Object Design Document

Project Enlightenment

Computer Training For Visually Impaired Automation Tool (CTVIAT)

Prepared By: Project Enlightenment

Date: October 14, 2010

Version No: v1.0

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

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

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 users many functionalities, there should be short respond time when carrying out these functionalities to satisfy user expectations. Because, during the uusing of the application, visually impaired people will be alone and must not wait too much when taking an exam. In another word, the application should has a high performance. However, to satisfy this, we should decrease degree of complexity. One solution to decrease the complexity is to keep size of sound files(answers ,question and hints will be stored in MP3 format in database) small. This will clearly make moregram 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 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 data more secure. This of course will be handled through a more reliable database management and independent design of subsystems. However, this will increase code size and memory usage will increase.

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 object class and responsible to manage database connection and enable other classes to reach database through this class.

//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. CustomerTableName attribute of the class AutoMailSender).

The names of class methods are defined as verbs, usually consisting of more than one word, where the initial letters of all the words except the first are capitalized (e.g. sendMail).

1.3 Definitions, Acronyms and Abbreviations

CTVI	**:**	Computer Training for Visually Impaired
CTVI AT	**:**	Computer Training for Visually Impaired Automation Tool

1.4 References

2. PACKAGES

2.1 Package Diagram

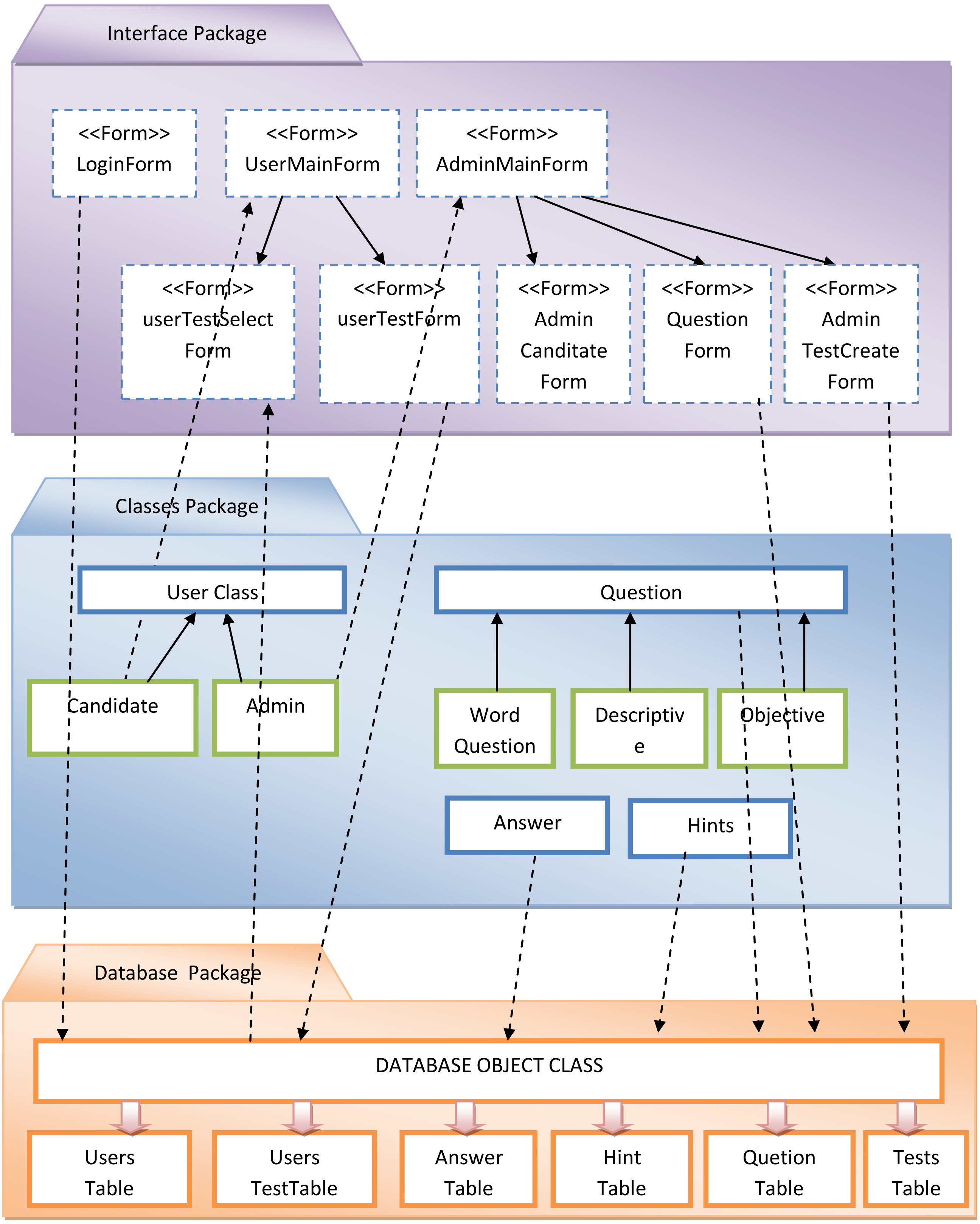


Figure 2.1 Package Diagram of the CTVIAT

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”, “UserMainForm” 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 the database. Also, Interface package includes some Candidate and Admin related classes to handle user specific operations. These classes are “userTestSelectForm”, “userTestForm”, “adminCandidateForm”, “QuestionForm” and “adminTestCreateForm”. All the classes in interface package has responsibility to enable user to use program interface efficiently. These classes takes inputs from users and according to these inputs users directed to related parts of program. The methods related to these classes will be explained in section-3.

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 “UserMainForm” operations and the second one is designed to handle “AdminMainForm” operations. “Question” class enable admins to create questions in different types. to handle this, we design 3 sub classes inherited from this class: “Word Question” 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 otherc classes reach the database through this class, it has a core importance and will be given special attention.

2.3 Class Diagrams

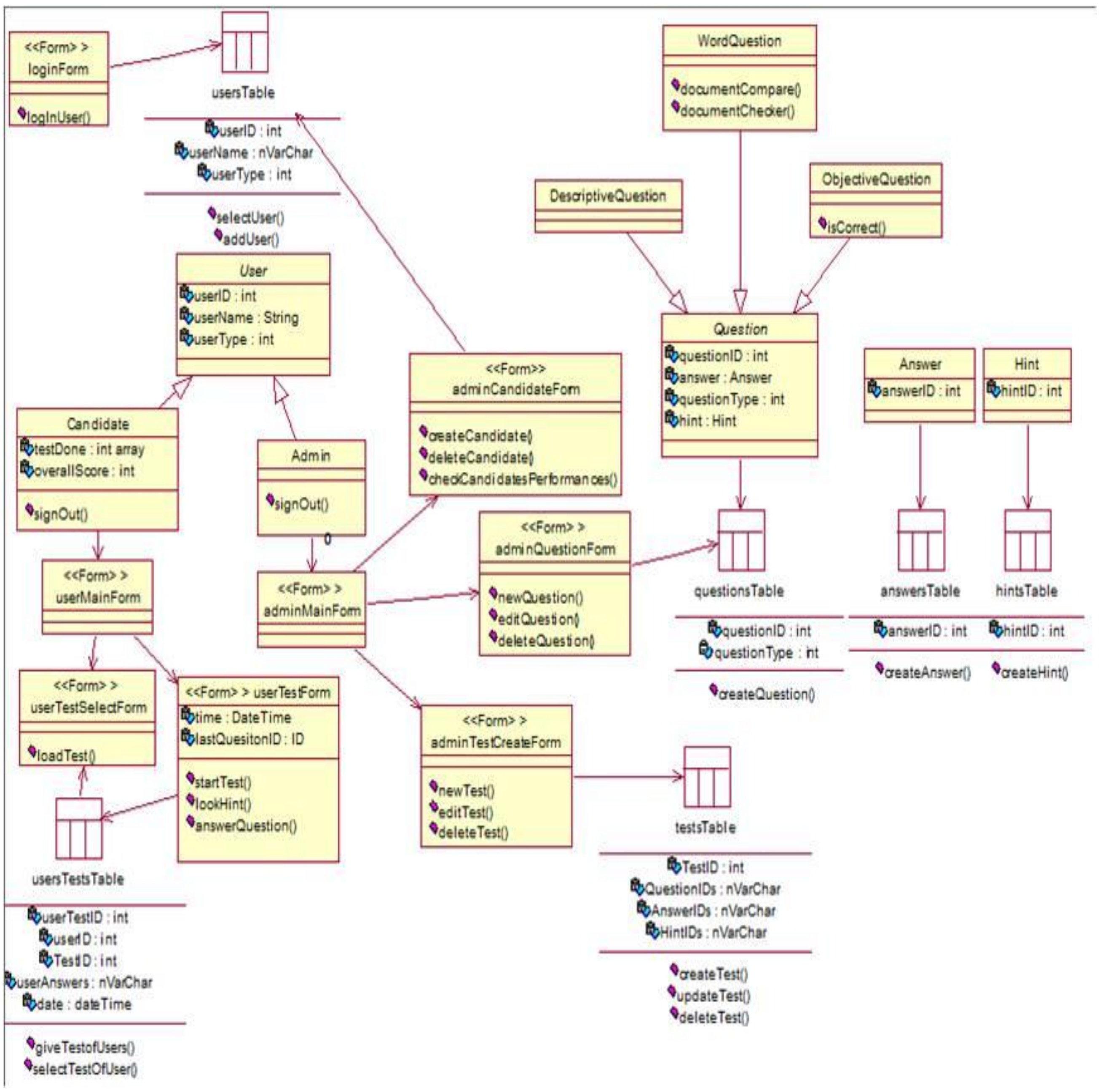
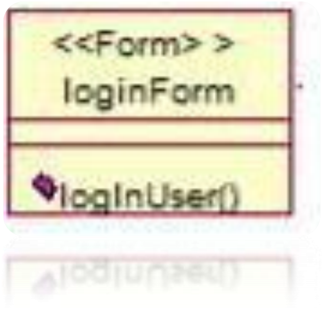


