Design Document

etaMath

Revision 1.0

Table of Contents

Revision History

Version	Name	Reason For Changes	Date
1.0	*Aigbedion Oghomwen*	*Initial Revision*	*11/17/2019*

Approved By

Name	Signature	Department	Date

1. Introduction

1.1 Purpose

To design a game called “etaMath”.

1.2 System Overview

This project involves designing a game to help pupils improve their mental arithmetic skills. The application is shipped without any graphical user interface. Instead, a text-based command-line tool shall be developed.

2. Design Considerations

All design considerations were handled in etaMath.

2.1 Assumptions

No assumptions were made.

2.2 Constraints

- RQ_01: The application shall perform a "mental arithmetic’s"
- - test, build of 20 Tasks. Where each task shall be one of the types defined in RQ_02 – RQ_05.
- RQ_02: Addition “with Overflow” of calculations resulting in a range of 21 to 99.
- RQ_03: Subtraction as 'reversion task' of RQ_02.
- RQ_04: Multiplication of the multiplication tables 3, 4, 5, 6, 7, 8, 9, 11, 12
- RQ_05: Division as 'reversion task' of RQ_04.
- RQ_06: Within a single run, no task shall be repeated
- RQ_07: Is the context of RQ_06: A * B == B * A
- RQ_08: In the context of RQ_06: A + B == B + A
- RQ_09: The user shall have 10 sec. of time for answering each task
- RQ_10: The application shall maintain a highscore-list
- RQ_11: The high score shall be ordered by correct answers
- RQ_12: The high score shall be ordered by 'used time' in case correct answers where equal.

Given the above constraints, only 2126 questions are possible in the game.

Clarifications

- Overflow The Term ‘Overflow’ in RQ_02 is understood as ‘decimal overflow’ such as:
  Given a + b = c

WHERE a % 10 > c % 10
AND b % 10 > c % 10

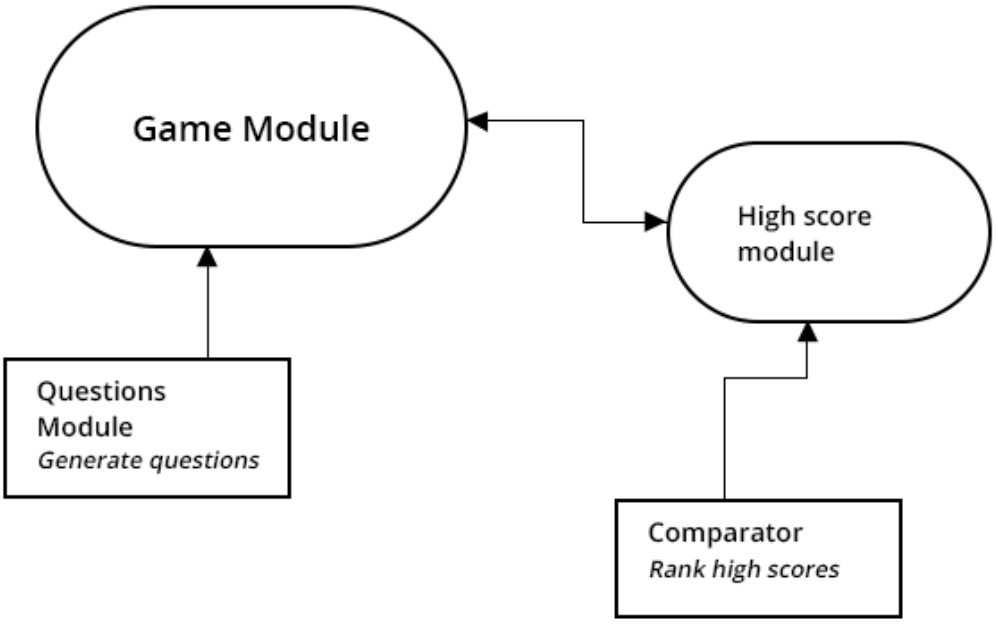
- ‘Reverse Tasks’ The Term ‘reverse tasks’ shall be understood as follows:
  c – b = a is understood as reverse task of a + b = c
  c – a = b is understood as reverse task of a + b = c
  c / b = a is understood as reverse task of a * b = c
  c / a = b is understood as reverse task of a * b = c

2.3 System Environment

The etaMath application is a text console-based command-line application written in Java.

3. Architecture

The system comprises of modules to create questions for the game, a module to store High scores, a comparator to rank High scores, and a module to display High score.



3.1 Subsystem, Component, or Module 1 …N

The questions module generates all possible questions before the game begins.

The game module displays questions at random. Random numbers between 1 and 4 are generated to determine the question type to show

1 for Addition
2 for Multiplication
3 for Subtraction
4 for division

The High score module stores and displays the High scores.

