Object Design Document

Project Enlightenment

Computer Training For Visually Impaired Automation Tool (CTVIAT)

Prepared By: Project Enlightenment

Date: December 12, 2010

Version No: v2.0

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

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(answers ,question and hints will be stored in MP3 format in database) small. This will clearly make program more efficient. 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 project's main architecture. 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 project 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 subclasses. In Interface package, there are classes for login screen, admin screen and candidate screen and all responsible to handle interface operations. In Database package, there is a database access object class and 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

2. PACKAGES

2.1 Package Diagram

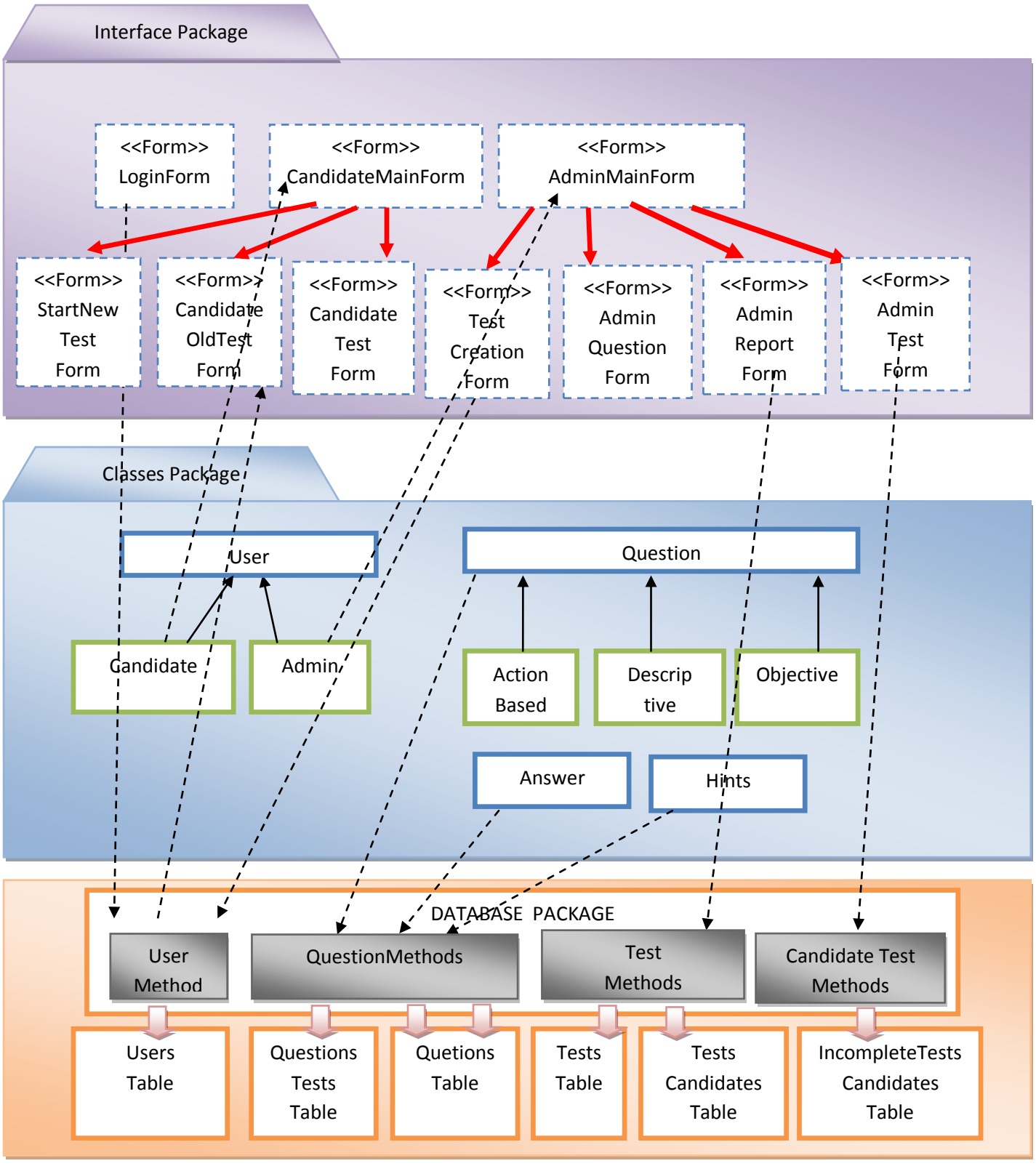


Figure 2.1 Package Diagram of the CTVIAT

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 in this version of the ODD, 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 "adminQuestionForm", "adminReportForm", "adminTestCreateForm". 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 four main classes named "User", "Question", "Answer" and "Hint". 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.

