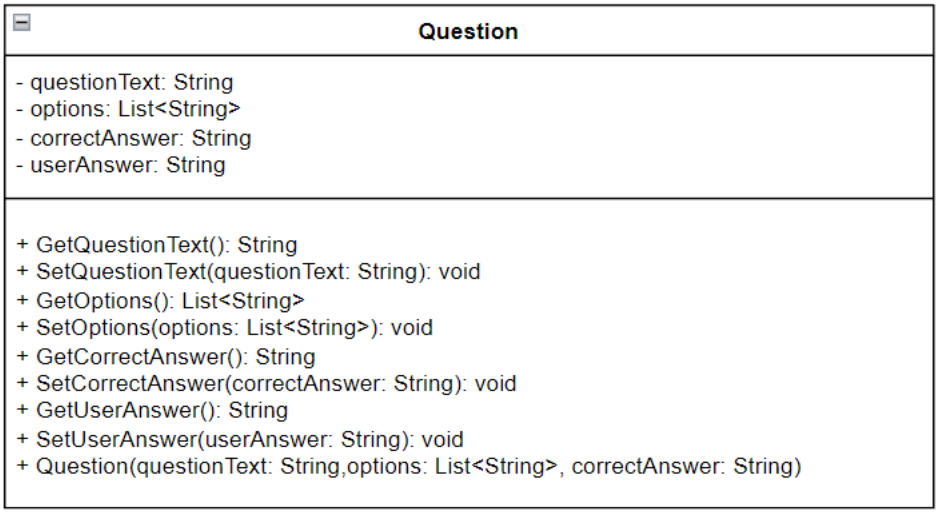
```
Question: In C# what keyword is used to indicate that a class cannot be inherited?, Your Answer: sealed, Correct Answer: sealed
Question: In C# a class can inherit from only one class. (True/False)?, Your Answer: FALSE, Correct Answer: TRUE
Question: What is the purpose of an interface in C#?, Your Answer: Unanswered, Correct Answer: Code reuse
Question: Can you create an instance of an interface in C#?, Your Answer: No, Correct Answer: No
Total Score: 2
```

Future Work

- Start project earlier.

Appendix A: Team Contract

Appendix B: UML Class Diagram


Question
- questionText: String
- options: List<String>
- correctAnswer: String
- userAnswer: String
+ GetQuestionText(): String
+ SetQuestionText(questionText: String): void
+ GetOptions(): List<String>
+ SetOptions(options: List<String>): void
+ GetCorrectAnswer(): String
+ SetCorrectAnswer(correctAnswer: String): void
+ GetUserAnswer(): String
+ SetUserAnswer(userAnswer: String): void
+ Question(questionText: String,options: List<String>, correctAnswer: String)