Object Design Document

Project Enlightenment

Computer Training For Visually Impaired Automation Tool (CTVIAT)

Prepared By: Project Enlightenment

Date: January 2, 2011

Version No: v4.0

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

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. One solution is to keep size of sound files that inform the visually impaired user about the parts and usage of the program. 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 to make the application more efficient and to decrease size of sound files. Also, we will increase the number of classes and subclasses. This will increase 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

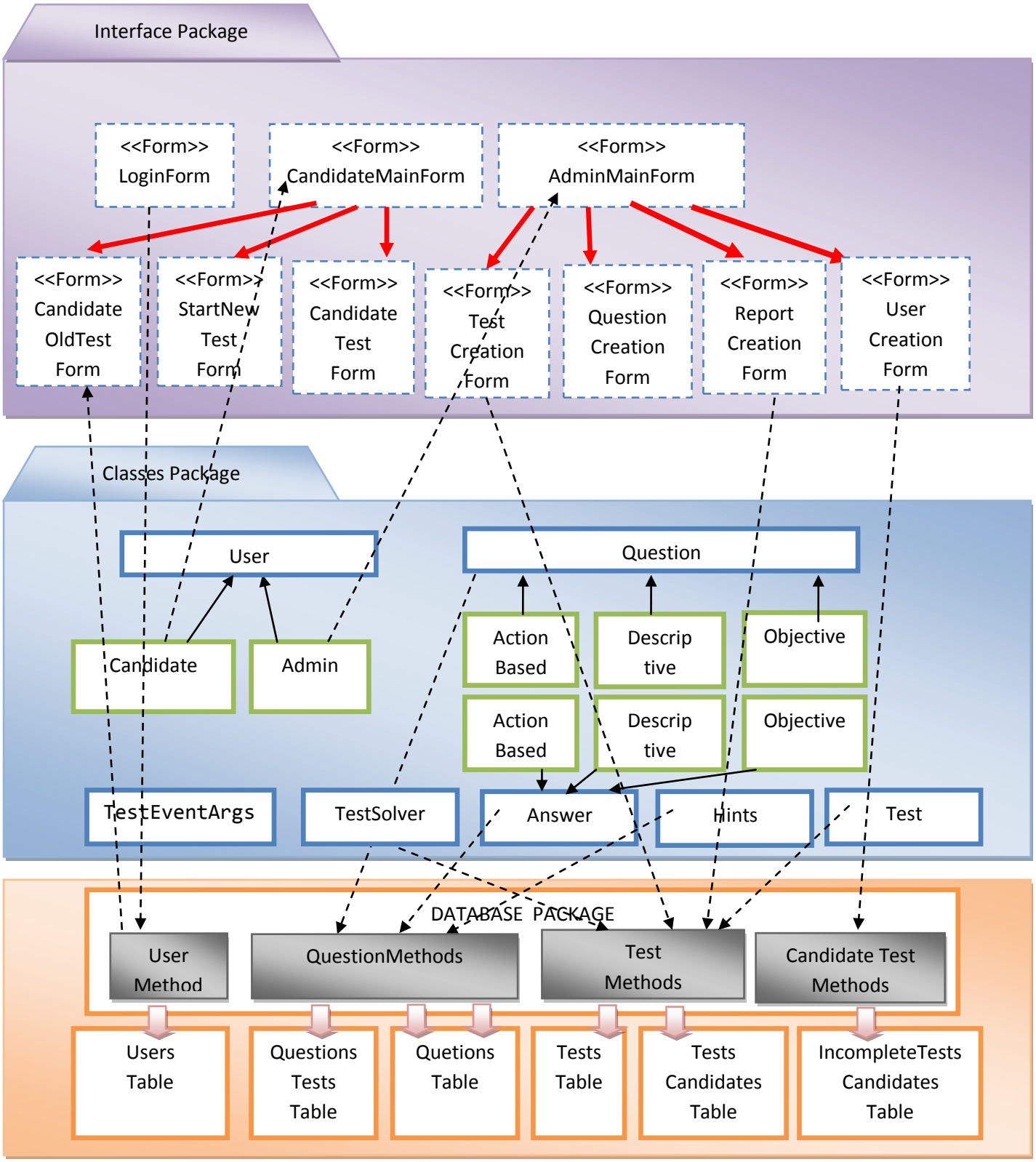


Figure 2.1 Package Diagram of the CTVIAT

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”, “CandidateMainForm” 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. These classes are “candidateTestForm”, “candidateOldTestForm”, “startNewTestForm”, “testCreationForm” and “QuestionCreationForm”, “ReportCreationForm”, “UserCreateForm”. All the classes in the interface package has responsibility to enable users to use program interface efficiently. These classes takes inputs from users and according to these inputs users directed to related parts of program.

