ADDICTION ASSISTANT

TECHNICAL AND TESTING DOCUMENT
PHASE 4

Luke van Rooyen - 191094020935

Table of Contents

EXTERNALLY SOURCED CODE 3

EXPLANATION OF CRITICAL ALGORITHMS 3

'ADDING HOURS' 3

'VIEWING GROUPED DATA' 4

'VIEWING SUMMARY DATA' 5

ADVANCED TECHNIQUES 6

GETGROUPEDDATA 6

GETSUMMDATA 7

DELETING HOUSES, GRADES AND STUDENTS 8

TEST PLAN AND RESULTS 9

KEY 9

DATA ENTERING WINDOW 9

NEW GAME DIALOG BOX 9

UPDATE STUDENT DIALOG BOX 9

DATA WINDOW 9

ADMIN LOGIN WINDOW 9

EDIT HOUSES WINDOW 10

UPDATE HOUSE DIALOG BOX 10

NEW HOUSE DIALOG BOX 10

EDIT GRADES WINDOW 10

UPDATE GRADE DIALOG BOX 10

NEW GRADE DIALOG BOX 10

EDIT STUDENTS WINDOW 10

NEW STUDENT DIALOG BOX 11

EDIT GAMES WINDOW 11

UPDATE GAME DIALOG BOX 11

EDIT DATA WINDOW 11

EDIT STUDENT DATA WINDOW 11

Externally Sourced Code

The code that was used to connect to the SQL database from my Java program was sourced from Dominic Gruijters, my IT teacher.

The code implemented to populate the GUI tables in many of my windows was adapted and extended from a post on https://stackoverflow.com.

All other code was written entirely by me, using my knowledge of Java and SQL from the previous 3 years.

Explanation of Critical Algorithms

Addiction Assistant is comprised of two main functions: adding data using a particular student and the number of hours as well as viewing this data using various filters and groupings. Therefore three of my most critical algorithms, used for adding data and viewing it, are 'Adding Hours', 'Viewing Grouped Data', and 'Viewing Summary Data'.

'Adding Hours'

When the 'Save' button is pressed in the 'Data Entering' Window:

Tries to run the following code (if an error occurs, a message is shown to the user):

Saves the number of hours from the spinner as an integer.

If the value of the spinner is negative, an error message is displayed.

The selected Game object is retrieved from the game combo box.
An integer is created using the 'game_id' getter method from the game class, using the selected Game object.
Another integer is created using the 'student_id' getter method from the student class, using the selected Student object.

A DBManager method is run, given the three integers from above, which tries to add the passed integers to the Hours table in the database using an Insert Query. If no error occurs, the method returns a value of 'true' otherwise it returns a value of 'false'. This result is stored as a Boolean.

If the Boolean is true, the 'Saved' label is shown.

Else if the Boolean is false, the 'Saved' label is set to an error message and made visible.

End.

'Viewing Grouped Data'

When the 'Grouped Data' Window is opened or either of the combo boxes is changed this helper method will run:

Retrieves the current jTable model.
Deletes all the current rows from the table model.

Declares an ArrayList of GroupedData objects from a DBManager method, given the passed integers from the combo boxes. The method uses a Select Query and joins three tables from the database to extract the summarised data and instantiate a GroupedData object, which is added to the ArrayList.

Declares a new Object array of size 4.

Loops through the ArrayList from the DBManager method.

Assigns a string to the first position of the Object array using the 'full_name' getter method from the GroupedData class.
Assigns an integer to the second position of the Object array using the 'total_hours' getter method from the GroupedData class.
Assigns a string to the third position of the Object array using the 'game' getter method from the GroupedData class.
Assigns an integer to the fourth position of the Object array using the 'hours' getter method from the GroupedData class.

Adds the object array as a row to the table model.

End loop.

End.

'Viewing Summary Data'

When the 'Summary Data' Window is opened or either of the combo boxes is changed this helper method will run:

Retrieves the current jTable model.
Deletes all the current rows from the table model.

Declares an ArrayList of SummaryData objects from a DBManager method, given the passed integers from the combo boxes. The method uses one of three Select Queries, based on the passed integers, and either joins four tables with three nested queries; or joins three tables without nested queries; or joins four tables with two nested queries to extract the summarised data and instantiate a SummaryData object, which is added to the ArrayList.

Declares a new Object array of size 4.

Loops through the ArrayList from the DBManager method.

