<u>COVER PAGE</u>

OHLONE COLLEGE

PROJECT NAME:

MATH GAME

TEAM MEMBERS:

ALEXANDER AMORES
VIKRAM ELANGO
JEFFREY GU
RYAN KAO

CLASS:

CS-170 Section 1

INDEX

1. WORK SPLIT:

a. ***ALEXANDER**:* Developed and coded the basic structure of the algorithm and set the stage for others to implement their codes.

b. ***VIKRAM**:* Added the music as well as implemented some of the GUI components into the code and maintained the team log.

c. ***JEFFREY**:* Developed the Hiscore file to read, display, and store scores from and to a text file "scores.txt" and implementing the high score capabilities to the game as well as some GUI components. Also polished the game in various areas to ensure game quality.

d. ***RYAN**:* Randomly generated math questions; overall design of the game by adding images, background color, font, and setting boundaries for buttons

2. DETAILED DESCRIPTION:
We designed a math game that covers basic addition, subtraction, and multiplication in a set of 10 questions. When first starting the program, the user will be greeted with our main menu screen, with a start and exit button, and a button that displays the high scores before they played. If the user clicks the start button, the game will then prompt the user to enter their name and play music, if they want to. After that, ten different questions will be displayed on the screen one at a time, and the user will have to enter their answer in a text box and submit through a button. At the end of the game, the top 5 scores will then be presented to the player, as well as a play again button and a quit button.

As for the code description, we have three files: Mathgame, Hiscore, and Source. More details will be explained in the next page.

3. DEFINITION OF EVERY CLASS

MathGame.java:

The MathGame class is an operation class that contains all methods needed to establish our GUI application, including components and event handlers, and generate a random question for the math game.

a. TreeMap<Integer, List<String>> readLeaders():
   The readLeaders method creates a Treemap that will obtain the data returned from the readScore method in the Hiscore class, which will read the scores.txt file to retrieve and store all names and scores in a TreeMap.

b. startGame():
   The startGame method establishes the math game application by setting the frame, background color, and font, adding components to the frame, and assigning buttons to an action listener for event handling. The method also creates the start menu of the game, containing the welcome screen, button for music, start and exit button, and a button for displaying the top 5 scores.

c. questionGenerator():
   The questionGenerator method generates random questions by assigning random numbers to two numbers of an equation. The operator is also randomized to expand the question types of the problems: addition, subtraction, and multiplication. If the problem is an addition or subtraction problem, the number assigned will be between 1 and 50; if the problem is a multiplication problem, the maximum number will be 10 to make it easier for children. Our group had also decided to not implement any division problems to reduce complications if the answer had any remainders.

d. exitActionListener:
   The exitActionListener exits the game if the exit button is clicked at the start menu or the scores page after the game.

e. startActionListener:
   The startActionListener will be initiated once the user chooses the start button from the start menu. The user will be prompted to enter their name. The event handler mainly sets bounds for the components, sets fonts for the buttons, and adds an image.

f. scoreActionListener:
   The scoreActionListener will display the top 5 scores by calling the display method in the Hiscore class. The event handler will also set bounds and fonts, and add 2 images to the frame.

g. nameActionListener:
   The nameActionListener sets the bounds for the label and buttons asking the user to play music or not. The action listener will be called after the user enters his or her name.

h. musicActionListener:
   The musicActionListener allows the user to decide whether they want music or not. A music file got Loop.wav is imported. Two buttons called "yes" and "no" respectively are on the frame, if the user clicks the "yes" button the music starts with the help of the clip.start() function, if "no" is clicked then no music is played and it goes to the first question.

i. mathActionListener:
   The mathActionListener will continue the game after the first question. The score will be incremented if the answer to the equation is correct. After 10 questions are up, the top 5 scores will be displayed at the end. The user will then be given the option to play again or exit the game.

Hiscore.java:

The Hiscore class is an operation class that contains the methods that handle the high score part of the application, with I/O capabilities so that the scores will be saved between multiple execution of the program.