The comparator handles ranking of high scores. The scores and first ranked based on number of correct questions answered. In the case of a tie on number of correct questions answered, the scores are ranked based on used time. In the case of a further tie on used time, the scores are ranked based on the player names.

4. Data Design

4.1 Data Description

Task model has 2 fields

- String question – stores the question
- Integer answer – store the answer to the question

Answer model has 2 fields

- String input – stores the answer entered in the console
- Long time – stores the time (in milliseconds) the answer was entered so we can compare it to when the question was asked.

long timeOfQuestion – stores the time a question was asked
Task currentTask – holds the current task being answered
List<Task> additionQuestions – holds addition questions
List<Task> subtractionQuestions – holds subtraction questions
List<Task> multiplicationQuestions - holds multiplication questions
List<Task> divisionQuestions - holds division questions
List<String> allQuestions - holds all questions in the game
int additionCounter – stores the current position in “List<Task> additionQuestions” when displaying questions from the list
int SubtractionCounter – stores the current position in “List<Task> subtractionQuestions” when displaying questions from the list
int multiplicationCounter – stores the current position in “List<Task> multiplicationQuestions” when displaying questions from the list

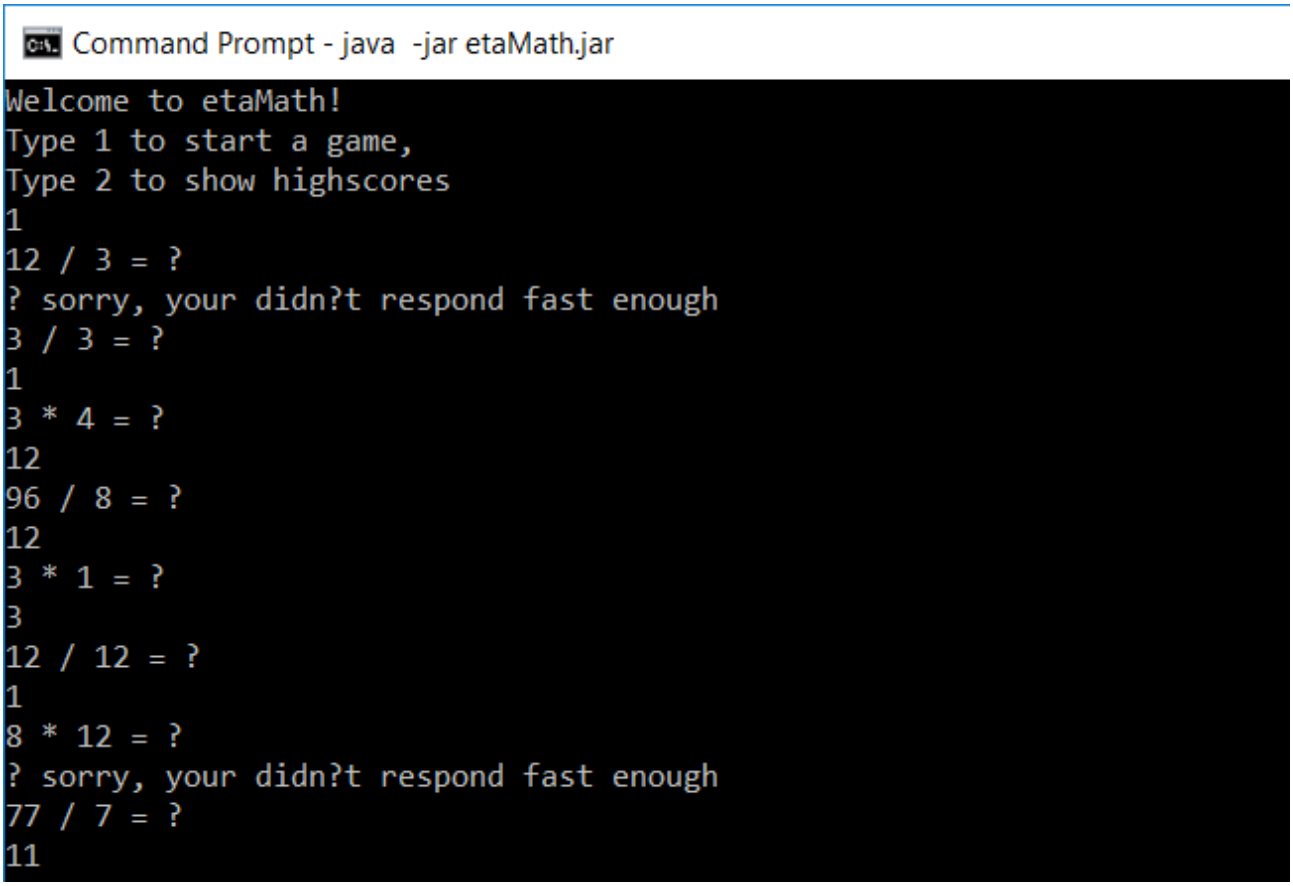
int divisionCounter – stores the current position in “List<Task> divisionQuestions” when displaying questions from the list
int currentQuestion – holds the value of the current question being answered in a game play
int score - holds the score of correctly answered questions
long totalTime – holds the total time spent answering questions
Queue<Answer> answers - stores the answer entered from the console
List<HighScore> highScores – stores the high scores
HighScoresComparator comparator – Ranks the high scores

