Object Design Document

Project Enlightenment

Computer Training For Visually Impaired Automation Tool (CTVIAT)

Prepared By: Project Enlightenment

Date: January 5, 2011

Version No: v6.0

Document Change Control

The following is the document control for revisions to this document.

Version Number	Date of Issue	Author(s)	Brief Description of Change
1.0	14/10/10	Project Enlightenment	Phase 1 - Preliminary draft for Phase 1
1.1	30/10/10	Project Enlightenment	Update for Phase 1 deliverables
1.2	30/11/10	Project Enlightenment	Update for Phase 1 deliverables
2.0	13/12/10	Project Enlightenment	Phase 1 & 2 – conform to new framework template, update for Phase 2 deliverables
3.0	28/12/10	Project Enlightenment	Phase 2 & 3 – conform to new framework template, update for Phase 3 deliverables
4.0	02/01/11	Project Enlightenment	Phase3 & 4 – conform to new framework template, update for Phase 4 deliverables
5.0	04/01/11	Project Enlightenment	Phase4 & 5 – conform to new framework template, update for Phase 5 deliverables
6.0	05/01/11	Project Enlightenment	Phase5 & 6 – conform to new framework template, update for Phase 6deliverables

Definition

The following are definitions of terms, abbreviations and acronyms used in this document.

Term	Definition
CTVI	Computer Training for Visually Impaired
RAD	Requirements Analysis Document
CTVIAT	Computer Training for Visually Impaired Automation Tool
JFW	JAWS for Windows
SDD	Software Design Document
ODD	Object Design Document
CPU	Central Processing Unit in a Computer

Table of Contents

1. INTRODUCTION

1.1 Object Design Trade-Offs

In our project, there is a trade-off between performance and complexity. Since, CTVI Automation Tool will provide many functionalities to users, there should be a short respond time when carrying out these functionalities to satisfy user expectations. Because, during the usage of the application, visually impaired people will be alone and must not wait too much during testing phase. In another word, the performance of the application is the main issue in our design. However, to satisfy this, we should decrease degree of complexity. In our previous design, we have decided to store such sound files in mp3 format in a directory and to keep the directory info for each sound file in the database. However, after our implementation goes further, we decided to use an API for this task, which comes with Visual Studio 2010 Packet and makes the application more efficient by cancelling out the need for sound files. Also, we increased the number of classes and subclasses. This increased the readibility of the program and if anyone reads the documentation of the project, he will easily be familiar with the main architecture of the project. Thus, in result, modifiying parts of project will be easier. However, all this will come with a cost: more complexity and low performance. Furthermore, our application should provide a safe platform to users to store their personal data more securely. This of course will be handled through a more reliable database management and independent design of subsystems. However, this will increase size of the code and in result require more memory.

1.2 Interface Documentation Guidelines

In our project, mainly, there are three packages are defined: Interface, Classes and Database. The Classes package containing the main classes used in the implementation of the CTVIAT application and in general each class consists of many subclasses. In Interface package, there are classes for login screen, admin screen and candidate screen and all classes responsible to handle interface operations. In Database package, there is a database access object class which is responsible to manage database connection and enable other classes to reach database through this class. This class includes User, Test, Question, Answer and Candidate-Test Medhods to handle Method related database operations.

Names of classes in the packages and class attributes are nouns starting with capital letters. If the name consists of more than one word, then the initial letter of each word is capitalized (e.g. TestsCandidates,IncompleteTestsCandidates,etc).

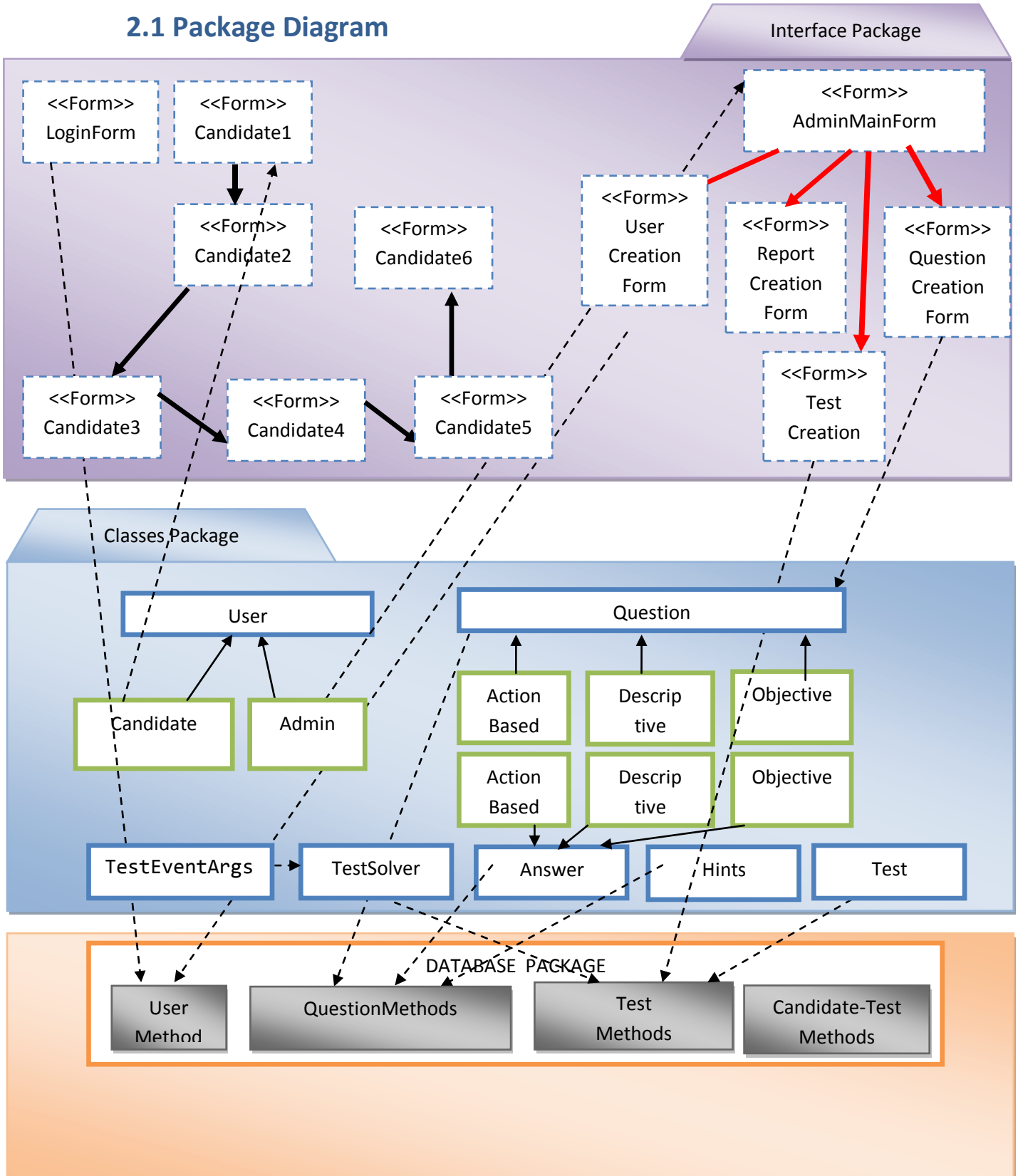
1.3 Definitions, Acronyms and Abbreviations