TreeMap<Integer, List<String>> storeScore (String name, int score, TreeMap<Integer, List<String>> hiscores):

This method is used to store a score to parameter TreeMap<Integer,List<String>> hiscore and return it. It will check if the key (score) exists in the TreeMap, if it doesn't, it will simply put the score and name to the TreeMap, otherwise, it checks the value of the TreeMap, which is a list, to see if the list contains the parameter name. If a match is found, it will remove and re-add the name to the end of the list so that it's considered a recent entry, else if the list reaches the end of its index, add the name to the end. At the end, the program will put the score (key) and list of names (values) who obtained that score to parameter hiscore and return the hiscore variable. This method is used to store the score when reading the scores.txt, and when the user finishes a game so that we may locally store the score within the program.

String display (TreeMap<Integer, List<String>> hiscores):

This method is used to return a string to display high scores. It first iterates by scores (key) from highest to lowest, then iterates the list of names (values) of the score that obtained it, starting at the end of the list so that it displays recent players first. It will add to a string for the top 5 scores to be displayed, and ends both loops early once 5 names and corresponding scores are in the string, or at the end of their indexes if scores.txt does not have 5 names and scores. After the end of the loop, it will return the string that contains the 5 high scores of the game to be displayed. This method is used when displaying the high scores.

void writeScore(String name, int score):

The method to write a new score to scores.txt. It will write in the file the parameter name and score in the format (name:score) with a newline to determine a new score in the file. This method is used when a player has finished their game, writing their name and score to the scores.txt file.

TreeMap<Integer, List<String>> readScores():

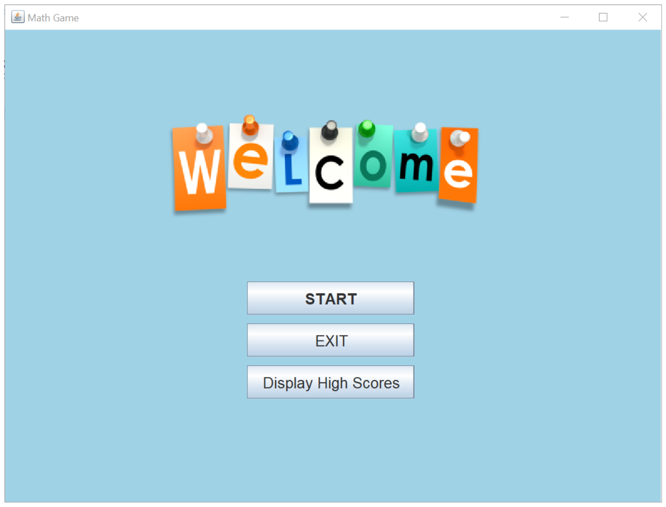
The method to read the scores.txt. It first initializes a TreeMap<Integer, List<String>> variable named readScore with the Collections.reverseOrder() method so that scores are sorted from highest to lowest. Then initializes needed variables name, score, and line. It uses BufferedReader to read scores.txt, then sets line to the first line in scores.txt. We have a while loop that ends when line is null (i.e finished reading scores.txt). In this while loop, because scores.txt is formatted in 'Name:Score', we set the name variable to the string before the character ':', and score to the value after ':', and calls the storeScores method, which TreeMap variable readScore is set to, to put the read data to readScore. After the scores.txt is fully read, readScore variable will have the entire data of scores.txt, and will be returned. This method is used for the initial start of the program, in order to read the previous saved scores of the game.

Source.java:

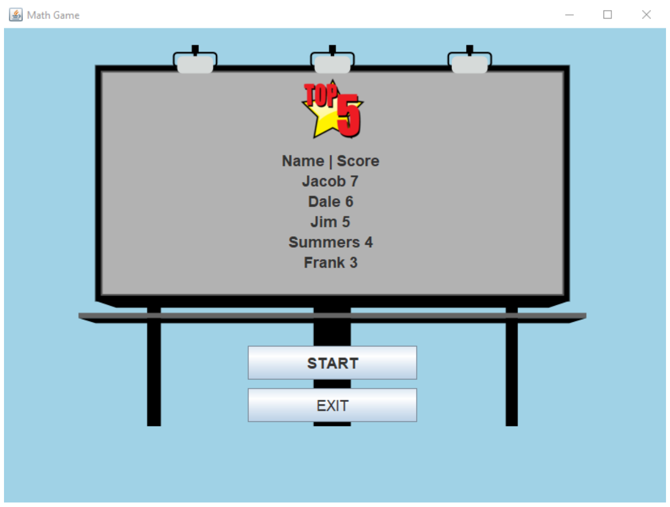
The Source class is a driver class that creates an instance of the MathGame class. The startGame method from the class will then be called to start the game.