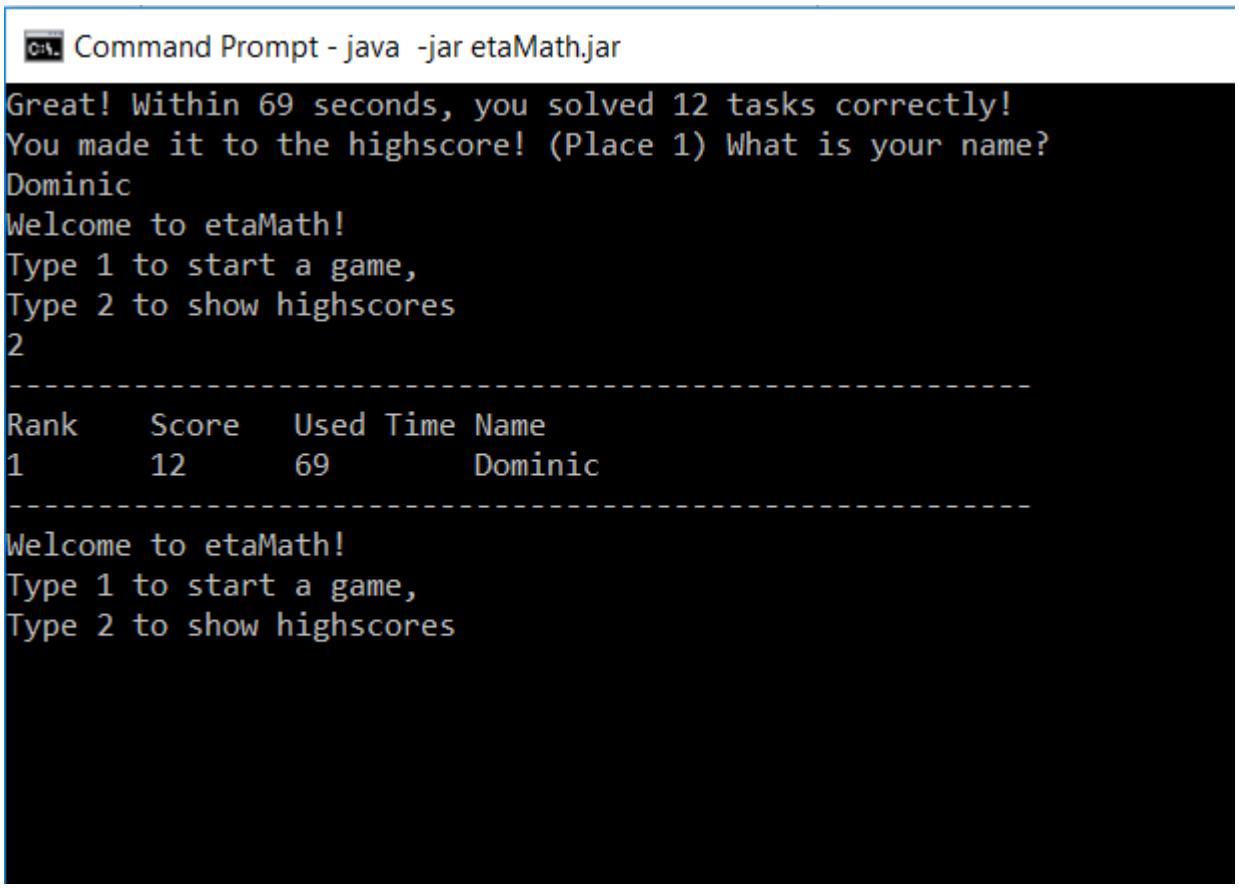
5. Interface

5.1 Overview of User Interface

The application is run from the command-line.

5.2 Screen Images




Command Prompt - java -jar etaMath.jar
Great! Within 69 seconds, you solved 12 tasks correctly!
You made it to the highscore! (Place 1) What is your name?
Dominic
Welcome to etaMath!
Type 1 to start a game,
Type 2 to show highscores
2
Rank Score Used Time Name
1 12 69 Dominic
Welcome to etaMath!
Type 1 to start a game,
Type 2 to show highscores