CTVI : Computer Training for Visually Impaired

CTVI AT : Computer Training for Visually Impaired Automation Tool

2. PACKAGES

2.1 Package Diagram



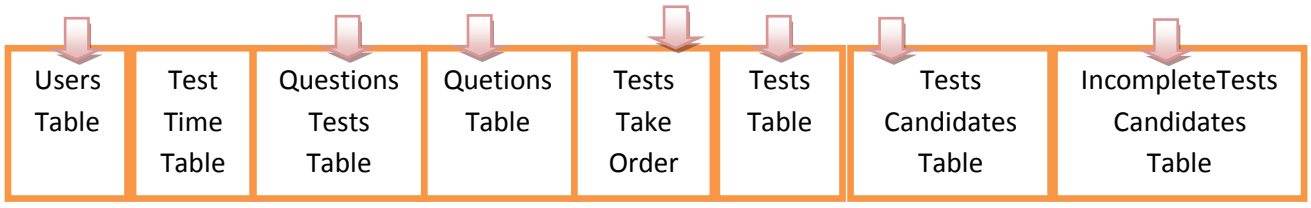


Figure 2.1 Package Diagram of the CTVIAT

There are three packages in object design model of CTVIAT. In Figure 2.1 these packages together with their classes shown.

2.2 Package Definitions

2.2.1 Interface Package

In this package, there are mainly three classes, named "LoginForm", "Candidate1", and "AdminMainForm" to control and manage all the tasks related to user interfaces. Because of the central role in the CTVIAT application, these classes have interactions with all other packages and with the database. Also, Interface package includes some Candidate and Admin related classes to handle user specific operations. For an admin "testCreationForm", "QuestionCreationForm", "ReportCreationForm" and "UserCreateForm" classes admin related operations like creating a test, question, user or report. For a candidate "Candidate2", "Candidate3", "Candidate4", "Candidate5", "Candidate6" classes present interfaces for candidate to undergoes an exam. This package has responsibility to enable users to use program interface efficiently. All classes in this package, takes inputs from users and according to these inputs users directed to related parts of program. In our previous design and implementations, many candidate related tasks were done on a single form window: "CandidateMainForm". However, since users of CTVIAT are mainly visually impaired ones and since CTVIAT application is dedicated to them, handling all tasks through one form window will be inefficient and useless for those users. To solve this issue, for each step of testing phase we redirected user to a related form. Thus, in result, our "CandidateMainForm" class became "Candidate1". For other windows we defined Instead we defined "Candidate2", "Candidate3", "Candidate4", "Candidate5", "Candidate6" classes. The detail will be given in "2.4 class definitions part".

2.2.2 Classes Package

On the top of this package, there are six main classes named "User", "Question", "Answer", "Hint", "Test" and "TestSolver". The "User" class is designed to control user related operations . Also, there are two sub-classes inherited from this class, namely "Candidate" and "Admin". The first one will be used to handle "Candidate1" class operations and the second one is designed to handle "AdminMainForm" operations. "Question" class enables admins to create questions in different types. To handle this, we will design 3 sub-classes inherited from this class: "ActionBased" class to create action-based questions, "Descriptive" class to create descriptive questions and "Objective" class to prepare objective questions. "Answer" class is designed to handle the answer part of the questions. For three kind of questions we defined three kind of answers:"ActionBasedAnswer", "Descriptive

Answer", "ObjectiveAnswer". "Hint " class is designed to supply hints to users when they are failed to do an operation during exam. "Test" class will be used to create test in different modules. Lastly, "TestSolver" class will be used to handle operations such as taking a text, re-taking a test, saving a test if user quits the test before finishing, all time related operations with a test, etc. For time issues of "TestSolver" class, we also implement a simple class called "TestEventArgs" which is just return a string .

2.2.3 Database Package

In this package there is our database tables and a class named "DatabaseObject" to enable other classes to connect to database. By implementing this class, all database related operations such as logging on to system, creating a test, creating a question, taking a test, keeping user info, etc handled more easily.