4. SCREENSHOTS OF OUTPUT:

The Starting Screen of the game. Has 3 buttons for the player to play, exit, and display the high scores of the game

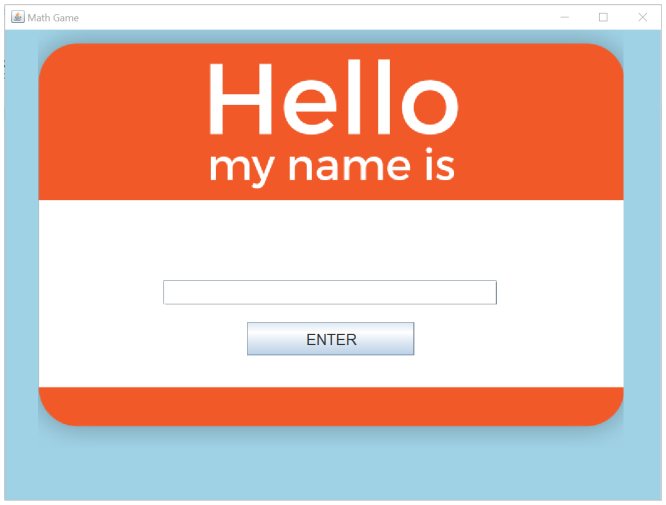


Output when the user presses Display High Score, shows the high scores that exist in scores.txt in order from highest to lowest scores, with two buttons to either start or exit the game.

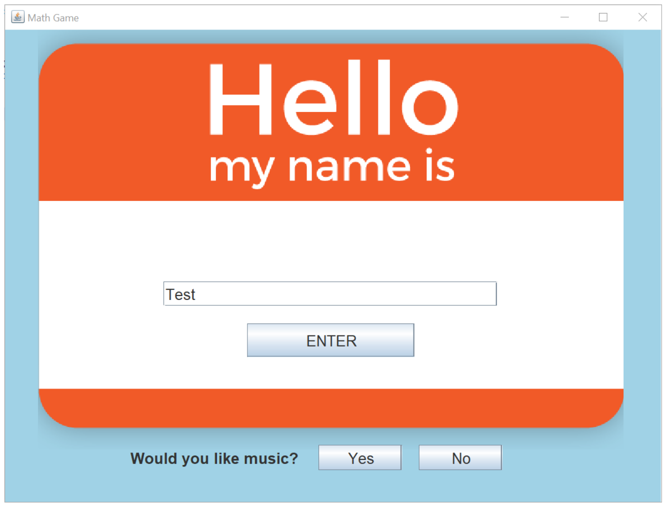


```
Name | Score
Jacob 7
Dale 6
Jim 5
Summers 4
Frank 3
```

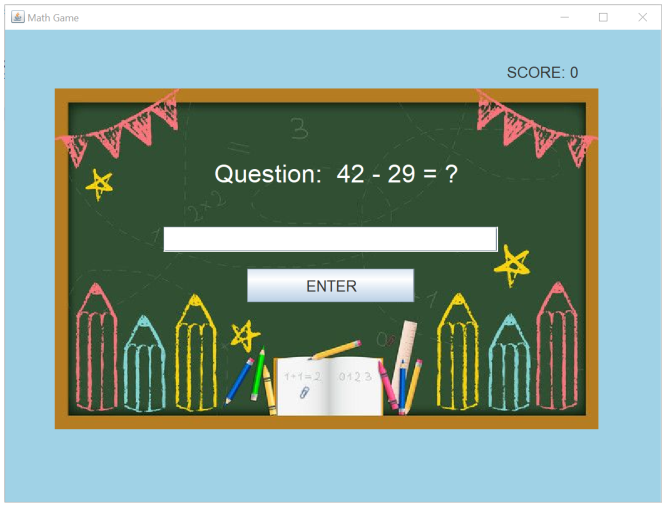
Screen after the user presses start on the start screen or the high score screen, prompts them to enter their name in the textbook with an image