Figure 2.2 Class Diagram of the CTVIAT

The class diagram of the CTVIAT represents the classes, their attributes and methods, and the interactions between classes in the first version of the ODD. In The new versions, there will be added more atributes and methods according to needs to omplete some functionalities of the CTVIAT application.

2.4 Class Definitions

2.4.1 LoginForm

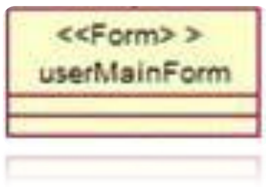


The LoginForm class is a part of Interface package and handles login operation of the users. This class create a database connection with user info and check user. İf authorization succeeded, it hands over its job to User class by calling it. LoginForm class has just one method loginUser():

Methods:

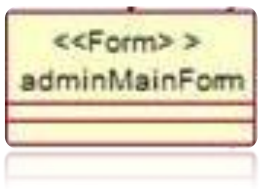
- loginUser() handle login operation of the users and redirect users to relevant interfaces.

2.4.2 UserMainForm



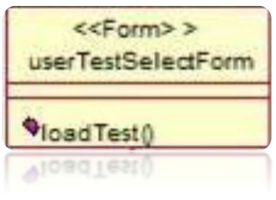
The UserMainForm class is also a part of Interface Package and handles Visually Impaired users operation. This class has no method and members in this design of ODD. These class will be used to enable visually impaired users to select a test and take that test.

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 UserTestSelectForm

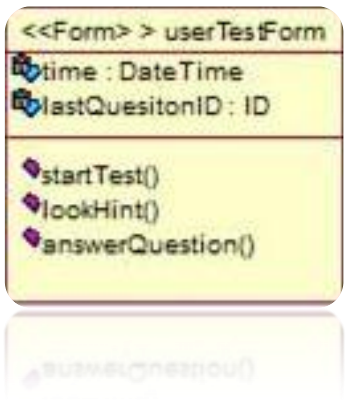


This class is inherited from UserMainForm class and a part of Interface Package. We designed this class to enable visually impaired users to select a test for examination. In this class for now there is a single method: loadTest().

Methods:

- loadTest() enable visually impaired user to select a test and load that test.

2.4.5 UserTestForm



This class is also inherited from UserMainForm class and a part of Interface Package. This class is designed for visually impared users to take a test . this class includes two members and three methods:

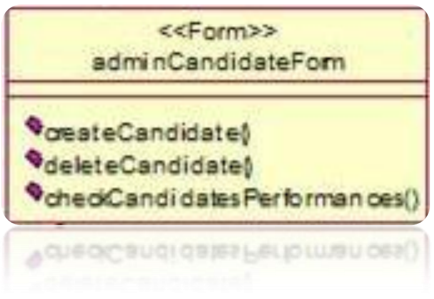
Members:

- time : keeps current date time
- lastQuestionID : 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 where he/she left.