2.3 Class Diagrams

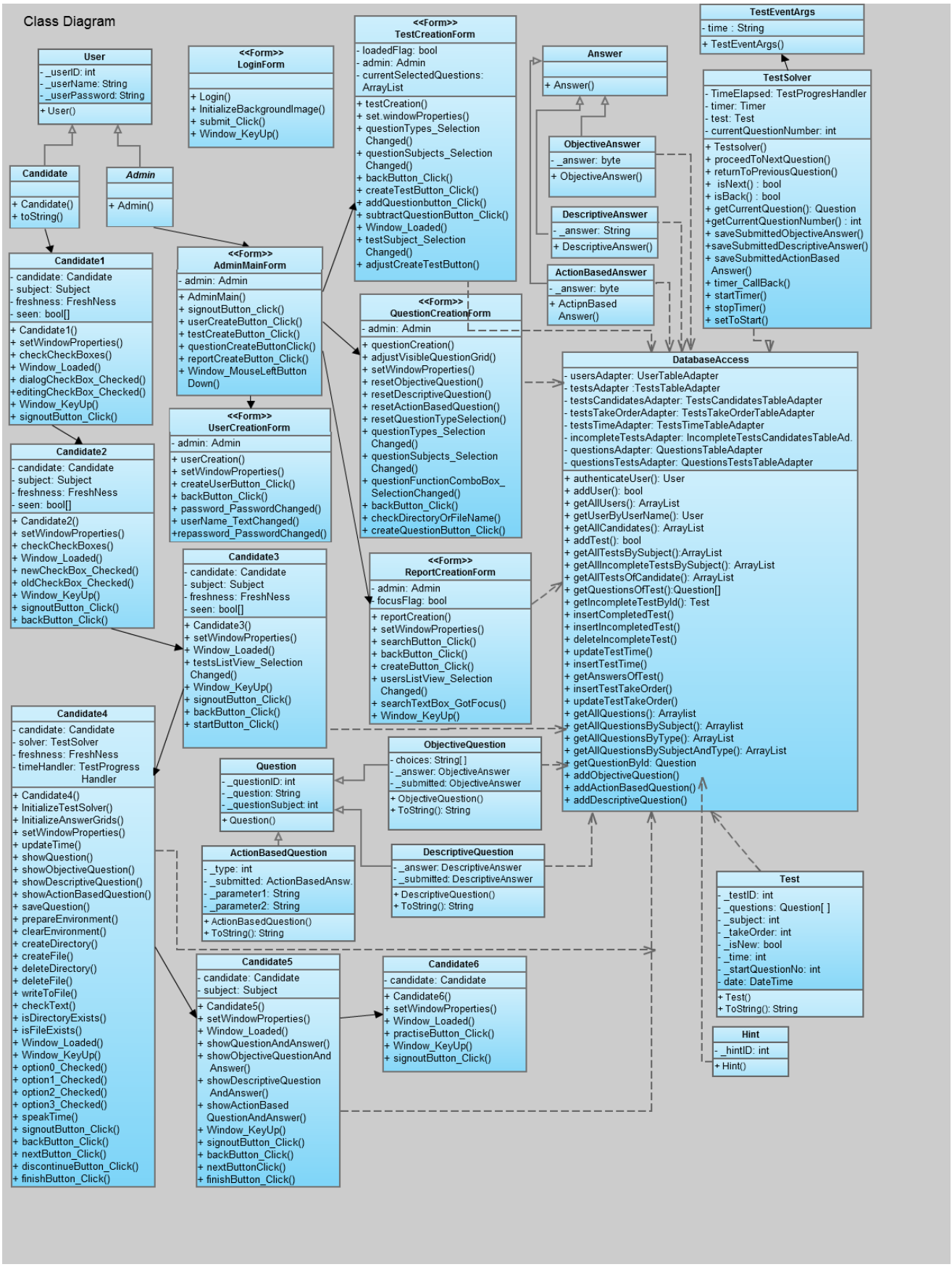
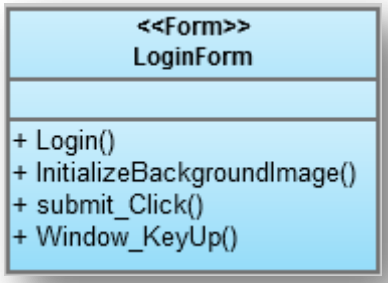


Figure 2.2 Class Diagram of the CTVIAT

The class diagram of the CTVIAT represents the classes, their attributes and methods, and the interactions between. Since we implement a lot, number of methods and members for each class incremented compared to old versions of ODD. Thus, in this version of the ODD, we changed our class diagram accordingly. (As the implementatin goes furher, number of classes and number of methods and members for each class increased. Therefore, to show all classes together with their methods and members on a single page class diagram, size of the diagram decreased. The same diagram with greater size can be found in SVN under directory “Design\ODD\Class Diagrams\ “)

2.4 Class Definitions

2.4.1 LoginForm

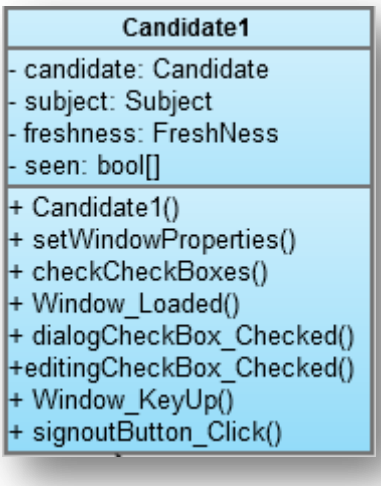


The LoginForm class is a part of Interface package and handles login operation of the users. This class create a database connection with user info through database access object class and check user. İf authorization succeeded, it hands over its job to User class by calling it. This class has 4 methods:

Methods:

- Login() : is the class constructor.
- InitalizeBackgroundImage(): this method sets a bitmap image as background image.
- Submit_Click(): when user pressed submit button in login screen this method called and authanticate of user starts.
- Window_KeyUp(): when user release a key this method capture the key.

2.4.2 Candidate1



The Candidate1 class is also a part of Interface Package and handles Visually Impaired users' Test Choice operation. In other words, candidates can select dialog or editing type tests by using this class. it has four members and eight methods:

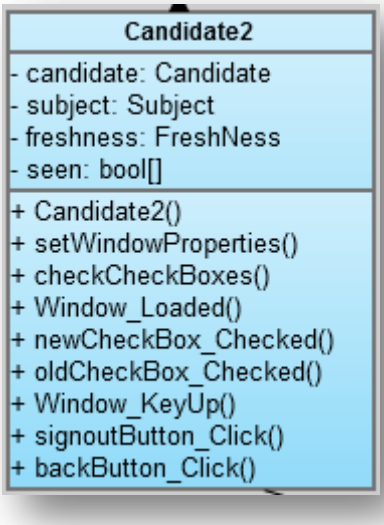
Members:

- candidate: Candidate object
- subject: an enumerator that is used to set subject of the test (none, dialog, editing)
- freshness: an enumerator that specify whether test is firstly taken or not (none, new, old)
- seen: boolean array that specify whether the window appeared or not

Methods:

- Candidate1(): this method is class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- checkCheckBoxes(): for users choices, related checkbox automatically checked
- dialogCheckBox_Checked(): subject of the test sets to be "Dialog"
- editingCheckBox_Checked(): subject of the test sets to be "Editing"
- signoutButton_Click(): when user pressed sign Out button, user session terminates
- Window_Loaded(): when window loaded, the API for speech informs the user for selections
- Window_KeyUp(): when user released a key this method capture the key

2.4.3 Candidate2



The Candidate2 class is part of Interface Package and enable visually impaired users to take a new test or an old test based on the selected test subject. it has four members and nine methods:

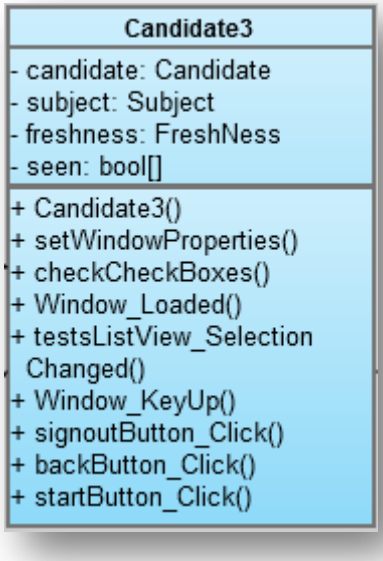
Members:

- candidate: Candidate object
- subject: an enumerator that is used to set subject of the test (none, dialog, editing)
- freshness: an enumerator that specify whether test is firstly taken or not (none, new, old)
- seen: boolean array that specify whether the window appeared or not

Methods:

- Candidate2(): this method is class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- checkCheckBoxes(): for users choices, related checkbox automatically checked
- newCheckBox_Checked(): if user select to take a new test, related check box checked
- oldCheckBox_Checked():if user select to take an old test, related check box checked
- signoutButton_Click(): when user pressed sign Out button, user session terminates
- Window_Loaded(): when window loaded, the API for speech informs the user for selections
- Window_KeyUp(): when user released a key this method capture the key
- backButton_Click(): by pressing back button user can go to main menu.

2.4.4 Candidate3



The Candidate3 class is part of Interface Package and enable visually impaired users to select a test with specified questions from the test list. it has four members and eight methods:

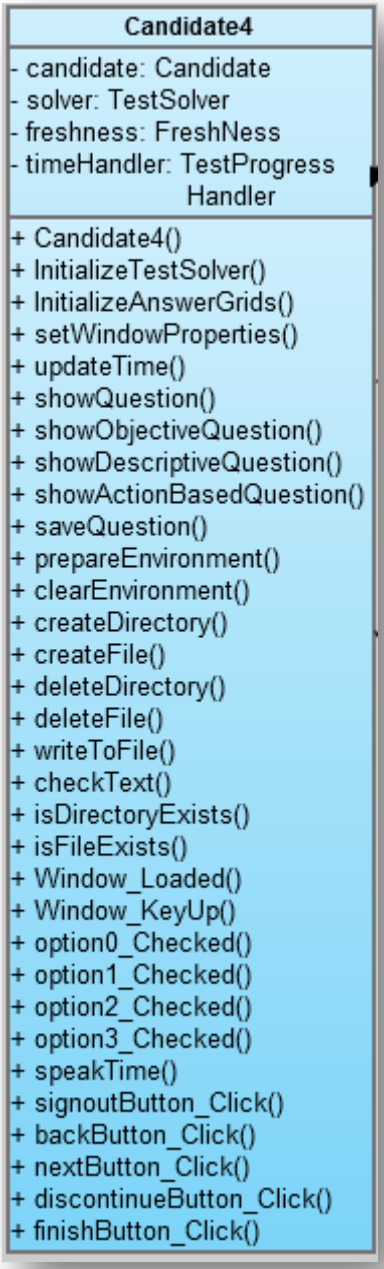
Members:

- candidate: Candidate object
- subject: an enumerator that is used to set subject of the test (none, dialog, editing)
- freshness: an enumerator that specify whether test is firstly taken or not (none, new, old)
- seen: boolean array that specify whether the window appeared or not

Methods:

- Candidate3(): this method is class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- testListView_SelectionChanged(): if user select a test from the list this method called
- signoutButton_Click(): when user pressed sign Out button, user session terminates
- Window_Loaded(): when window loaded, the API for speech informs the user for selections
- Window_KeyUp(): when user released a key this method capture the key
- backButton_Click(): by pressing BACK button user can go to previous menu(Candidate2 Form).
- startButton_Click(): by pressing START button, user can start to take selected test.

2.4.5 Candidate4



The Candidate4 class is also a part of Interface Package and enable visually impaired users to take the selected test. By the way, a timer runs and keeps time passed for the test. The user can pass away between questions by pressing next or back buttons. İf user quits the test before finishing, the program store that test. İf user in a later time wants to continue the test, he/she can continue where he left. This class has four members and twenty-one methods:

Members:

- candidate: Candidate object
- solver: TestSolver object

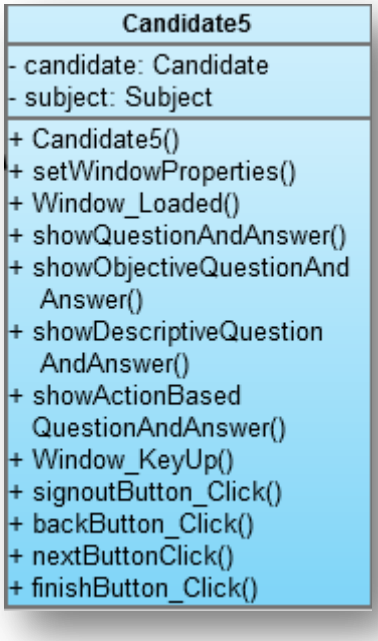
- freshness: an enumerator that specify whether test is firstly taken or not (none, new, old)
- timeHandler: TestProgressHandler object that keeps the test time.

Methods:

- Candidate4(): this method is class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- initalizeTestSolver(): initalize “solver” object
- initalizeAnswerGrids(): hiddens objective answer and descriptive answer grids for readibility.
- updateTime(): updates time label to show passed time.
- showQuestion(): for the type of question calls related method to prints the question on screen
- showObjectiveQuestion(): for objective type questions prints the question text and four options.
- showDescriptiveQuestion(): for descriptive type questions prints the question text
- showActionBasedQuestion(): for action based type questions prints the question text
- saveQuestion(): if user quits the test or pass to another question, the answer for current question stored by this method
- Option0_checked(): for objective type questions if user select first answer, option0 checkBox checked.
- Option1_checked(): for objective type questions if user select second answer, option1 checkBox checked.
- Option2_checked(): for objective type questions if user select third answer, option2 checkBox checked.
- Option3_checked(): for objective type questions if user select fourth answer, option3 checkBox checked.
- speakTime(): passed time for the test, speeched by the API to user.
- signoutButton_Click(): when user pressed sign Out button, user session terminates
- Window_Loaded(): when window loaded, the API for speech informs the user for selections
- Window_KeyUp(): when user released a key this method capture the key
- backButton_Click(): by pressing BACK button user can go to previous questions.
- nextButton_Click(): by pressing NEXT button, user can can go to next questions.
- discontinueButton_Click(): when user press QUIT button this method called
- finishButton_Click():when user press FINISH button this method called
- prepareEnvironment(): according to type of action based question creates necessary directory or files
- clearEnvironment(): according to type of action based question deletes unnecessary directory or files
- createDirectory(): creates a directory with given name
- createFile():creates a file with given name
- deleteFile():if exists , delete the file with given name

