|  |
| --- |
| DiSH group |
| DiSH Program Design Document |
| Tonya Burton, Jaleesa Elston, Lori Liles, Logan Moore, Andrew Peil, Tony Teem |
|  |
| **Jacksonville State University – MCIS Department** |
| **2/22/2009** |

|  |
| --- |
|  |

Contents

[Experiment Class 3](#_Toc223089755)

[Experiment User Interface Class 3](#_Toc223089756)

[Participant Information Class 3](#_Toc223089757)

[Method Parent Class 4](#_Toc223089758)

[Decreasing Adjustment Subclass 4](#_Toc223089759)

[Double Limit Subclass 4](#_Toc223089760)

[Multiple Choice Subclass 5](#_Toc223089761)

[Real Time Subclass 5](#_Toc223089762)

[Method User Interface Parent Class 5](#_Toc223089763)

[Decreasing Adjustment User Interface Class 6](#_Toc223089764)

[Double Limit User Interface Class 6](#_Toc223089765)

[Multiple Choice User Interface 6](#_Toc223089766)

[Real Time User Interface Class 6](#_Toc223089767)

[Scenario Class 6](#_Toc223089768)

[Scenario User Interface Class 6](#_Toc223089769)

[Option Class 6](#_Toc223089770)

[Option User Interface Class 7](#_Toc223089771)

[Reward Class 7](#_Toc223089772)

[Time Class 7](#_Toc223089773)

[Main Class 7](#_Toc223089774)

[Database Class 7](#_Toc223089775)

[Experiment Loader 8](#_Toc223089776)

[Database Table and Field Descriptions 9](#_Toc223089777)

[DiSH Class Diagram 11](#_Toc223089778)

[Interface Design 12](#_Toc223089779)

[Database Diagram 13](#_Toc223089780)

[Experiment Loader Diagram 14](#_Toc223089781)

### Experiment Class

Name: experiment

Description:  This class contains the main execution of the program.

Attributes:

* List<Methods> (see below)
* title -- The title of the JFrame (window) represented by a String.
* number -- This is the experiment number based on the name of the XML file represented by a String.
* conclusion -- When the test is finished this will display to give a message from the experiment creator to the participant represented by a String.
* participantInformation -- See ParticipantInformation Class information below.

Operations:

*Name*:  addPersonnel()  
 *Arguments*:  investigator  
 *Returns*:  No return value

*Precondition*:  None  
*Postcondition*: Adds the identification of the Experimenter to the experiment. This is automatically called when the test is selected from the load menu.  
*Exception:*  None

### Experiment User Interface Class

Name:  experimentUI

Description:  This class is the user interface module for the experiment window. It will control how the experiment is displayed to the user.

Attributes:  Only local private attributes exist for this method. It will not contain any functionality other than controlling the UI.

### Participant Information Class

Name:  ParticipantInformation

Description:  Whenever a participant enters their information it will be stored in this class which is a part of the Experiment main class.

Attributes:

* id -- The student ID of the participant represented by an integer.
* firstName -- The first name of the participant represented by a String.
* lastName -- The last name of the participant represented by a String.
* age -- The age of the participant represented by an Integer.
* race -- The race of the participant. The choices will consist of:
  + American Indian or Alaska Native
  + Asian
  + Black or African American
  + Hispanic or Latino
  + Native Hawaiian or Other Pacific Islander
  + White
* gender -- The gender of the participant. Choices will consist of Male or Female.

Operations: There are no special operations for this method.

### Method Parent Class

Name:  method

Description:  The parent class of the 4 different testing methods.  All 4 methods will extend this class and include the following.

Attributes:

* minReward -- The lowest reward the participant the can earn represented by a float.
* maxReward -- The highest reward the participant can earn represented by a float.
* intertrialInterval -- The amount of time to wait between closing one method and opening the next in the window represented by a float.
* totalTimeElapsed -- The amount of time that has elapsed between the beginning of the method and the end of that method represented by a float.
* methodType -- What subclass the method should invoke represented by an enumerated data type that can include DEC\_ADJ, DBL\_LIM, MULTI\_CHOICE, REAL\_TIME
* indifferencePoint -- The indifference point is simply the final value of x after the specified number of scenarios. This is an object of type Option.
* List<scenario> -- A list of possible scenarios to be used in the method.

Operations: There are no special operations for this method.

### Decreasing Adjustment Subclass

Name: decreasingAdjustment

Description: A subclass of the Method class.

Attributes:

* numScenarios -- The maximum number of scenarios allowed before this method ends.
* responseDelay -- The time it takes for the participant to read the options and click the submit button.
* doubleDelay -- A boolean value that indicates if the double delay method is activated. It is a modifier to the "Later" time such that the "Sooner" time's value is added to the "Later" time's value.

Operations:

*Name*:  calcIndifferencePoint()

*Arguments*:  None

*Returns*:  float

*Precondition*:  None

*Postcondition*: Calculates the indifference point and returns it as the result of the method.