Assigns an integer to the first position of the Object array using the 'total_hours' getter method from the SummaryData class.
Assigns a string to the second position of the Object array using the 'highest' getter method from the SummaryData class.
Assigns a string to the third position of the Object array using the 'popgame' getter method from the SummaryData class.
Assigns an integer to the fourth position of the Object array using the 'hourspopgame' getter method from the SummaryData class.

Adds the object array as a row to the table model.

End loop.

End.

Advanced Techniques

getGroupedData

This method in my DBManger class is used to return an ArrayList of GroupedData objects to my 'Grouped Data' Window, given two integers from the local combo boxes.

The query returns the full name of the current student, as a string; the total number of hours for the current student as an integer; the name of the current student's most popular game by hours, as a string; and the number of hours for their most popular game.

As I mentioned earlier there are also two combo boxes in the window that are used to refine the data that is displayed and so depending on the 'mode' integer, the "WHERE" clause in the SQL query is changed to retrieve either a house or a grade. The 'filter' integer provides the ID of the specific house or grade that must be shown and is also used in the "WHERE" clause.

To retrieve the 'full_name' as a string I used a relatively straight forward join between tblStudents and tblHours on 'student_id'. The same was done to retrieve the 'game' as a string by joining tblGames and tblHours on 'game_id'. To retrieve the 'total_hours' I used a nested query that sums the hours column from the tblHours table for a given 'student_id'. The 'hours' integer was extracted straight from tblHours.

Due to the fact that I wanted only the most popular game and it's corresponding hours I also had to use the 'ORDER BY' and 'LIMIT' keywords in my query to return only the highest game by number of hours.

There was also another problem in that the algorithm had to return a list of all the students from an entire house or grade and this code only returns the game and corresponding hours for a single student at a time. Therefore, this SQL query was placed in a for loop that uses the 'getAllStudents' method to retrieve the total number of students in the database and then repeats the SQL query on each student, using an integer 'i'.

At the end of each loop a Grouped Data object is instantiated and added to the ArrayList, which is then returned to the window after the loop is complete and the results are displayed using a table.

getSummData

This method, also in my DBManager class, was also very complicated. It returns the total hours for a certain house or grade, the highest grade (if the 'house' mode is selected) or the highest house (if the 'grade' mode is selected), the most popular game for the highest house or grade, and the hours in the most popular game for that house or grade. Again there are two local combo boxes that allow the user to select whether to view the highest grade and therefore the highest house within that grade or the highest house and therefore the highest grade within that house. Due to this fact, the query and the three nested queries are modified using strings that are determined using two if statements that are activated based on the value of this 'mode' integer.

If 'grade' mode is selected, the "WHERE" clause is modified so that only a certain grade is extracted, using the 'filter' integer, and the "GROUP BY" and "SELECT" clauses are modified so that the data is grouped by 'house_id' and the 'house' name is extracted. If the 'house' mode is selected, the "WHERE" clause is modified so that only a certain house is extracted, using the 'filter' integer, and the "GROUP BY" and "SELECT" clauses are modified so that the data is grouped by 'grade_id' and the 'grade' name is extracted. Again I only wanted to retrieve the highest house or grade, their most popular game and its corresponding hours so I used the "ORDER BY" and "LIMIT" key words.

To extract the 'total_hours' I simply summed the 'hours' column from tblHours.

To extract the highest grade or house I used a nested query and joined tblStudents and tblHours on 'student_id'; tblStudents and tblGrades on 'grade_id'; and tblStudents and tblHouses on 'house_id' and then extracted only a certain house or grade using the 'filter' integer and a string determined in the if statements mentioned above. This nested query was then grouped by 'house_id' or 'grade_id', again determined by the if statements, and ordered by the sum of the hours column, descending, and restricted to the first result.

To extract the most popular game for the highest house or grade I used another nested query. I joined tblHours and tblGames on 'game_id' and tblHours and tblStudents on 'student_id'. I then restricted the results to only a certain house or grade using the 'filter' integer and a string (determined from the if statements). These results were then grouped by the 'game' field and ordered by the sum of the hours column, descending, and limited to the first result.

To extract the hours for the most popular game I used the same nested query as above and only changed the "SELECT" clause to retrieve the sum of the hours column for the highest game instead of the name of the highest game.

This data is then used to instantiate a SummaryData object which is added to an ArrayList and returned to the 'Summary Data' Window.

Deleting Houses, Grades and Students

Due to the way I chose to normalize my database many fields rely on many other fields in different tables, for example a student record contains a ‘house_id’ and ‘grade_id’ that refer to records in tblHouses and tblGrades, respectively.