- deleteDirectory():if exists , delete thedirectory with given name
- writeToFile(): writes the given text to specified file
- checkText(): controls the text inside a file
- isDirectoryExists(): returns a boolean value based on existence of a directory
- isFileExist(): returns a boolean value based on existence of a file

2.4.6 Candidate5



The Candidate5 class is part of Interface Package and enable visually impaired users to control their answer after finishing the test. it has two members and eleven methods:

Members:

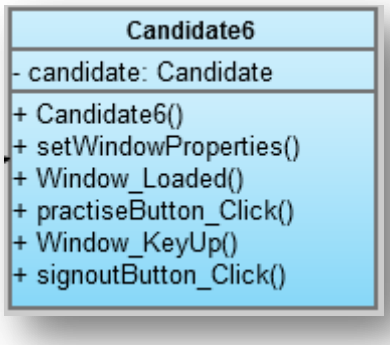
- candidate: Candidate object
- subject: an enumerator that is used to set subject of the test (none, dialog, editing)

Methods:

- Candidate5(): this method is class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- showQuestionAndAnswer():for the type of question calls related method to prints the question and answer to screen.
- showObjectiveQuestionAndAnswer():for objective type questions prints the question text and correct answer and user choice.
- showdescriptiveQuestionAndAnswer():for descriptive type questions prints the question text , correct answer and user answer.

- showActionBasedQuestionAndAnswer(): for action based type questions prints the question text and correctness of the answer
- testListView_SelectionChanged(): if user select a test from the list this method called
- signoutButton_Click(): when user pressed sign Out button, user session terminates
- Window_Loaded(): when window loaded, the API for speech informs the user for selections
- Window_KeyUp(): when user released a key this method capture the key
- backButton_Click(): by pressing BACK button user can go to previous questions.
- nextButton_Click(): by pressing NEXT button, user can can go to next questions.
- finishButton_Click():when user press FINISH button this method called

2.4.7 Candidate6



The Candidate6 class is part of Interface Package and enable visually impaired users to continue practising or terminating their session. it has one member and six methods:

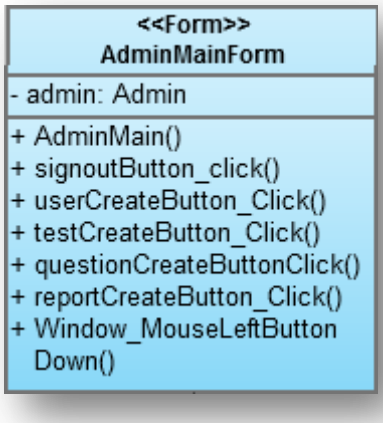
Members:

- candidate: Candidate object

Methods:

- Candidate6(): this method is class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- signoutButton_Click(): when user pressed sign Out button, user session terminates
- Window_Loaded(): when window loaded, the API for speech informs the user for selections
- Window_KeyUp(): when user released a key this method capture the key
- practiseButton_Click(): by pressing “PRACTISE” button candidate can undergoes anoather exam.

2.4.8 AdminMainForm



The AdminMainForm class is a part of Interface Package and handles admins' operations. This class has one member and six methods. This class will be used to enable admins to create users of type admin or candidate, to create questions and tests for visually impaired users and to create reports on candidates' performance:

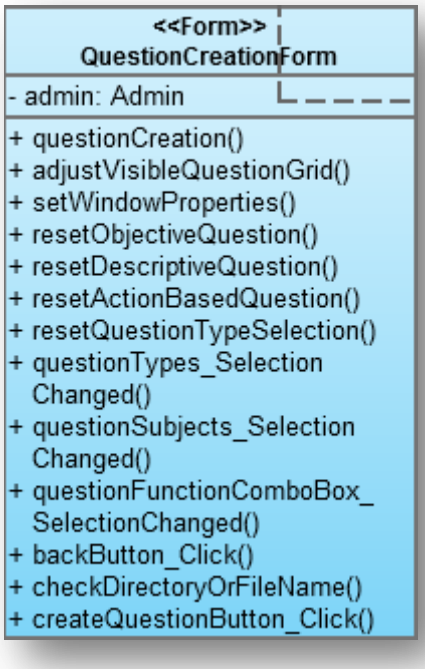
Members:

- admin: Admin object

Methods:

- AdminMain(): this method is class constructor and enbles to create AdminMain type objects.
- signoutButton_Click(): when user pressed sign Out button, user session closed.
- userCreateButtonclick(): open User Creation Form
- testCreateButtonClick(): open Test Creation Form
- questionCreateButtonClick(): open Question CreationForm
- reportCreateButton_Click(): open ReportCreationForm
- Window_mouseLeftButtonDown(): enable dragging the AdminMainForm

2.4.9 QuestionCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has seven methods and one members. It enables admins to create questions in diffrent type and subject for candidate users. This class needs a database connecton to complete its tasks.

Members:

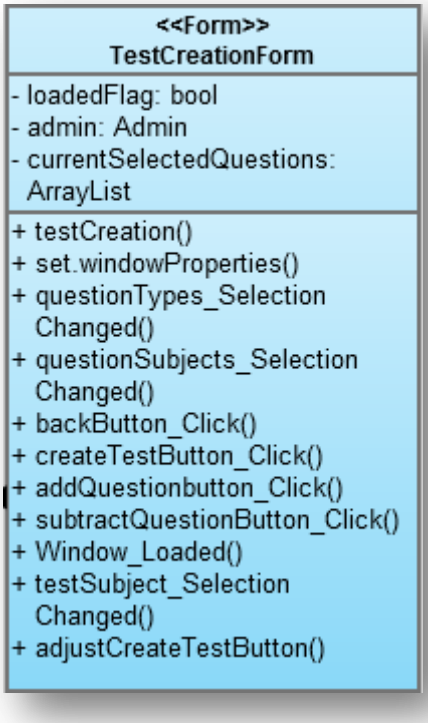
- admin: Admin object

Methods:

- questionCreation(): is the class constructor.
- adjustVisibleQuestionGrid(): make hidden or visible Question Grid for readibility
- questionTypes_SelectionChanged(): if admin chose a question type from drop down list this method called.
- questionSubjects_SelectionChanged():if admin chose a question subjectfrom drop down list this method called.
- resetObjectiveQuestion(): when user creating questions, if a descriptive or action based question chosen, the information about objective question, if entered, cleared by this method
- resetdescriptiveQuestion(): when user creating questions, if a objective or action based question chosen, the information about descriptive question, if entered, cleared by this method
- resetActionBasedQuestion(): when user creating questions, if a descriptive or objective question chosen, the information about action based question, if entered, cleared by this method
- resetQuestionTypeSelection(): it clears the question type and subject information

- questionfunctioncomboBox_SelectionChanged(): when admin creates action based questions, he should select a type from questionFunctionCombobox. The related events handles in this method
- checkDirectoryOrFileName(): for a given string this method controls whether the string is a directory, file or not.
- backButton_Click(): when admin pressed back button he goes back to AdminMain Form screen.
- createQuestionButton_Click(): when admin pressed create question button this method creates the question with specified info.
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode

2.4.10 TestCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has 3 members and 11 methods and enable admins to create tests for the visually impaired users. All operations run under a successful database connection.