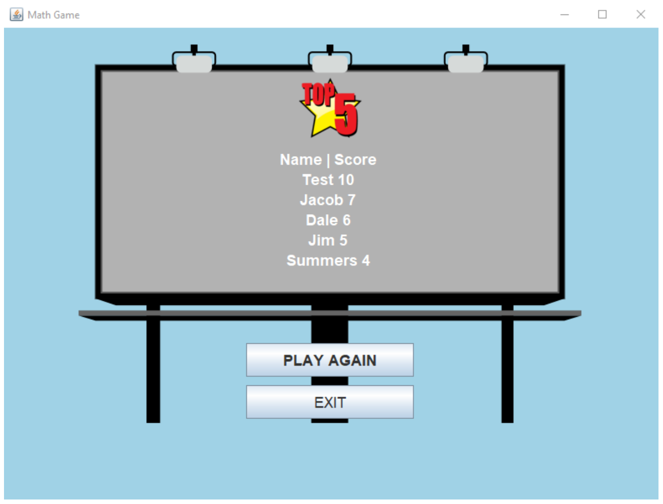
Screen after the user entered their name, prompts the user if they wish for music in their game. Clicking yes starts the game with music, while clicking no starts the game with no music



Screen that will have the gameplay. The game will randomly generate an addition, subtraction, or multiplication problem 10 times. The multiplication problems has been tuned to have lowered values (1-10) to decrease difficulty. If the player enters the correct answer, the score is incremented by 1 on the top right, else if the incorrect answer is entered, score is not incremented, for a possible perfect score of 10.



Screen after the player has answered 10 questions. The program will store their name and score to the scores.txt file, and display the high scores. If the player got a top 5 score, the program will display the new high score, in order by highest to lowest score. Play again button will set them back at the screen to enter their name, and exit button will exit the math game.