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 “CandidateMainForm” 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 and “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, we also implement a simple class called “TestEventArgs” which is just return a string and used in “TestSolver” class.

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 loginning to system, creating a test, creating a question, taking a test, keeping user info, etc handled more easily.

2.3 Class Diagrams

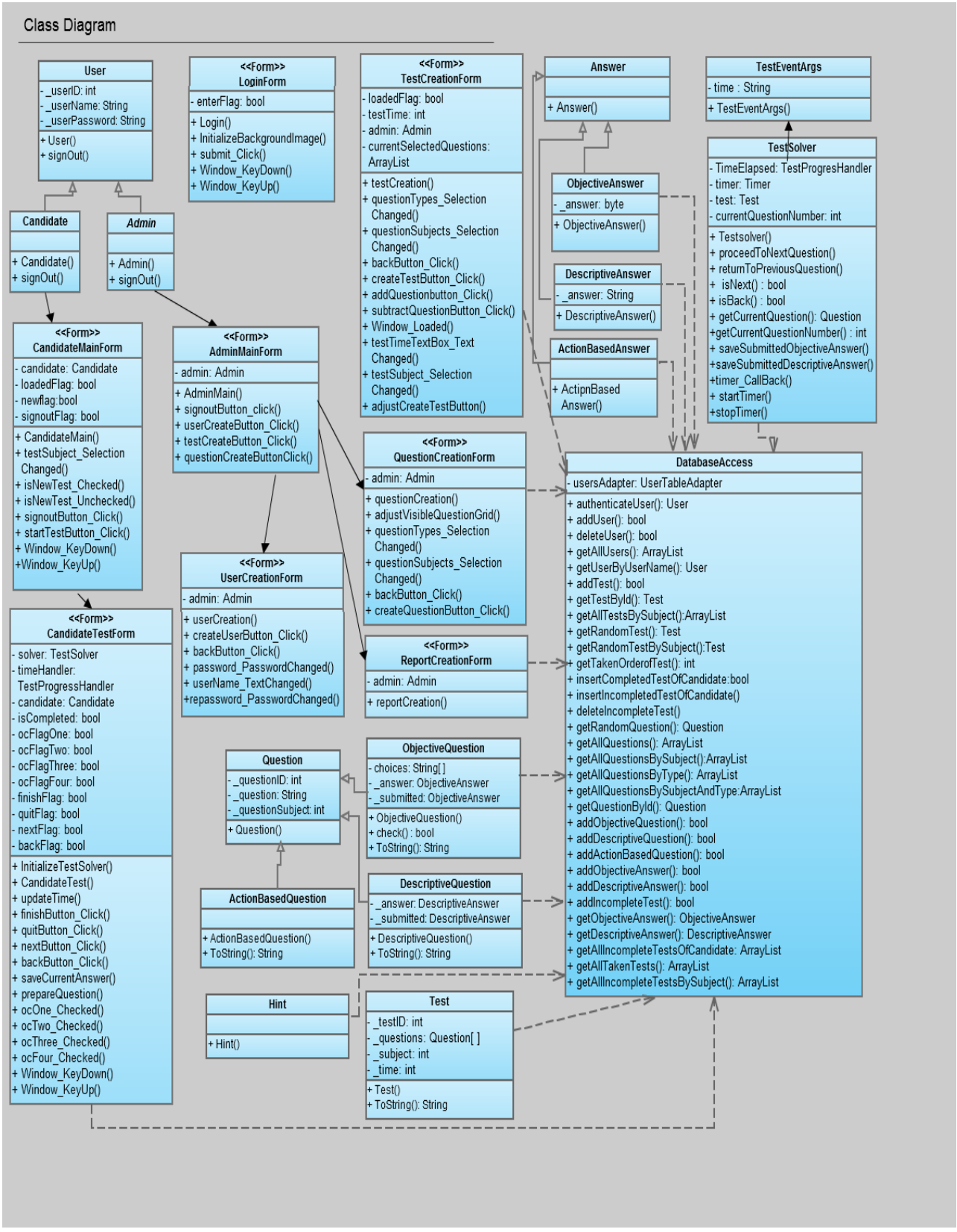


Figure 2.2 Class Diagram of the CTVIAT

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.

