QuickMaths

Project

Samuel Jothimuthu

5/9/2018

In depth manual for the source code of the Game.

Contents

1. Specification

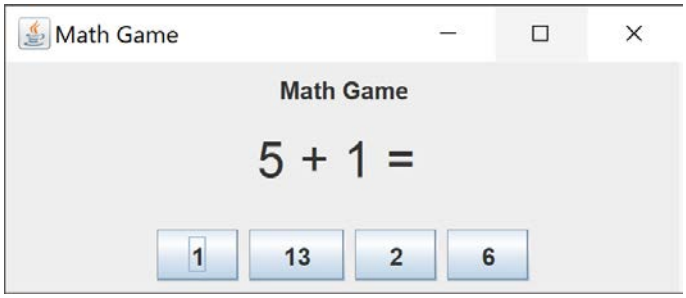
This is QuickMaths. The game consists of elementary level arithmetic (addition) questions in repeated succession until 60 seconds are up. The amount correct is stored along with the player's name to generate a list of Top Scores and individual scores. The list of Top Scores reveals the top scores across all players in a simple and easy to read Table. The user can also search his/her name to find all the scores they have made. Then finally there is an exit button to allow the user to exit the program.

Features

1. Stores all information so that it can be accessed later
2. Very simple and intuitive UI so that students are not struggling to understand the program.
3. Fun to play and helps students learn basic math quickly.

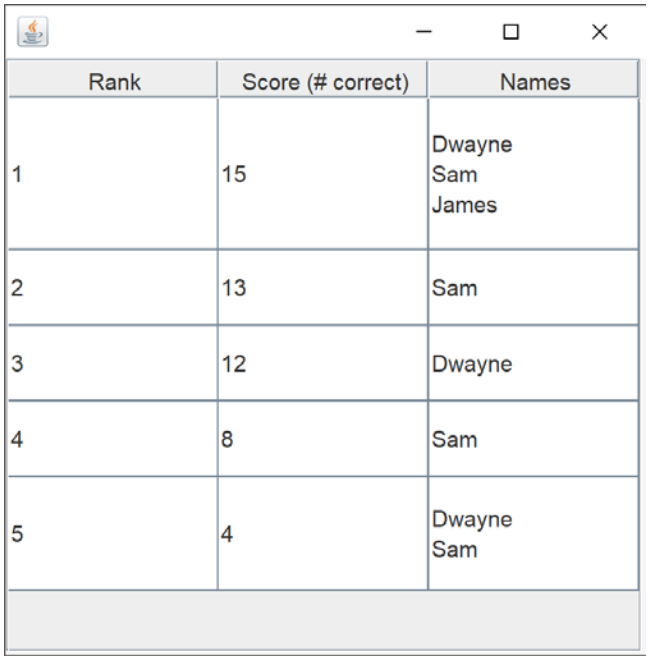
2. Analysis

1) When the program begins the user will be greeted with a page that tells the user the instructions. It provides the user with 4 button choices. The first is Start which begins the game. The second is Top 5 Scores which displays exactly that. Search Scores allows the user to search for their scores on the test. And Exit is for exiting.



2) This is the questions panel. It displays the question and 4 possible choices in a random order. After every selection a new window pops up and the correct or incorrect counter goes up based on the users answer.

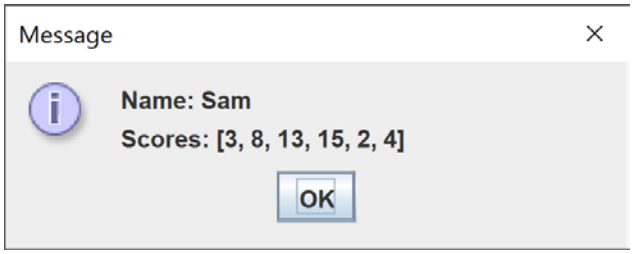
3) This is the table that displays the top 5 scores. It allocates individual row height based upon a calculation done to ensure the table is neither too big nor too small. It displays the rank, the score from highest to lowest and all the names that have scored that value.

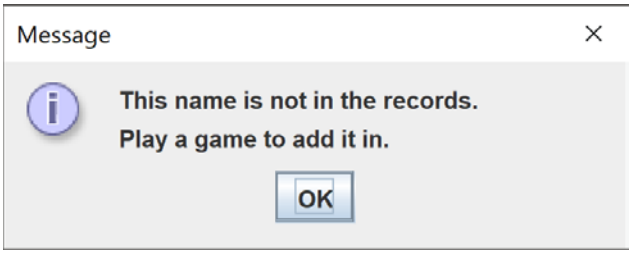


Rank	Score (# correct)	Names
1	15	Dwayne Sam James
2	13	Sam
3	12	Dwayne
4	8	Sam
5	4	Dwayne Sam

4) This is the search panel. When the user enters their name and press OK then the next window is pulled up.

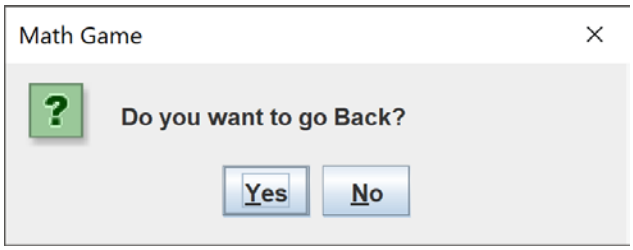
5) This window is after the user submits the name. Each of the scores is submitted. They are stored in the order of first to last.