*Exception:*  None

*Name*:  calcTotalTime()

*Arguments*:  None

*Returns*:  float

*Precondition*:  None

*Postcondition*: Calculates the total time it took to complete each method.

*Exception:*  None

### Double Limit Subclass

Name: doubleLimit

Description: A subclass of the Method class.

Attributes:

* doubleDelay -- A boolean value that indicates if the double delay method is activated. It is a modifier to the "Later" time such that the "Sooner" time's value is added to the "Later" time's value.
* responseDelay -- A float representing the delay in seconds for the Submit button to be enabled.
* numScenarios -- An integer that represents the number of Scenarios in the method.

Operations:

*Name*:  calcIndifferencePoint()

*Arguments*:  None

*Returns*:  float

*Precondition*:  None

*Postcondition*: Calculates the indifference point and returns it as the result of the method.

*Exception:*  None

*Name*:  calcTotalTime()

*Arguments*:  None

*Returns*:  float

*Precondition*:  None

*Postcondition*: Calculates the total time it took to complete each method.

*Exception:*  None

### Multiple Choice Subclass

Name: multipleChoice

Description: A subclass of the Method class.

Attributes:

* pointDistType -- An integer representing the distribution type.  For example ("1"=linear, "2"=exponential base n, etc.)
* responseDelay -- A float representing the delay in seconds before the Submit button is enabled.
* escape -- A boolean representing whether or not the participant escaped from the experiment.

Operations: There are no special operations for this method.

### Real Time Subclass

Name: realTime

Description: A subclass of the Method class.

Attributes:

* pointDistType -- An integer representing the distribution type.  For example ("1"=linear, "2"=exponential base n, etc.).
* escape --A boolean representing whether or not the participant escaped from the experiment.
* totalPoints --A float representing the total points earned while reading.  This will also be the indifference point.
* wordsRead -- An integer representing the total number of words the participant read.
* numStop -- An integer representing the number of times the participant stopped reading for more than x seconds.
* wordsSkipped -- An integer representing the number of words the participant skipped while reading.

Operations: There are no special operations for this method.

### Method User Interface Parent Class

Name: methodUI

Description: This class is the user interface module for the method window. It will control how the method is displayed to the user.

### Decreasing Adjustment User Interface Class

Name: decreasingAdjustmentUI

Description: This class is the user interface module for the decreasing adjustment window. It will control how the decreasing adjustment is displayed to the user.

### Double Limit User Interface Class

Name: doubleLimitUI

Description: This class is the user interface module for the double limit window. It will control how the double limit is displayed to the user.

### Multiple Choice User Interface

Name: multipleChoiceUI

Description: This class is the user interface module for the multiple choice window. It will control how the multiple choice is displayed to the user.

### Real Time User Interface Class

Name: realTimeUI

Description: This class is the user interface module for the real time window. It will control how the real time is displayed to the user.

### Scenario Class

Name: scenario

Description: This class contains multiple options for each scenario.

Attributes:

* List <Option> -- A list of options for each scenario.
* choice -- The Option that was selected by the participant.

Operations: There are no special operations for this method.

### Scenario User Interface Class

Name: scenarioUI

Description: This class is the user interface module for the scenario window. It will control how the scenario is displayed to the user.

### Option Class

Name: option

Description: This class will contain a reward and a time.

### Option User Interface Class

Name: OptionUI

Description: This class is the user interface module for the option window. It will control how the option is displayed to the user.

### Reward Class

Name: reward

Description: This class contains the value and units of the reward.

Attributes:

* value -- An integer representing the reward.
* units -- An enumerated data type for the reward.  This will include POINTS, MONEY, etc.

Operations: There are no special operations for this method.

### Time Class

Name: time

Description: This class contains the value and units of the time.

Attributes:

* value -- An integer representing the time.
* units -- An enumerated data type for the time.  This will include HOURS, MINUTES, and SECONDS.

Operations: There are no special operations for this method.

### Main Class

Name:  main

Description:  This class will contain the execution parameters and will extend a JFrame. It will contain the menu and functions for the menu items. It will display its own title until the experiment is loaded.

Attributes:  This class will not contain anything other than internal objects and variables.

Operations:  There are no special operations for this method.

### Database Class

Name: database

Description: A class that provides interaction with the database.

Attributes:

* username -- The user's username that will be verified with the database
* password -- The user's password that will be verified with the database
* privilege -- The user's privilege such as administrator or experimenter

Operations:

*Name*:  verifyLogin()

*Arguments*:  username, password

*Returns*:  int

*Precondition*:  None

*Postcondition*: Compares the username and password entered with those in database, if valid it returns a 1, and then returns a 1 for administrator privilege and a 2 for experimenter privilege

*Exception:*  None

*Name*:  insertIntoDB()

*Arguments*:  experiment

*Returns*:  void

*Precondition*:  None

*Postcondition*: Takes experiment and inserts data into the database

*Exception:*  None

*Name*:  createDBUI()

*Arguments*:  none

*Returns*:  void

*Precondition*:  None

*Postcondition*: When called will load a JPane with the admin functions

*Exception:*  None

### Experiment Loader

Cass Name: ExperimentLoader

Class Description: This class checks that the xml file is valid and loads it into the application.