Therefore, I had to add code in that accounted for this fact. When the user presses the ‘Delete’ button in the ‘Edit Houses’ Window or in the ‘Edit Grades’ Window a query is run that deletes the selected House or Grade as well as all the students that belong to that House or Grade, thus ensuring that a student record never refers to a House or Grade that does not exist anymore.

In much the same way, a tblHours record contains a ‘game_id’ and ‘student_id’ that refer to tblGames and tblStudents, respectively. Thus I added an SQL query that ensures that when a Game or Student is deleted all the associated data is deleted in tblHours as well. Again ensuring the database retains referential integrity.

Deleting the Houses and Grades is done with a simple deletion query in tblHouses or tblGrades. Deleting Students is also done with a deletion query but in tblStudents and using ‘house_id’ or ‘grade_id’ in the “WHERE” clause. Finally, Hours are deleted using a third deletion query in tblHours that deletes all records where the ‘student_id’ is not in a list extracted from tblStudents, using a nested query in the “WHERE” clause.

Deleting Games and Students is also done with a simple deletion query in tblGames or tblStudents, respectively. tlbHours is updated using either a simple deletion query with the ‘game_id’ in the “WHERE” clause or another deletion query with a nested “SELECT” query to remove any records where the ‘student_id’ is not in tblStudents.

Test Plan and Results

Key

Runs correctly. (no error thrown)	Error checking was successful. (message was displayed)	Program crashed.

Data Entering Window

GUI Component	Normal	Extreme	Erroneous
“Hours:” Spinner	9	0	-99

New Game Dialog Box

GUI Component	Normal	Extreme	Erroneous
“Game” Text Field	Clash of Clans	Cläsh of’Clans	*Blank*
“Publisher” Text Field	Supercell	Supēr’cell!	*Blank*

Update Student Dialog Box

GUI Component	Normal	Extreme	Erroneous
“Full Name” Text Field	Luke Van Rooyen	Lukę Adam van’Roòyen!	*Blank*

Data Window

GUI Component	Normal	Extreme	Erroneous
“Student Name” Text Field	Luke Van Rooyen	Lukę Adam van’Roòyen!	*Blank*

Admin Login Window

GUI Component	Normal	Extreme	Erroneous
“Username” Text Field	nely3	Në!ly3	*Blank*
“Password” Text Field	Jv87Fgc	Jv87Fgć!	*Blank*

Edit Houses Window

GUI Component	Normal	Extreme	Erroneous
“Delete” Button	*Record Selected*	*None*	*No Selection*
“Update” Button	*Record Selected*	*None*	*No Selection*

Update House Dialog Box

GUI Component	Normal	Extreme	Erroneous
“Current Name” Text Field	Founders	Fôunder’s-Hoüsé!	*Blank*

New House Dialog Box

GUI Component	Normal	Extreme	Erroneous
“New House” Text Field	Raiffe	Raíffe’s-Hoüsé!	*Blank*

Edit Grades Window

GUI Component	Normal	Extreme	Erroneous
“Delete” Button	*Record Selected*	*None*	*No Selection*
“Update” Button	*Record Selected*	*None*	*No Selection*

Update Grade Dialog Box

GUI Component	Normal	Extreme	Erroneous
“Current Name” Text Field	Grade 9	Grâde’9!	*Blank*

New Grade Dialog Box

GUI Component	Normal	Extreme	Erroneous
“New Grade” Text Field	Grade 7	Grâde’7!	*Blank*

Edit Students Window

GUI Component	Normal	Extreme	Erroneous
“Delete” Button	*Record Selected*	*None*	*No Selection*
“Update” Button	*Record Selected*	*None*	*No Selection*

New Student Dialog Box

GUI Component	Normal	Extreme	Erroneous
“New Name” Text Field	Luke Van Rooyen	Lukę van’Roòyen!	*Blank*

Edit Games Window

GUI Component	Normal	Extreme	Erroneous
“Delete” Button	*Record Selected*	*None*	*No Selection*
“Update” Button	*Record Selected*	*None*	*No Selection*

Update Game Dialog Box

GUI Component	Normal	Extreme	Erroneous
“Name” Text Field	Clash of Clans	Cläsh of’Clans	*Blank*
“Publisher” Text Field	Supercell	Supēr’cell!	*Blank*

Edit Data Window

GUI Component	Normal	Extreme	Erroneous
“Student’s Name” Text Field	Luke Van Rooyen	Lukę van’Roòyen!	*Blank*

Edit Student Data Window

GUI Component	Normal	Extreme	Erroneous
“Delete” Button	*Record Selected*	*None*	*No Selection*