6) This is displayed if the name cannot be found.

7) This is displayed when the user selects Exit on the main Page. It gives the user a second chance before closing the application down for good.



These are all the intractable elements of the program. There are several elements such as data structures, File input – output, and so much more that will be discussed in the Design section.

3. Design

Classes:

1) MathGame:

MathGame is the main operation class that holds most of the code. It is where the GUIs are created along with the Math Problems. It has 3 constructors, 17 methods, and a private class (ButtonListener).

2) ButtonListener:

This is a private class in MathGame that implements ActionListener. It is for the buttons in the GUI. When the Buttons are pressed an object of this class is created. Whatever Button is pushed is stored as an action event and turned into a string which is checked against multiple if statements to determine the next step in the programs execution.

3) MathGameApp:

This class is the driver class for the whole project. It calls the No Argument Constructor from MathGame to begin the program.

4) Table:

Table is another operation class for creating a table. This table is for displaying the Top 5 Scores of all players. A special feature of it is that creates the specific row height in case multiple users scored the same. The image in the previous section shows this.

Methods:

1)public void Problem():

This function creates two random numbers from 1 to 10 and adds them to create an answer. This is the problem that will be displayed in the GUI.

2) public void createQuestion():

Calls Problem() and initializes problem(String) using the values generated in Problem().

3) public void Question():

Similar to Problem() and initializes ch1(String) & correctAnswer(String). Ch1 is used in the GUI panel to display the first choice.

4) public void Answers():

This function does 3 things

1) It makes sure no duplicate answers are created. In the do-while loop the randomly generated wrong answers are checked against the right answer to see if they are the same. If they are, the wrong answer is regenerated. The loop continues until all the answers are different. Then the answers are converted into strings.

2) It adds the values to the Arraylist. The generated strings are added to the Arraylist.

3) Shuffles them in a random order so they are not displayed in the same order each time. The shuffle method randomly sorts the elements in the Arraylist.

5) private static final Countdown():

This is simply counts down from whatever number of seconds is specified. For my game that number is 60. It uses the timer class. It also initializes correct, incorrect, and total which was for debugging purposes.

6) private void readScores():

This reads the information in the file topScores.txt. It takes the information and adds it to two different Tree Maps (topScores, userNames). It uses a getCanonicalPath() method that gets the current directory the project is in.

7) public void displayNames():

This is a display function I use for debugging. Displays the topScores and userNames Tree Maps.

8) public void updateScores(int score, String name) throws IOException

This writes the new score and name into the file topScores.txt

9) public void search() throws IOException

This is a search function. A JOptionPane with a text input option is displayed. The exact look can be seen in the previous section. It handles if the user enters the wrong name, no name, and selecting cancel. If the user enters a name it displays the scores associated with that name in the order of first score earned to last score earned.

10) public void display(String name, ArrayList<Integer> arrayList)

This displays the scores that are found in the search() function. It is called in that function. It creates a string with the information and the displays it on a JOptionPane.

11)public void addtopScores(int correct, String name)

This function is called in readScores. It creates a Arraylist for each score. If the Arraylist exists it adds a name to it. If it does not it creates a new one.

12) public void adduserNames(String name, Integer correct)

This functions the same way as the addtopScores but it stores the data with the key as the name and the Arraylist as the scores associated with the name.

13) public void getUserName()

This function is called after the gams is played so that the score can be stored with a name. It is equipped to handle an empty string and if the user presses cancel.

14) public void setData()

This function is for the table. It generates the 2 dimensional array that is used to create the value. First it creates an array of the keys from topScores (TreeMap). Then it displays the rank in the first column, and the top scores in the second

column. The third column takes the Arraylist associated with the key and creates a temporary array called name. Using the string builder class and html tags I build the string with each name on next line. To accommodate for this the row's height need to be adjusted accordingly to fit all the names in the cell. To do this I created an array that would hold each rows individual height. Every time a name would appear in a new line the row height would be adjusted accordingly. This array is passed into Table's only 3 Argument constructor.

15) public buildpanel() and mainpanel()

Both these functions build the GUI's with all the elements generated from previous methods.

16) public void actionPerformed(ActionEvent e)

This function responds to the buttons being pushed calling previously mentioned functions if the corresponding button is pressed. It also resets the timer once the game ends. It is where Table's constructor is called. It also contains the exit operation.

17) public void sound()

It plays a sound if the answer is correct. The sound is ding.

Constructors:

1) public MathGame()

This constructor is the first one that is called in execution. It creates the start page (mainpanel()) and calls readscores() (explained in the previous section).

2) public MathGame(int i)

This constructor is called when start is pushed on the previous panel. It create the panel with the math problem on it and starts the countdown.

3) public MathGame(int i, int j)

This constructor is called for subsequent problems to be created. It calls buildpanel().

4) public Table(Object[][] data, Object[] colNames, int[] rowSize)

This is the constructor for table. It is called in the ButtonListener. It takes in 3 parameters. The first is a 2D array that houses the data of each cell. The next is an array that holds the name of the columns. The final array holds the height of each row that was created in setData(). The ability to edit the tables has been turned off. And the for loop inside is used to set the row height.