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 one member and 5 methods:

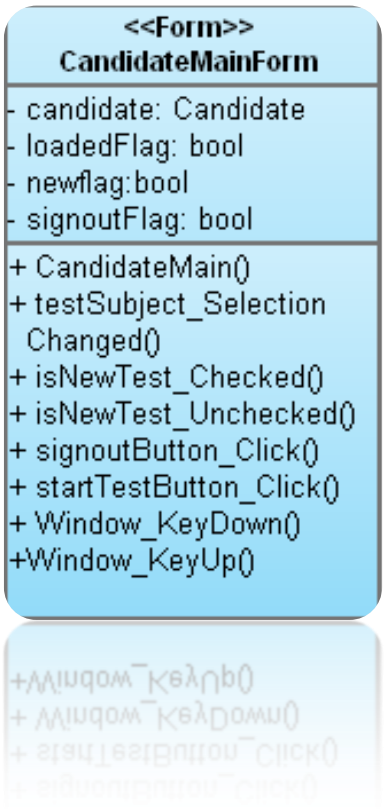
Members:

- enterFlag: whether user has login or not

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_KeyDown(): when user pressed a key this method capture the key
- Window_KeyUp(): when user release a key this method capture the key

2.4.2 CandidateMainForm



The candidateMainForm class is also a part of Interface Package and handles Visually Impaired users' operation. This class has four members and eight methods. These class will be used to enable visually impaired users to select a test and take that test:

Members:

- candidate: Candidate object
- loadedFlag: when form loaded this flag sets to true
- newFlag: this flag sets to true if user select new test
- signoutFlag: if user press S-key, this flag sets to true such that user can sign out

Methods:

- CandidateMain(): this method is class constructor
- testSubject_SelectionChanged() : when user select a new test subject in drop down list this method called
- isNewTest_Checked():
- isNewTest_Unchecked():
- signoutButton_Click(): when user pressed sign Out button, user session closed.
- startTestButton_Click(): when user pressed start test button this method start a test for visually impaired.
- Window_KeyDown():when user pressed a key this method capture the key.
- Window_KeyUp():when user released a key this method capture the key

2.4.3 AdminMainForm

The AdminMainForm class is a part of Interface Package and handles admins' operations. This class has one member and 5 methods in this design of ODD. These class will be used to enable admins to create questions and tests for visually impaired users.

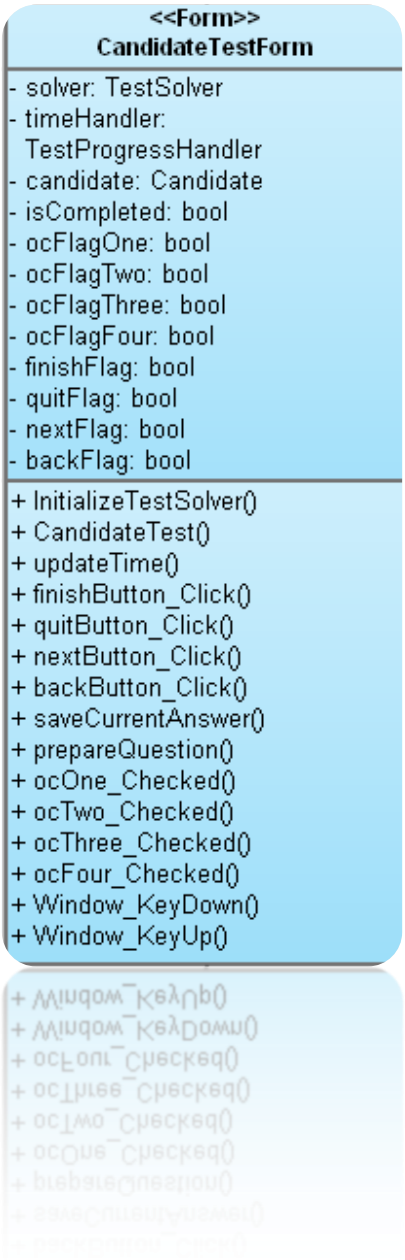
Members:

- 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 Creation form

2.4.4 CandidateTestForm



This class is inherited from candidateMainForm class and a part of Interface Package. We designed this class to enable visually impaired users to select a test for examination. In this class there are 12 members and 15 methods are implemented :

Members:

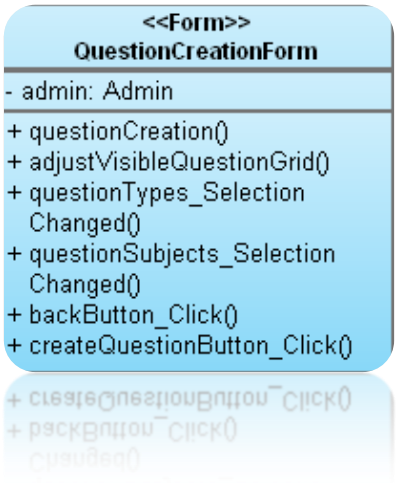
- solver: TestSolver object
- timeHandler: TestProgressHandler object
- candidate: Candidate object
- isCompleted: specify whether test is completed or not
- ocFlagOne: user selected first answer
- ocFlagTwo: user selected second answer
- ocFlagThree: user selected third answer
- ocFlagFour: user selected fourth answer
- finishFlag: specify whether user finished the test or not

- quitFlag: specify user pressed quit butoon or not
- nextFlag: specify whether user chose to go next question or not, namely next question button pressed or not
- backFlag: specify whether user chose to go previous question or not, namely previous question button pressed or not

Methods:

- initalizeTestSolver(): initalize a test for candidate
- CandidateTest(): is the class constructor
- updateTime(): controls remaining time for the test
- finishButton_Click(): when finish button pressed, related events handled here
- quitButton_Click():when quit button pressed, related events handled here
- nextButton_Click():when next question button pressed, related events handled here
- backButton_Click():when previous question button pressed, related events handled here
- saveCurrentAnswer(): stores the user answers during the examination
- prepareQuestion():
- ocOne_Checked():for objective type questions, if user choose first option, answer info stored
- ocTwo_Checked():for objective type questions, if user choose second option, answer info stored
- ocThree_Checked():for objective type questions, if user choose third option, answer info stored
- ocFour_Checked():for objective type questions, if user choose fourth option, answer info stored
- Window_KeyDown():when user pressed a key this method capture the key.
- Window_KeyUp():when user released a key this method capture the key

2.4.5 QuestionCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has six methods and one member. it enables admins to create questions for users. This class needs a database connecton to complete its tasks.

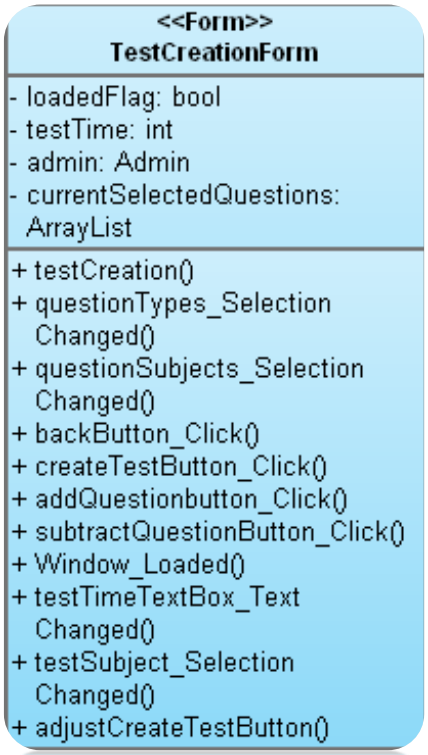
Members:

- admin: Admin object

Methods:

- questionCreation(): is the class constructor.
- adjustVisibleQuestionGrid():
- questionTypes_SelectionChanged(): if admin chose a question type from drop down list this methd called.
- questionSubjects_SelectionChanged():if admin chose a question subjectfrom drop down list this methd called.
- 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.