Methods:

- startTest() : this function starts the selected test
- lookHint() : enables visually impaired user to look at the hints about the question
- answerQuestion(): load an mp3 sound file to give answer to user

2.4.6 AdminCandidateForm

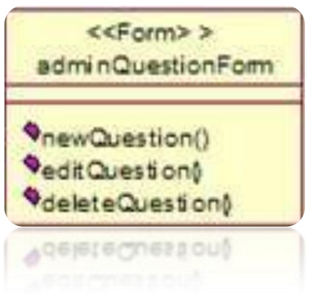


This class is inherited from AdminMainForm class and is a part of Interface Package. The class has three methots and enable admins to control candidates. To handle operations a successfull database connect is required.

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

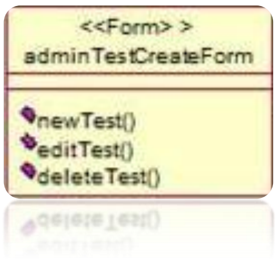


This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has three methots and enable admins to create questions for users. This class needs a database connecton to complete its tasks.

Methods:

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

2.4.8 AdminTestCreateForm

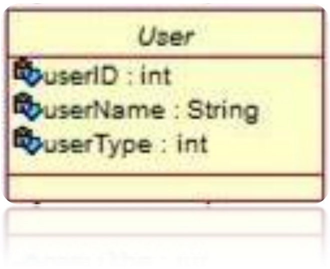


This class is also inherited from AdminMainForm class and is a part of Interface Package. The class has three methots and enable admins to create/delete or update tests for the visually impaired users. All operations run under a successful database connection.

Methods:

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

2.4.9 User

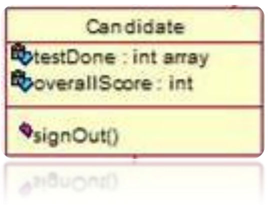


The User class in Classes Package and handles users related oprations . this class has three members:

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

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:

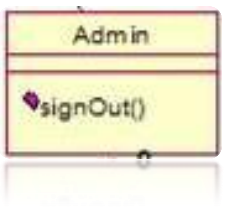
- testDone : an int array type variable to store info about done tests.

- overallScore: keeps the users score info for completed tests.

Methods:

- signOut() : enable log-off for candidate

2.4.11 Admin

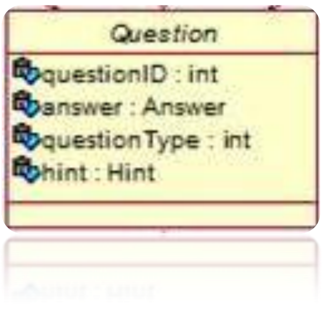


This class is also inherited from User class and will be used to handle admin related operations together with adminMainForm class. It has one method.

Methods:

- signOut() : enable log-off for admin

2.4.12 Question

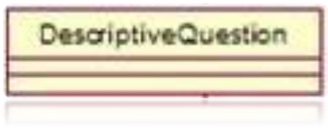


This class is also in Classes Package and designed to encapsulete the data of questions read from database. It has four members:

Members:

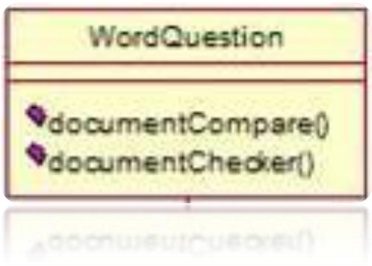
- quetionID : keeps primary key of question table
- answer: a Answer Class object to store answer of the qustion
- questionType: specify whether the question is objective,desciptive or action-based.
- Hint: a Hint Class object to store hints related with the question

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 WordQuestion

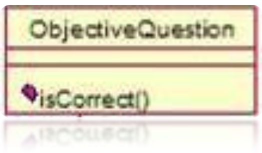


This class also inherited from Question Class and designed for action-based type questions. It has 2 methods:

Methods:

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

2.4.15 ObjectiveQuestion

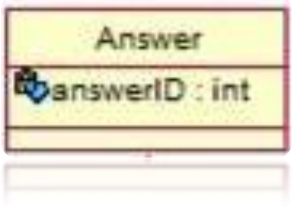


This class also inherited from Question Class and designed for objective type questions. It has one methods:

Methods:

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

2.4.16 Answer

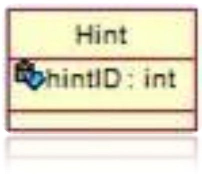


This class is in Classes Package and designed to encapsulete the data of answer for a spesific question. It has one member:

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