**Programmer's Manual**

**Survey System Anywhere**

**Lizard League Software**

**Table Of Contents**

Additional Resources............................................................................................... 3

Set up for coding in Eclipse...................................................................................... 3

File List..................................................................................................................... 4

Package List ............................................................................................................ 4

List of Classes, Listeners, and Interfaces.................................................................. 5

Class Heirachey........................................................................................................ 7

Class Methods.......................................................................................................... 8

Terminology ............................................................................................................ 10

SQLite Database....................................................................................................... 10

Example of how one might add functionality ......................................................... 11

## Additional Resources

Additional documents outside of the Programmer's Manual may help in your understanding of the software. The documents that should be reviewed are:

Survery Software DB relational diagram.pdf

Users Documentation.docx

surveysoftwareUML.jpg

## Set Up For Coding In Eclipse

The team that initially created Survey Software Anywhere used Eclipse. We have not tested how the code will behave in other Java editors.

Users can create a Pull Request on GitHub for the software code files. The project can be found at <https://github.com/UIS-CSC478-project/survey-software>.

Once you have the code, perform the following steps:

1) Create a new Java project in Eclipse.

2) Create a new package in the Eclipse project and call it surveysoftware.

3) Import the code files into the surveysoftware package.

4) Download the SQLite JDBC and place it in a lib folder under your project name in the same location where you see your bin and src folder. At the time of this manual, the jar could be downloaded from bitbucket: <https://bitbucket.org/xerial/sqlite-jdbc/downloads>.

5) List the jar file in your build path.

If you need assistance using Eclipse. the Eclipse documentation is posted on eclipse.org: <http://www.eclipse.org/documentation/>.

## File List

Here is a list of all files that currently make up the software.

Survey\_Main.java

Choice.java

SurveyGUI.java

SurveyGive.java

SurvRef.java

SurvRev.java

Db\_Interface.java

Survey\_Db.java

Survey\_Inteface.java

Survey\_Actions.java

**List Of Packages**

**surveysoftware** - Holds all classes listed under List of Classes.

**List Of Classes, Listeners, and Interfaces**

**Survey\_Main** - Calls the first window, Choice, for the start of the program.

**Choice** - First window that is shown after program starts. Holds the buttons Create Survey, Lookup Survey, and Quit.

**Choice.ButtonListener** - Calls SurveyGUI when the button Collect Survey is clicked. Calls SurvRef when Lookup Survey is clicked. Ends the program when the Quit button is clicked.

**SurveyGUI** - Shows window after click of the button Give Survey from the Choice Class. Holds the text areas for survey name, the survey question, the possible answers for answers a, b, c, or d, and the correct answer. Holds the buttons for next question and done.

**SurveyGUI.ButtonListener** - Listens to the Next Question and Done buttons in SurveyGUI. Collects data from the text areas in SurveyGUI and calls addNewQuestionWithAnswer from the Survey\_Actions class. If Next Question is clicked, it clears the question and answers, but not the survey name, and stays on the same window. If Done is clicked, the window is closed and the Choice window is opened.

**SurveyGive** - Shows window after click of the button Give Survey from the Choice class. Shows a list of all survey questions and their possible answers. Holds blank text fields that users can enter answers into. Holds 2 Done buttons, one at the top of the survey and one at the bottom.

**SurveyGive.ButtonListener** - Listens to the Done button in SurveyGive. Collects data from the text areas in SurveyGive, which are the survey responses a user gives. Checks the validity of the data entered into the text areas. Then calls addResults from the Survey\_Actions class. After that is closes the SurveyGive window and opens the SurveyRef window.

**SurvRef** - Shows window after click of the button Look Up Survey from the Choice Class. Holds the drop-down box for choosing a survey, and the buttons Review Survey and Give Survey.

**SurveyRef.ButtonListener** - Listens to all buttons in the SurvRef class. It calls the SurvGiv class when Give Survey is clicked, and calls the SurvRev class when Review Survey is clicked.

**SurvRev** - Shows window after click of the button Review Survey from the Choice class. Shows a list of all survey questions and their possible answers. Holds text fields that users can enter answers into. Holds 2 Done buttons, one at the top of the survey and one at the bottom.

**SurveyRev.ButtonListener** - Listens to the Done button in SurveyRev. It closes the SurveyRev window and opens the SurveyRef window again.

**Db\_Interface** - Interface that will hold the methods required by the class that will access the survey database directly: survey\_db.

**Survey\_Db** - Class that implements the db\_interface.db interface. This class will manipulate the data that goes in and out of the database.

**Survey\_Inteface** - Interface that will hold the methods required by the class that will access the survey database class: survey\_actions.

**Survey\_Actions** - Interface that will hold the methods required by the class that will access the survey database.

**Class Hierarchy**

surveysoftware.Db\_Interface

surveysoftware.Survey\_Db

surveysoftware.Survey\_Interface

surveysoftware.Survey\_Actions

surveysoftware.Survey\_Main

ActionListener

surveysoftware.SurveyGUI.ButtonListener

surveysoftware.SurvGive.ButtonListener

surveysoftware.SurvRef.ButtonListener

surveysoftware.SurvRev.ButtonListener

JFrame

surveysoftware.Choice

surveysoftware.SurveyGUI

surveysoftware.SurvGive

surveysoftware.SurvRef

surveysoftware.SurvRev

**Class Methods**

**Survey\_Main**

* static void main (String[] args) - Calls the first window, Choice, for the start of the program.

**Choice**

* Choice() - constructor
* void initChoice() - Creates the buttons Look Up Survey, Create Survey, and Quit.

**SurveyGUI**

* SurveyGUI () - constructor
* void initGUI () - Calls methods multiCh() and header().
* void header () - Creates editable text area for the survey name.
* void multiCh () - Creates editable text areas for the survey answers.