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. This class designed to handle database operations more easily. Since all other classes reach the database through this class, we implemented following methods for this package:

- User Methods
  - authenticateUser(String username, String password)
  - addUser(String userName, String password, byte type)
  - deleteUser(String userName, String password)

    - GetAllUsers()
- Test Methods
    - addTest(Test test)
    - getTestById(int testId)
    - getRandomTest()
    - getRandomTestBySubject(int subject)
- Question Methods
    - getRandomQuestion()
    - getQuestionById(int questionId)
    - getRandomObjectiveQuestionById(int questionId)
    - getRandomActionBasedQuestionById(int questionId)
    - getRandomDescriptiveQuestionById(int questionId)
    - getRandomObjectiveQuestion()
    - getRandomActionBasedQuestion()
    - getRandomDescriptiveQuestion()
    - getRandomQuestion(int subject)
    - addObjectiveQuestion(ObjectiveQuestion question)
    - addDescriptiveQuestion(DescriptiveQuestion question)
    - addActionBasedQuestion(ActionBasedQuestion question)
- Candidate-Test Medhods
    - addObjectiveAnswer(int candidateID, int testID, int questionID, int takeOrder, byte answer)
    - addDescriptiveAnswer(int candidateID, int testID, int questionID, int takeOrder, String answer)
    - addIncompleteTest(int candidateID, int testID, int questionID, int takeOrder)
    - getObjectiveAnswer(int candidateID, int testID, int questionID, int takeOrder)
    - getDescriptiveAnswer(int candidateID, int testID, int questionID, int takeOrder)
    - getAllIncompleteTestsOfCandidate(int candidateID)
    - getAllTakenTests(int candidateID)