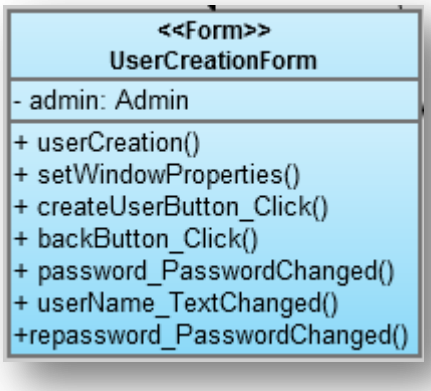
Members:

- loadedFlag:when form loaded this flag sets to true
- admin: Admin object
- currentSelectedQuestions: an ArrayList object to store selected questions

Methods:

- testCreation(): is the class constructor.
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- questionTypes_SelectionChanged(): when user select a new question type from drop down list this method called.
- questionSubjects_SelectionChanged(): when user select a new question subject from drop down list this method called.
- backButton_Click(): when admin pressed back button he goes back to AdminMain Form screen.
- createTestButton_Click() :when admin pressed create test button this method creates the test with specified info.
- addQuestionbutton_Click(): this method enables to add a question to test
- subtractQuestionButton_Click(): this button enables to substract a question from the test.
- Window_Loaded(): sets loaded flag true
- testSubject_SelectionChanged(): when admin select a new test subject from drop down list this method called.
- adjustCreateTestButton():

2.4.11 UserCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has 1 member and 7 methods and enable admins to create candidate/admin type users. All operations run under a successful database connection:

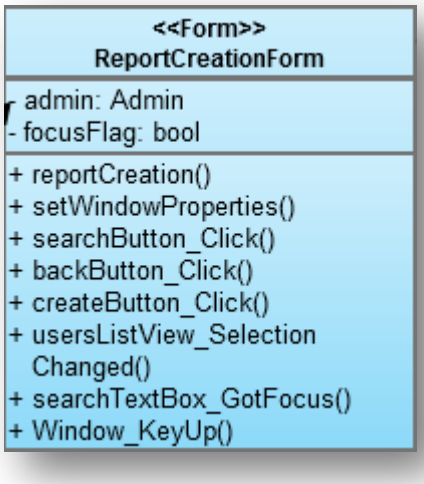
Members:

- admin: Admin object

Methods:

- userCreation(): is the class constructor
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- createUserButton_Click(): when admin pressed the create user button a new user with specified info created.
- backButton_Click(): enable admin to go back to AdminMain Form
- password_PasswordChanged(): checks user password and activate “create user button”
- userName_TextChanged():checks user name and activate “create user button”
- repassword_PasswordChanged():checks user password and activate “create user button”

2.4.12 ReportCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has 2 membesr and 8 methods and enable admins to create candidate related reports:

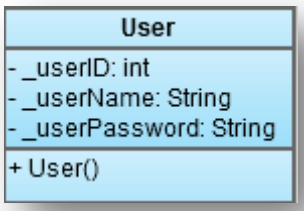
Members:

- admin: Admin object
- focusFlag: when admin wants to search a candidate this flag sets to true

Methods:

- reportCreation(): is the class constructor.
- setWindowProperties() : when the form loaded this method makes the program to run in full-screen mode
- searchButton_Click():when admin wants to search a candidate, he/she enters the name of candidate to searchTextBox and the click search button. After that, this method caled and specified candidate queried in database.
- backButton_Click(): when admin pressed back button he goes back to AdminMain Form screen.
- createButton_Click(): by clicking CREATE button, a candidate related repors created in PDF format
- usersListView_SelectionChanged(): when admin choose a candidate from the list CREATE button activated
- searchTextBox_GotFocus(): sets focusFlag true
- Window_KeyUp(): when user release a key this method capture the key.

2.4.13 User



The User class is in Classes Package and handles users related oprations . This class has three members and one method:

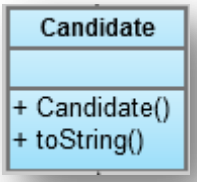
Members:

- userID : specify the user ID to bring data from database.
- userName: store users name info
- userType: specify whether user is an candidate or an admin

Methods:

- User() : is the constructor of the class.

2.4.14 Candidate



This class is inherited from User class and will be used to handle candidate related operations. It has 2 methods.

Members:

Methods:

- toString(): return username
- Candidate():is the constructor of the class.

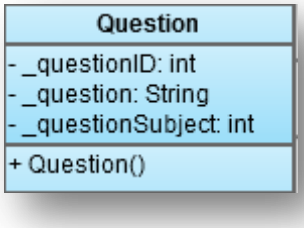
2.4.15 Admin

This class is also inherited from User class and will be used to handle admin related operations together with adminMainForm class. It has one method.

Methods:

- Admin(): is the class constructor

2.4.16 Question



This class is also in Classes Package and designed to encapsulete the data of questions read from database. It has three members and one method:

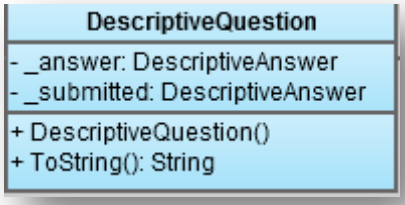
Members:

- _quetionID : keeps primary key of question table
- _question: keeps the question info
- _questionSubject: specify whether the question is objective,desciptive or action-based.