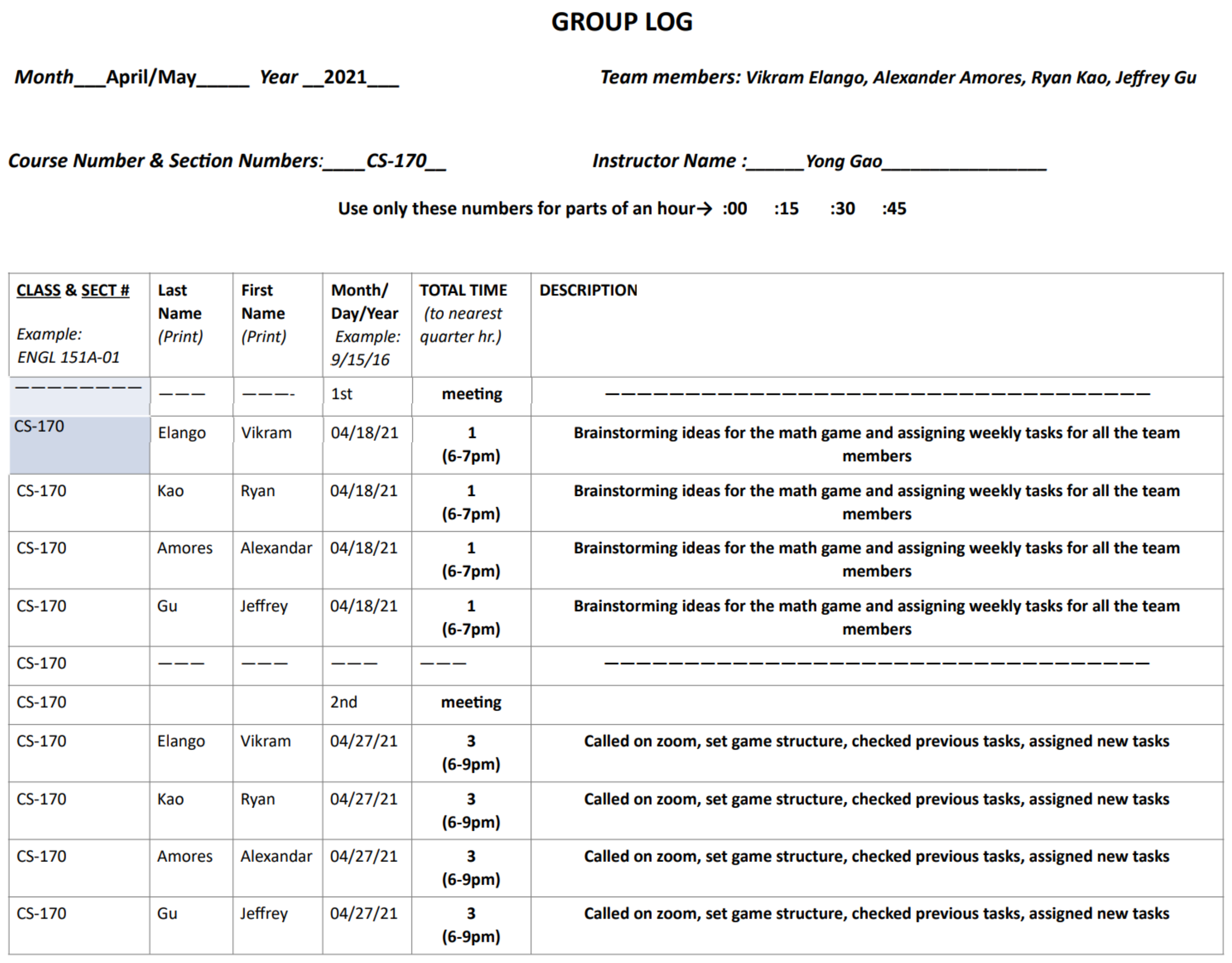
Name | Score
Test 10
Jacob 7
Dale 6
Jim 5
Summers 4

5. MEETING LOGS:

Notes:

If this meeting log is difficult to read for any reason, a separate pdf version of the meeting log is included with this pdf in the same file.

All meeting locations took place online via Zoom calls



GROUP LOG

Month*___April/May_____ *Year* __2021___** ***Team members: Vikram Elango, Alexander Amores, Ryan Kao, Jeffrey Gu

***Course Number & Section Numbers:*____*CS-170*__** ***Instructor Name :*______*Yong Gao*__________________**

Use only these numbers for parts of an hour→ :00 :15 :30 :45

CLASS & SECT # *Example: ENGL 151A-01*	**Last Name** *(Print)*	**First Name** *(Print)*	**Month/ Day/Year** *Example: 9/15/16*	**TOTAL TIME** *(to nearest quarter hr.)*	**DESCRIPTION**
————————	———	———-	1st	**meeting**	————————————————————————————————
CS-170	Elango	Vikram	04/18/21	**1 (6-7pm)**	**Brainstorming ideas for the math game and assigning weekly tasks for all the team members**
CS-170	Kao	Ryan	04/18/21	**1 (6-7pm)**	**Brainstorming ideas for the math game and assigning weekly tasks for all the team members**
CS-170	Amores	Alexandar	04/18/21	**1 (6-7pm)**	**Brainstorming ideas for the math game and assigning weekly tasks for all the team members**
CS-170	Gu	Jeffrey	04/18/21	**1 (6-7pm)**	**Brainstorming ideas for the math game and assigning weekly tasks for all the team members**
CS-170	———	———	———	———	————————————————————————————————
CS-170			2nd	**meeting**	
CS-170	Elango	Vikram	04/27/21	**3 (6-9pm)**	**Called on zoom, set game structure, checked previous tasks, assigned new tasks**
CS-170	Kao	Ryan	04/27/21	**3 (6-9pm)**	**Called on zoom, set game structure, checked previous tasks, assigned new tasks**
CS-170	Amores	Alexandar	04/27/21	**3 (6-9pm)**	**Called on zoom, set game structure, checked previous tasks, assigned new tasks**
CS-170	Gu	Jeffrey	04/27/21	**3 (6-9pm)**	**Called on zoom, set game structure, checked previous tasks, assigned new tasks**