2.3 Class Diagrams

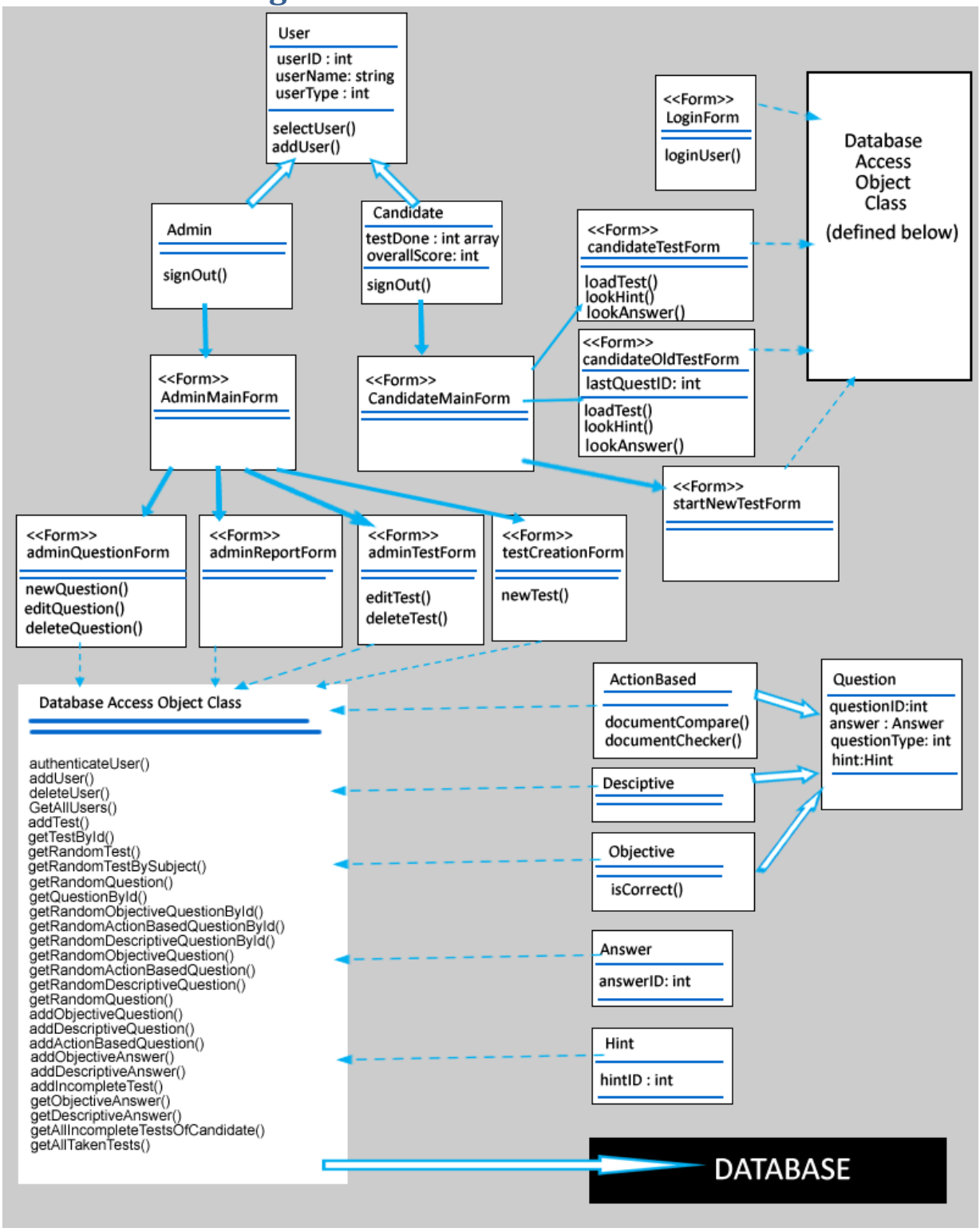


Figure 2.2 Class Diagram of the CTVIAT

The class diagram of the CTVIAT represents the classes, their attributes and methods, and the interactions between classes in the second version of the ODD. As the implementation

of the code goes further, to handle some operations, we need some new classes. Thus, in this verion of the ODD, we changed our class diagram accordingly.

2.4 Class Definitions

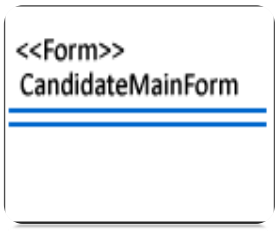
2.4.1 LoginForm

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 just one method loginUser():

Methods:

- loginUser() : to handle login operation of the users and redirect users to relevant interfaces.

2.4.2 CandidateMainForm

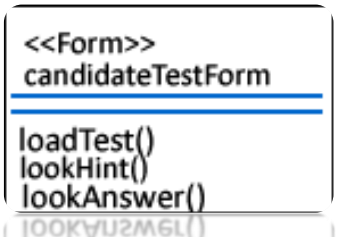


The candidateMainForm class is also a part of Interface Package and handles Visually Impaired users' operation. This class has no method and members in this design of ODD. These class will be used to enable visually impaired users to select a test and take that test.

2.4.3 AdminMainForm

The AdminMainForm class is a part of Interface Package and handles admins' operations. This class has no method and members in this design of ODD. These class will be used to enable admins to create questions and tests for visually impaired users.

2.4.4 CandidateTestForm

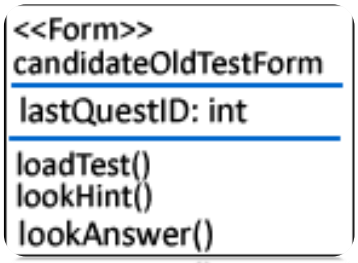


This class is inherited from candidateMainForm class and a part of Interface Package. We designed this class to enable visually impaired users to select a test for examination. In this class there are threemethod: selectTest() and loadTest().

Methods:

- loadTest() : this function enable selecting and loading an old test.
- lookHint() : enables visually impaired user to look at the hints about the question
- lookAnswer(): load an mp3 sound file to give answer to user

2.4.5 CandidateOldTestForm



This class is also inherited from candidateMainForm class and a part of Interface Package. This class is designed for visually impared users to re-take an old test. this class includes one member and three methods:

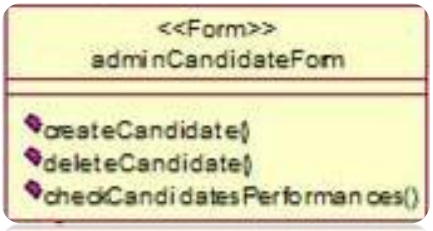
Members:

- lastQuestID : keeps the last question that visually impaired solve. This member will be useful when user stops one test and later wants to re-take the same test .