**SurveyGive**

* SurvGive (String s) - contructor. Needs the survey name for which survey it will present.
* void initGUI () - Shows the survey selected to the user. Creates editable text areas for users to enter answers into. Has two buttons called Done for exiting out of the window.

**SurvRef**

* SurvRef () - constructor.
* void initGUI () - Creates dropdown field that lists all the current survey names. Creates two buttons for choosing which action to act upon the survey name.

**SurvRev**

* SurvRev (String s) - constructor. Needs the survey name of the survey that will be presented to users.
* void initGUI () - Creates uneditable text areas that are used to display the survey questions and possible answers. Has two buttons called Done for exiting out of the review area.

**Survey\_Db** (Same methods are listed in the Db\_Interface)  
All methods in this class interact directly with the SQLite database.

* Survey\_Db () - constructor
* boolean check4Db () - Check for the presence of an existing survey database: surveydatabase.db.
* void createDb () - Create a new survey database with tables if the check4Db returns false.
* void addNewSurvey (String mysurveyname) - Add a new survey name to the survey table.
* void addNewQuestionWithAnswer (ArrayList questionanswer, int numanswers) - Add a new question plus its possible answers to the tables Survey\_Questions and Answers.
* void addResults (int quesID, String answer) - Input results, one letter result with no additional text, from survey\_actions into database.
* void addResults (int quesID, String answer, String other) - Input results, one letter result with additional text, from survey\_actions into database.
* void deleteDb () - Delete the current survey database, surveydatabase.db.
* ArrayList getSurveyNames () - Return a list of all the surveys that have been created so far.
* ArrayList getSurveyQuestionsAnswers (int surveyID) - Returns an arraylist of arraylists that holds all the questions and possible answers for a survey.
* int getSurveyId (String newsurvey) - Returns the survey id for the survey name sent to it.
* int getNumberOfQuestions (int surveyID) - Returns the number of questions that are in a survey.
* int getResultSetNumRows (ResultSet rs) - Returns the number of row that are in a result set.
* int getAnswerId (int quesID, String answer) - Returns the ID for the answer sent to it.

**Survey\_Actions** (Same methods are listed in the Survey\_Interface)  
This class does not directly access the database. When it needs to send data to and from the database, it calls the Survey\_Db class.

* Survey\_Actions () - constructor. Calls the method in Survey\_Db for checking if there is an existing database. If the database does not exist, it calls the Survey\_Db.CreateDb() function to create the database.
* void addNewSurvey (String surveyName) - Receives the survey name from SurvGUI and sends it to Survey\_Db where it is added to the database.
* void addNewQuestionWithAnswer (ArrayList< String > questionAnswer, int numAnswers) - Receives the arraylist with all the questions and answers from SurveyGUI and sends them to SurveyDb.addNewQuestionWithAnswer().
* void addResults (int quesId, String answer) - Receives the answer letter a user provides when taking the survey from the SurvGive class and sends it to Survey\_Db.addResults().
* void addResults (int quesId, String answer, String other) - Receives the answer letter and additional text a user provides when taking the survey from the SurvGive class and sends it to Survey\_Db.addResults().
* int getSurveyId (String surveyName) - Returns the survey ID it receives from SurveyDb.getSurveyID().
* int getNumberOfQuestions (int surveyId) - Returns the number of questions in a survey that is provided by Survey\_DB.getSurveyId().
* ArrayList getSurveyQuestionsAnswers (String surveyName) - Returns the arraylist of arraylists that holds all the questions and possible answers that come from Survey\_DB.getSurveyQuestionsAnswers().
* String[] getSurveyNames () - Receives an arraylist of survey names from Survey\_Db.getSurveyNames(). It converts the arraylist to a string array and returns the string array.
* boolean survey\_Exists (String surveyName) - Receives a list of all the survey names from Survey\_DB.getSurveyNames() and compares those names to the survey name sent to it by SurvGive. It returns true if any names that currently exist match the one entered in SurvGive.

**Terminology**

Result Set - The information that is returned from a SQL select statement.

SQLite - The database used for this software package.

**SQLite database**

Table: Survey

S\_ID Primary Key ID of the survey name

SURVEY\_NAME Text

TABLE: SURVEY\_QUESTION

Q\_ID Primary Key ID of the question

FK\_S\_ID Foreign Key ID of the survey name

QUESTION\_TEXT Text

QUESTION\_Answer Text Correct answer to the question.

TABLE: POSSIBLE\_ANSWERS

PA\_ID Primary Key ID of the possible answer

FK\_Q\_ID Foreign Key ID of the question

LETTER Text Can only be a, b, c, or d.

ANSWER\_TEXT Text

TABLE: ANSWERS

A\_ID Primary key ID of the answer given.

FK\_Q\_ID Foreign Key ID of the question

FK\_PA\_ID Foreign Key ID of the possible answer

OTHER\_RESP Text Additional text user can add to an answer if allowed in a survey

**Example of how one might add functionality**

A programmer decides to add a new window that will have some new functionality to it. Here are the steps one might use:

1) Create a new class for the new window, and add any items you need in that window.

2) If the new window needs a user to look up a survey name first, then a good place to create a button leading to the new window would be in the SurvRef class. If the functionality doesn't need to look up a survey name, then a good place to create a button leading to the window might be in the Choice class.

3) If the new windows needs data to and from the database, do not call the Survey\_Db function directly from the new window class. Instead call a function in Survey\_Actions. Manipulation of data can happen in Survey\_Actions, but direct database access is reserved for Survey\_Db. Then have Survey\_Actions call a function in Survey\_Db to get the data it needs from the database or to add data to the database.