MORE BELOW

GROUP LOG

Month*___April/May_____ *Year* __2021___** ***Team members: Vikram Elango, Alexander Amores, Ryan Kao, Jeffrey Gu

***Course Number & Section Numbers:*____CS-170__** ***Instructor Name :*______Yong Gao________________**

Use only these numbers for parts of an hour→ :00 :15 :30 :45

CS-170	———	———	———	———	————————————————————————————————
CS-170					
CS-170			3rd	**meeting**	
CS-170	Elango	Vikram	05/01/21	**3 (6pm-9pm)**	**checking previous tasks (music), assign new tasks, compiling everyones work**
CS-170	Kao	Ryan	05/01/21	**3 6pm-9pm)**	**checking previous tasks (question generator), assign new tasks, compiling everyones work**
CS-170	Amores	Alexandar	05/01/21	**3 (6pm-9pm)**	**checking previous tasks (main structure), assign new tasks, compiling everyones work**
CS-170	Gu	Jeffrey	05/01/21	**3 (6pm-9pm)**	**checking previous tasks (hiscores), assign new tasks, compiling everyones work**
CS-170	———	———	———	———	————————————————————————————————
			4th	**meeting**	
CS-170	Elango	Vikram	05/05/21	**4 (8pm-12am)**	**compiling everyones work, set base game with system.out**
CS-170	Kao	Ryan	05/05/21	**4 (8pm-12am)**	**compiling everyones work, set base game with system.out**
CS-170	Amores	Alexandar	05/05/21	**4 (8pm-12am)**	**compiling everyones work, set base game with system.out**
CS-170	Gu	Jeffrey	05/05/21	**4 (8pm-12am)**	**compiling everyones work, set base game with system.out**
CS-170	———	———	———	———	————————————————————————————————

MORE BELOW

GROUP LOG

Month*___April/May_____ *Year* __2021___** ***Team members: Vikram Elango, Alexander Amores, Ryan Kao, Jeffrey Gu

Course Number & Section Numbers:____CS-170__ ***Instructor Name :______Yong Gao_________________***

Use only these numbers for parts of an hour→ :00 :15 :30 :45

			5th	**meeting**	
CS-170	Elango	Vikram	05/07/21	**5 (7-12am)**	**starting GUI components**
CS-170	Kao	Ryan	05/07/21	**5 (7-12am)**	**starting GUI components**
CS-170	Amores	Alexandar	05/07/21	**5 (7-12am)**	**starting GUI components**
CS-170	Gu	Jeffrey	05/07/21	**5 (7-12am)**	**starting GUI components**
CS-170	———	———	———	———	—————————————————————————————————
			6th	**meeting**	
CS-170	Elango	Vikram	05/11/21	**4 (8pm-12pm)**	**completed GUI components and added finishing touches**
CS-170	Kao	Ryan	05/11/21	**4 (8pm-12pm)**	**completed GUI components and added finishing touches**
CS-170	Amores	Alexandar	05/11/21	**4 (8pm-12pm)**	**completed GUI components and added finishing touches**
CS-170	Gu	Jeffrey	05/11/21	**4 (8pm-12pm)**	**completed GUI components and added finishing touches**
CS-170	———	———	———	———	—————————————————————————————————

GROUP LOG

Month*___April/May_____ *Year* __2021___** ***Team members: Vikram Elango, Alexander Amores, Ryan Kao, Jeffrey Gu

Course Number & Section Numbers:____CS-170__ ***Instructor Name :______Yong Gao_________________***

Use only these numbers for parts of an hour→ :00 :15 :30 :45

Total Number of Hours	20 hours

END OF DOCUMENT