Methods:

- loadTest() : this function enable selecting and loading an old test.
- lookHint() : enables visually impaired user to look at the hints about the question
- lookAnswer(): load an mp3 sound file to give answer to user

2.4.6 AdminCandidateForm

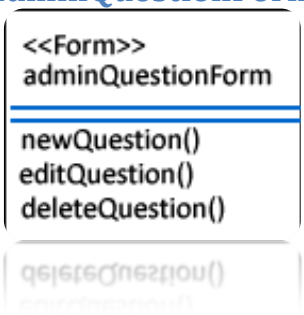


This class is inherited from AdminMainForm class and is a part of Interface Package. The class has three methots and enable admins to control candidates. To handle operations a successfull database connect is required.

Methods:

- createCandidate() : register a new candidate to the system
- deleteCandidate() : delete Candidate info from the system
- checkCandidatesPerformance(): enable admins to control performance of candidates

2.4.7 adminQuestionForm



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

Methods:

- newQuestion() : add a new question to database
- editQuestion() : updates a question
- deleteQuestion() : delete the question from the database

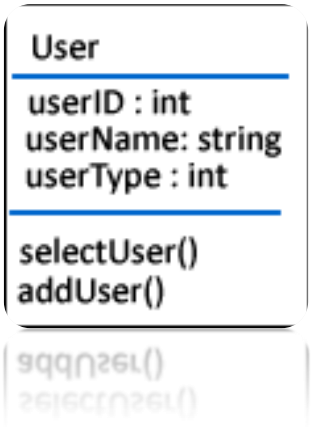
2.4.8 TestCreationForm

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

Methods:

- newTest() : create a new test

2.4.9 User



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

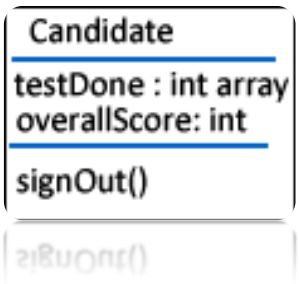
Members:

- 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:

- selectUser() : set userType to admin or candidate
- addUser() : enable creating admin or candidate type users

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 members and one method.

Members:

- testDone : an int array type variable to store info about done tests.
- overallScore: keeps the users score info for completed tests.

Methods:

- signOut() : enable log-off for candidate

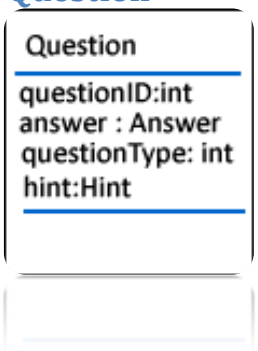
2.4.11 Admin

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

Methods:

- signOut() : enable log-off for admin

2.4.12 Question

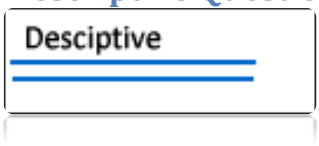


This class is also in Classes Package and designed to encapsulete the data of questions read from database. It has four members:

Members:

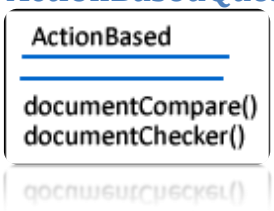
- quetionID : keeps primary key of question table
- answer: a Answer Class object to store answer of the qustion
- questionType: specify whether the question is objective,desciptive or action-based.
- Hint: a Hint Class object to store hints related with the question

2.4.13 Descriptive Question



This class is inherited from Question Class and designed to encapsulate the data about descriptive question type. In this design document of ODD it has no member and method.