Methods:

- Question(): is the class constructor

2.4.17 Descriptive Question



This class is inherited from Question Class and designed to encapsulate the data about descriptive question type. It has 2 members and two methods:

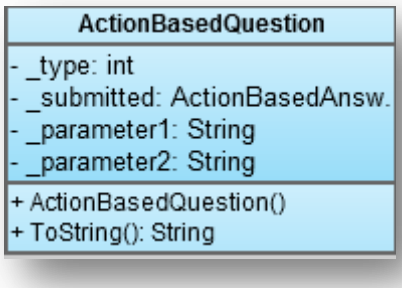
Members:

- _answer: DescriptiveAnswer object
- _submitted: DescriptiveAnswer object

Methods:

- DescriptiveQuestion(): is the class constructor.
- toString(): overrides toString method for descriptive type questions.

2.4.18 ActionBasedQuestion



This class also inherited from Question Class and designed for action-based type questions. It has 2 methods and four members:

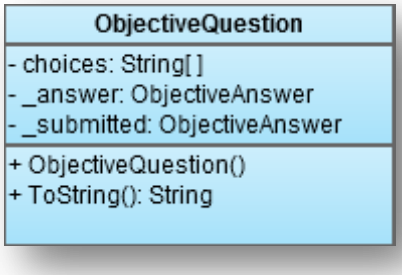
Methods:

- ActionBasedQuestion(): is the class constructor
- toString(): overrides toString method for action based type questions.

Members:

- _type: specify whether the action based question will be directory creation/deletion, file creation/deletion, etc.
- _submitted: specify whether question created or not
- _parameter1:
- _parameter2:

2.4.19 ObjectiveQuestion



This class also inherited from Question Class and designed for objective type questions. It has 3 members and 2 methods:

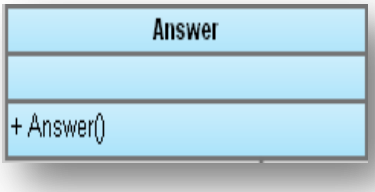
Members:

- choices: a string array that store the user choices
- _answer: ObjectiveAnswer object
- _submitted: specify whether question created or not

Methods:

- ObjectiveQuestion(): is the class constructor.
- toString(): overrides toString method for objective type questions.

2.4.20 Answer

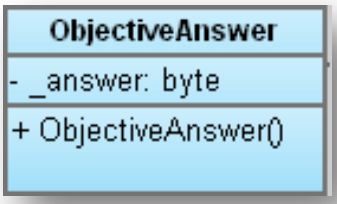


This class is in Classes Package and designed to encapsulete the data of answer for a spesific question. It has one method:

Methods:

- Answer(): is the class constructor

2.4.21 ObjectiveAnswer



This class is in Classes Package and inherited from Answer class and designed to encapsulete the data of objective type answer for objective type questions. It has one method and one member:

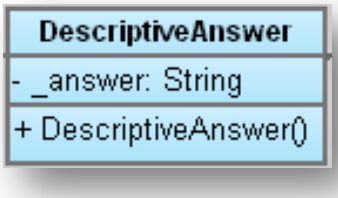
Members:

- _answer : keeps the answer info as byte

Methods:

- ObjectiveAnswer(): is the class constructor

2.4.22 DescriptiveAnswer



This class is in Classes Package and inherited from Answer class and designed to encapsulete the data of descriptive type answer for descriptive type questions. It has one method and one member:

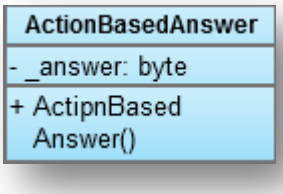
Members:

- _answer : keeps the answer info as String

Methods:

- DescriptiveAnswer(): is the class constructor

2.4.23 ActionBased Answer



This class is in Classes Package and inherited from Answer class and designed to encapsulete the data of actionbased type answer for actionbased type questions. It has one methodand one member:

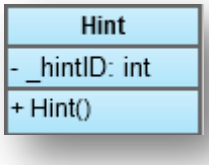
Members:

- _answer: keeps the answer info correct or wrong as byte

Method:

- ActionBasedAnswer(): is the class constructor

2.4.25 Hint



This class is in Classes Package and designed to encapsulete the data of hints for a spesific question. It has one method and one member:

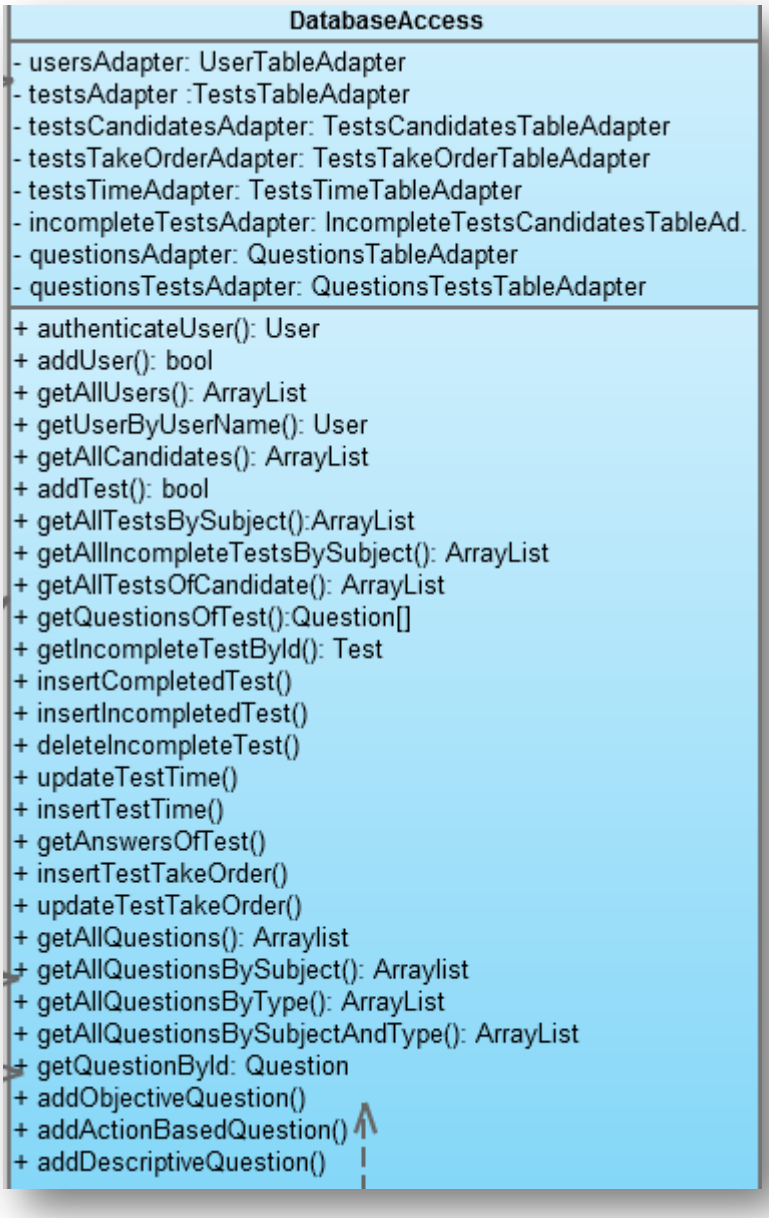
Members:

_hintID: int variable that keeps primary key of Hint table

Methods:

- Hint(): is the class contructor

2.4.26 Database Access Object Class



This class is a part of Database Package and enable all other classes to connect database efficiently. To handle database connection operation we implemented eight members and twenty-seven methods for this class:

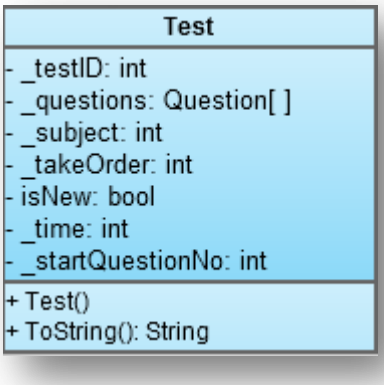
Members: // each following member is the table adapter definitions for each table in database

- UserAdapter
- testAdapter
- testsCandidatesAdapter
- testsTakeOrderAdapter
- testsTimeAdapter
- incompleteTestAdapter
- questiosnAdapter
- questionsTestsAdapter

Methods:

- authenticateUser(): authenticate users as Admin or Candidate.
- addUser(): enable adding a user to database.
- getAllUsers(): brings the information about all users.
- getUserByUserName(): brings user data for a specified name.
- getAllCandidates():
- addTest(): enable adding a new test to database.
- getAllTestsBySubject(): For a specified test subject brings all the test info.
- getAllIncompleteTestsBySubject(): brings data of all incompleted tests for a specified subject.
- getAllTestsOfCandidate(): for a candidate, this method brings all test taken by the canidate
- getQuestionsOfTests(): this method brings the questions of atest
- addIncompleteTestByID(): for a specified test ID, adds data of an incompleted test to database
- insertCompletedTest (): insert a completed test info into database
- insertIncompletedTest (): insert a incompleted test info into database
- deleteIncompletedTest (): deletes information of an incompleted test
- updateTestTime(): this method updates test time of a candidate for a spesific test in database
- insertTestTime(): this method insert test time of a candidate for a spesific test into database
- getanswerOfTest(): this method brings the answers of a test
- InsertTestTakeOrder(): test taking order info inserted into database for a candidate and a tets
- updateTestTakeOrder(): test taking order updates after every re-taking the test by the same candidate
- getAllQuestions(): brings the information about all questions.
- getAllQuestionsBySubject(): brings the information about all question for a specified subject.
- getAllQuestionsByType(): brings the information about all question for a specified type.
- getAllQuestionsBySubjectAndType: brings the information about all question for a specified subject and type.
- getQuestionById(): brings the information about all question for a specified question ID.
- addObjectiveQuestion(): add data of an objective type question.
- addDescriptiveQuestion():add data of a descriptive type question.
- addActionBasedQuestion():add data of an action based type question

2.4.27 Test



This class is also in Classes Package and designed to encapsulete the data of Test read from database. It has seven members and two methods:

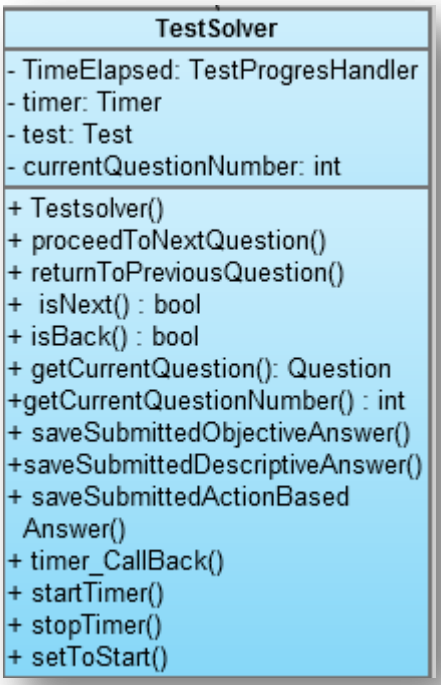
Members:

- _testID: keeps the primary key of the test in database
- _questions: keeps the information about question of the tests in a Qustion array
- _subject: keeps the subject of the test as int
- _takeOrder: specify how many times a candidate took a test
- _isNew: boolean value that specify test is firstly taken or not
- startQuestionNo: specify the question number such that when user start to take the test related question will be shown.
- _time: keeps the time information of the test

Methods:

- Test(): is the class constructor
- toString(): overrides toString method

2.4.28 TestSolver



This class is in Classes Package and designed to carry out all test solving related issues at the candidate side. When candidate start to solve a test this class keeps time of test, enable user to go through questions by pressing next question / previous question buttons and if user quits a test before finishing, it store the time of the test, given answers, current question number,etc for later use. İf candidate wants to finish test at a later time or wants to re-take the same test, with the help of this stored info, program easily handle above issues. It has four members and 12methods:

Members:

- timeElapsed: keeps the left time for each test
- timer: a Timer class object that handles time operations
- test: Test class object that keeps data about current test
- currentQuestionNumber: store the current Question number for later use if user quits the test

Methods:

- TestSolver(): is the class constructor
- proceedToNextQuestion(): increments “currentQuestionNumber”, if it is not the last question
- returnToPreviousQuestion():decrements “currentQuestionNumber”, if it is not the first question
- isNext(): checks whether current question is the last question or not
- isBack():checks whether current question is the first question or not
- getCurrentQuestion() : bu using currentQuestionNumber index, finds the current Question
- getCurrentQuestionNumber(): returns currentQuestionNumber
- savesubmittedObjectiveAnswer(): store the submitted answers for objective questions
- savesubmittedDescriptiveAnswer(): store the submitted answers for descriptive questions

- savesubmittedActionBasedAnswer(): store the submitted answers for action based questions
- timer_CallBack(): updates the time of test
- startTimer(): starts Timer
- stopTimer(): stops Timer
- setToStart(): sets current question number to zero.

2.4.29 TestEventArgs

This class is in Classes Package and designed and used to handle time related operations in Testsolver class by returning Time as string. It has one members and one method:

Members:

- time: a string object

Methods:

- TestEventArguments(): is the class constructor.