2.4.6 TestCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has 4 members and 11 methods and enable admins to create tests for the visually impaired users. All operations run under a successful database connection.

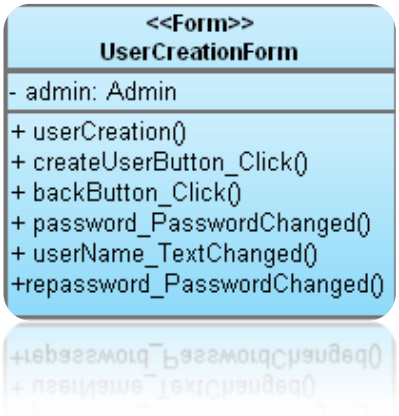
Members:

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

Methods:

- testCreation(): is the class constructor.
- 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
- testTimeTextBox_TextChanged():
- testSubject_SelectionChanged(): when admin select a new test subject from drop down list this method called.
- adjustCreateTestButton():

2.4.7 UserCreationForm



This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has 1 member and 6 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
- 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”

2.4.8 ReportCreationForm

This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has 1 member and 1 method and enable admins to create candidate related reports:

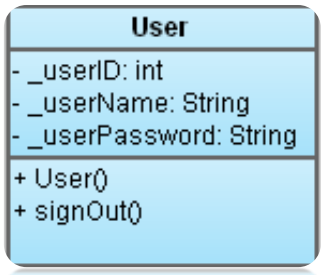
Members:

- admin: Admin object

Methods:

- reportCreation(): is the class constructor.

2.4.9 User



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

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.
- signOut() : enable users to end their session.

2.4.10 Candidate

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

Members:

Methods:

- signOut() : enable log-off for candidate
- Candidate():is the constructor of the class.

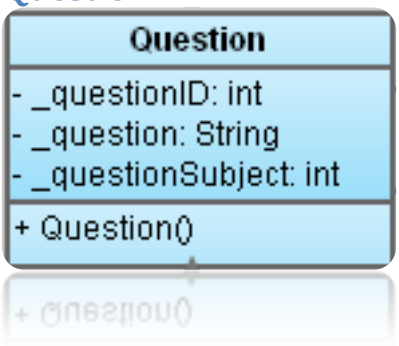
2.4.11 Admin

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

Methods:

- signOut() : enable log-off for admin
- Admin(): is the class constructor

2.4.12 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:

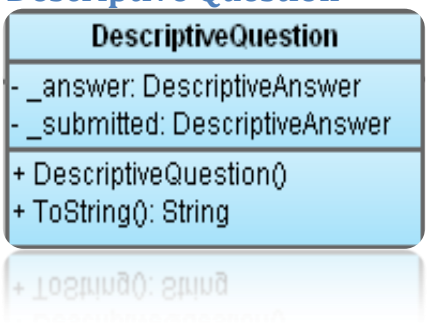
Members:

- _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

2.4.13 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:

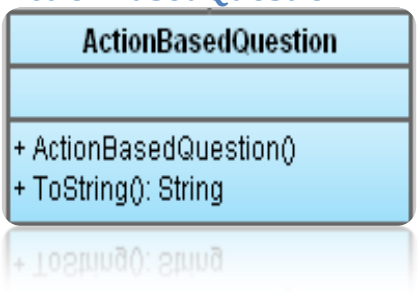
Members:

- _answer: DescriptiveAnswer object
- _submitted: DescriptiveAnswer object

Methods:

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

2.4.14 ActionBasedQuestion

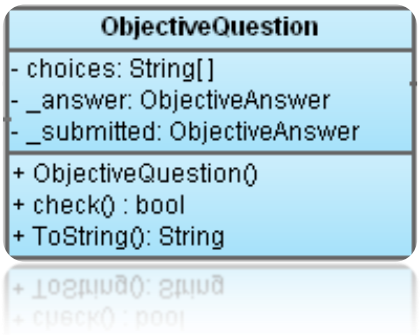


This class also inherited from Question Class and designed for action-based type questions. It has 2 methods:

Methods:

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

2.4.15 ObjectiveQuestion



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

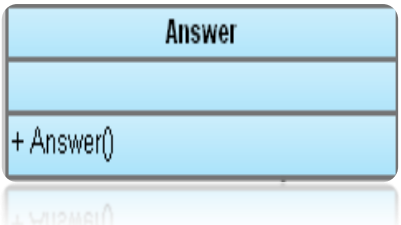
Members:

- choices: a string array that store the user choices
- _answer: ObjectiveAnswer object
- _submitted: ObjectiveAnswer object

Methods:

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

2.4.16 Answer

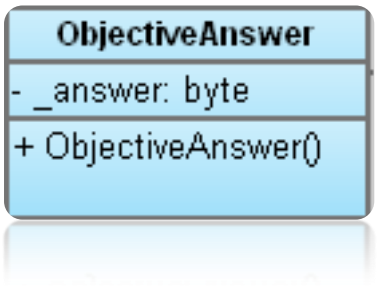


This class is in Classes Package and designed to encapsulete the data of answer for a spesific question. It has one method:

Methods:

- Answer(): is the class constructor

2.4.17 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:

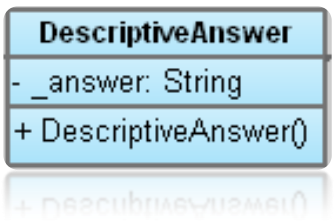
Members:

- _answer : keeps the answer info as byte

Methods:

- ObjectiveAnswer(): is the class constructor

2.4.18 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:

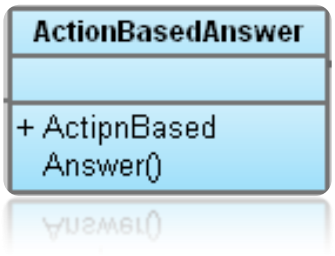
Members:

- _answer : keeps the answer info as String

Methods:

- DescriptiveAnswer(): is the class constructor

2.4.19 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 method:

Members:

Methods:

- ActionBasedAnswer(): is the class constructor

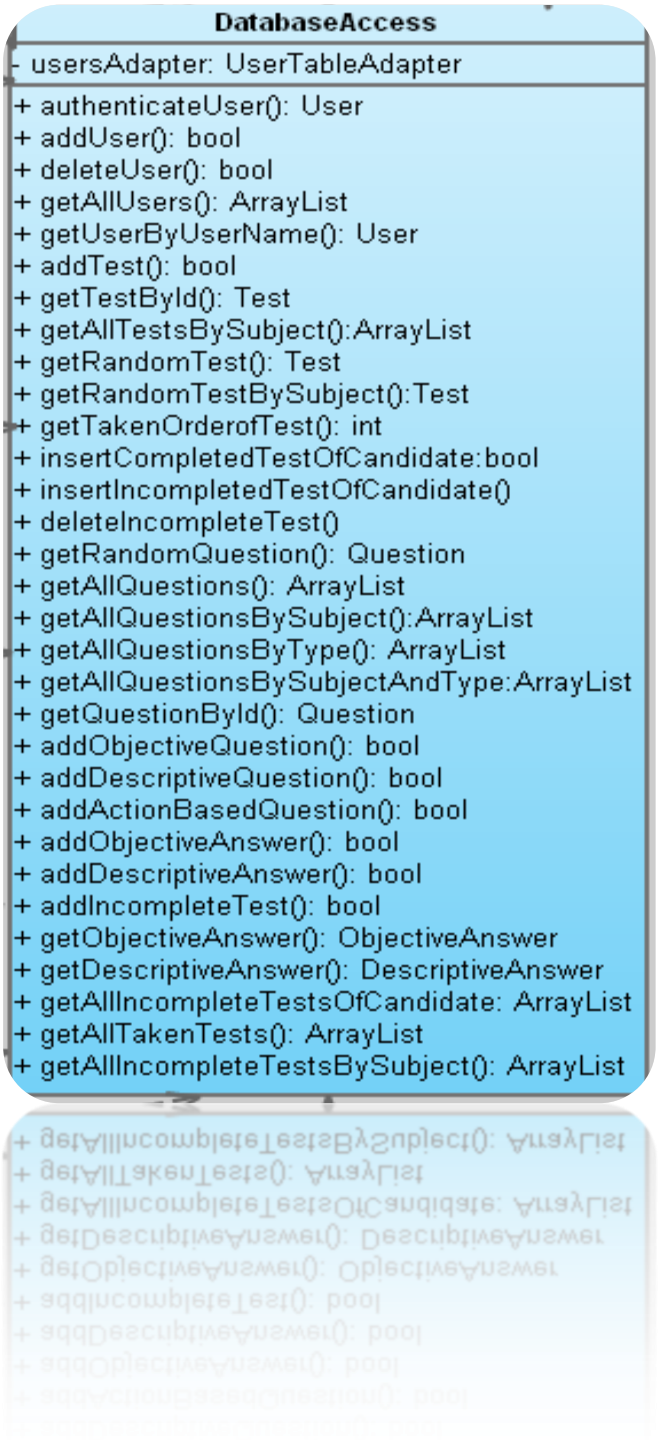
2.4.20 Hint

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

Methods:

- Hint(): is the class contructor

2.4.21 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 implement ed one member and 31 methods for this class:

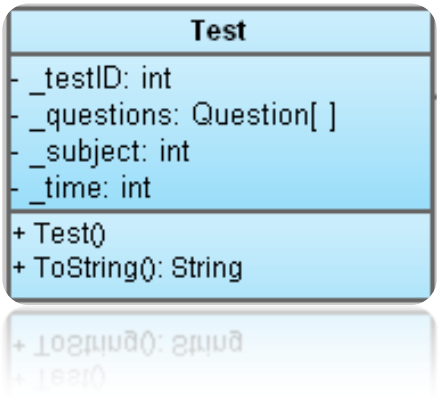
Members:

- userAdapter: userTableAdapter object that enable to search and bring data from database

Methods:

- authenticateUser(): authenticate users as Admin or Candidate.
- addUser(): enable adding a user to database.
- deleteUser(): deletes auser info from database.
- getAllUsers(): brings the information about all users.
- getUserByUserName(): brings user data for a specified name.
- addTest(): enable adding a new test to database.
- getTestById(): brings the information about a test specified by Test ID..
- getAllTestsBySubject():For a specified test subject brings all the test info.
- getRandomTest(): brings the data of a test randomly
- getRandomTestBySubject():For a specified test subject brings a test info randomly
- getTakenOrderofTest(): brings the taken order of a test
- insertCompletedTestOfCandidate(): insert a completed test info into database
- insertIncompletedTestOfCandidate():insert an incompleted test info into database
- deleteIncompleteTest(): deletes the info for an incompleted test
- getRandomQuestion(): brings the data of a random question
- 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
- addObjectiveAnswer():add data of an objective type answer
- addDescriptiveAnswer():add data of an objective type answer
- addIncompleteTest():add data of an incompleted test
- getObjectiveAnswer():brings data of an objective type answer
- getDescriptiveAnswer():brings data of an objective type answer
- getAllIncompleteTestsOfCandidate: for a candidate brings data of all incompleted tests.
- getAllTakenTests():brings data of all taken tests.
- getAllIncompleteTestsBySubject():brings data of all incompleted tests for a specified subject.

2.4.22 Test



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

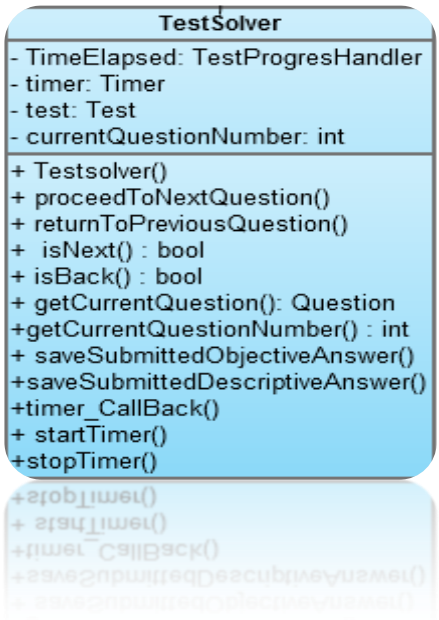
Members:

- _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
- _time: keeps the time information of the test

Methods:

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

2.4.23 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
- timer_CallBack(): updates the time of test
- startTimer(): starts Timer
- stopTimer(): stops Timer

2.4.24 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.