2.4.14 ActionBasedQuestion

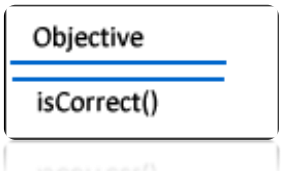


This class also inherited from Question Class and designed for action-based type questions. It has 2 methods:

Methods:

- documentCompare (): compare the task done by the visually impaired user and actual document
- documentChecker() : check whether user handled assigned task

2.4.15 ObjectiveQuestion

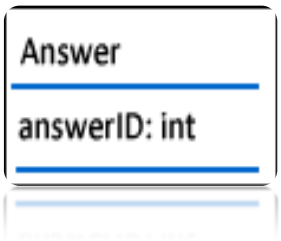


This class also inherited from Question Class and designed for objective type questions. It has one methods:

Methods:

- isCorrect(): check whether the question answered correctly or not

2.4.16 Answer

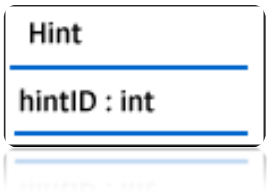


This class is in Classes Package and designed to encapsulete the data of answer for a spesific question. It has one member:

Member:

- answerID: keeps primary key of answer table

2.4.17 Hint



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

Member:

- answerID: keeps primary key of hint table

2.4.18 adminReportForm

This class is also inherited from AdminMainForm class and is a part of Interface Package. The class will be used to present a detailed report about candidates' testing performance. In his design document of ODD, this class has no method or members.

2.4.19 adminTestForm

This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has two methots and enable admins to delete or update tests for the visually impaired users. All operations run under a successful database connection.

Methods:

- editTest() : updates a test
- deleteTest() : delete a test from the system

2.4.20 Database Access Object Class

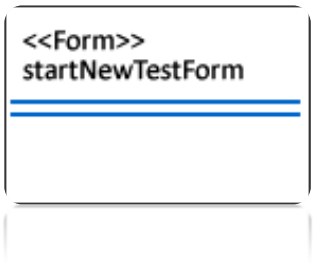
This class is a part of Database Package and enable all other classes to connect database efficiently. To handle database connection operation we implement ed 27 methods for this class:

Methods:

- User Methods
  - authenticateUser(String username, String password): check whether used registered to system
  - addUser(String userName, String password, byte type) : creates admin or candidate type user
  - deleteUser(String userName, String password) : delete a users info from the system
  - GetAllUsers() : find all users in the system
- Test Methods
  - addTest(Test test) : creates a new test
  - getTestById(int testId) : bring the relevant test by test id
  - getRandomTest() : bing a random test
  - getRandomTestBySubject(int subject) : for a selected subject test info bring from database randomly
- Question Methods
  - getRandomQuestion() : pick a random question from database
  - getQuestionById(int questionId) : bring a question by question id
  - getRandomObjectiveQuestionById(int questionId): an objective type question brought randomly
  - getRandomActionBasedQuestionById(int questionId)): an action-basec type question brought randomly
  -
  - getRandomDescriptiveQuestionById(int questionId)): an descriptive type question brought randomly
  - getRandomObjectiveQuestion()
  - getRandomActionBasedQuestion()
  - getRandomDescriptiveQuestion()
  - getRandomQuestion(int subject)
  - addObjectiveQuestion(ObjectiveQuestion question) : creates an objective type question
  - addDescriptiveQuestion(DescriptiveQuestion question) : creates an descriptive
  - addActionBasedQuestion(ActionBasedQuestion question) : creates an action-based type question
- Candidate-Test Medhods
  - addObjectiveAnswer(int candidateID, int testID, int questionID, int takeOrder, byte answer) : add objective type answer to database
  - addDescriptiveAnswer(int candidateID, int testID, int questionID, int takeOrder, String answer) : add descriptive type answer to database
  - addIncompleteTest(int candidateID, int testID, int questionID, int takeOrder) : store data about an incomplete test in database
  - getObjectiveAnswer(int candidateID, int testID, int questionID, int takeOrder) : bring the objective type answer
  - getDescriptiveAnswer(int candidateID, int testID, int questionID, int takeOrder) : bring the descriptive type answer

- getAllIncompleteTestsOfCandidate(int candidateID) : bring the info about all uncompleted tests
- getAllTakenTests(int candidateID) : brings the info about all the tests a candidate taken.

2.4.21 StartNewTestForm



This class will be used to enable candidate to take a new Test and also inherited from candidateMainForm class. In this design of ODD it has no members and methods.