Attributes: Experiment- An enumerated datatype- the xml file with the desired experiment

Operations:

*Name:* validate ()

*Description:* This method takes a Java string of the filename (the XML filename) and returns a string. If the file is valid, it returns NULL. If validate returns something other than NULL, the "load" method will throw the InvalidXMLFile exception.

*Arguments:* xml filename (string)

*Returns:* string; if the string is NULL, the xml file is valid

*Precondition:* None

*Postcondition:* None

*Exceptions:* None

*Name:* load ()

*Description:* It takes the xml filename string and returns (or loads) a thing of type "Experiment," which is an enumerated datatype. After this method has executed, the desired experiment should be loaded.

*Arguments:* xml file (string)

*Returns:* Experiment

*Precondition:* the validate method must execute before this method does

*Postcondition:* None

*Exception:* FileNotFoundException, IOException, InvalidXMLException

### Database Table and Field Descriptions

#### Investigator

This table stores the data for the investigators (experimenters, administrators, experiment creators). The following data are stored for each experimenter:

* username -- This is the login name the investigator provides when logging in to the system.
* password -- This is the password the investigator provides along with username to log in.
* first\_name -- This is the investigator’s first name.
* last\_name -- This is the investigator’s last name.
* department -- This is the department for which the investigator works
* privilege -- This is a foreign key that points to a table which stores the privilege level for each investigator (see Privilege table).

#### Privilege

This table stores the level of access that an investigator has to the system. Presently, the only available privilege levels are Administrator and User. Users may only load experiments and act as experimenters. Administrators may do this, as well as perform database queries, add, delete or modify users, or perform other tasks. An investigator’s level of access is stored in a field called **type**.

#### Investigator-Experiment

This table stores ID numbers linking a specific investigator to a specific experiment.

* iid -- This is a foreign key that points to a table which stores the data for the investigator who presided over a given experiment.
* eid -- This is a foreign key that points to a table which stores the data for a given experiment.

#### Experiment

This table stores the data for a given experiment. The following data are stored for each experiment:

* title -- This is the name that is given to a particular experiment.
* date\_time\_start -- This is the date and time that the given experiment began.
* date\_time\_end -- This is the date and time that the given experiment ended.
* exp\_num -- This is a foreign key that points to a table which stores the ID number for a given experiment and the ID numbers of any methods that experiment included.

#### Experiment-Method

This table stores ID numbers linking a given experiment to the methods it included.

* eid -- This is a foreign key that points to a table which stores the data for a given experiment.
* mid -- This is a foreign key that points to a table which stores the data for a specific method.

#### Method

This table stores the data for a method. The following data are stored:

* type -- This is the method type. The type of method may be double limit (DL), decreasing adjustment (DA), real time (RT), multiple choice (MC), or a combination thereof.
* max\_time -- This is not a required field.
* resp\_delay -- This is the amount of time that elapses between the moment a scenario is presented and the moment a response is submitted. This is not a required field.
* min\_reward -- This is the minimum reward value that a method will present to the participant. This is not a required field.
* max\_points -- This is the maximum reward value that a method will present to the participant. This is not a required field.

#### Method-Scenario

This table stores the ID numbers for a scenario and the method through which that scenario was presented.

* mid -- This is a foreign key that points to a table which stores the data for a specific method.
* sid -- This is a foreign key that points to a table which stores the data for a specific scenario.

#### Scenario

This table stores a reference to another table which stores the participant’s response to a given scenario. This reference is stored in a field called **choice**. This field’s value may be set to null because one or more methods may not present the participant with any options, while another method may present the partcipant with many options.

#### Scenario-Option

This table stores the ID numbers for a scenario and the option that was selected.

* sid -- This is a foreign key that points to a table which stores the data for a specific scenario.
* oid -- This is a foreign key that points to a table which stores the data for the answer choice that was chosen by the participant.

#### Option

A scenario in a DL, DA or MC method will present the participant with more than one response choice; these choices are called options. This table stores the following data for each option the participant selects:

* value -- This is the numeric value of the reward offered in the option the participant selected.
* units -- This is the unit value (such as minutes) of the reward offered in the option the participant selected.

#### Participant-Experiment

This table stores the ID numbers for a given experiment and the individual(s) who took part in it.

* eid -- This is a foreign key that points to a table which stores the data for a given experiment.
* pid -- This is a foreign key that points to a table which stores the demographic data for a participant in a given experiment.

### DiSH Class Diagram

(Double click to view all pages for all Visio diagrams below)

### Interface Design



### Database Diagram



### Experiment Loader Diagram

