Fernando Gonzalez

Professor Kanemoto

CPSC-39-12112

6 December 2024

Final Project Report

Hello, my name is Fernando, and I decided to make a game called the Simple Math Test.

Overview of The Game:

The Simple Math Test is an interactive, time-based math game designed to challenge players' arithmetic skills. The player is asked 20 random math questions, and they have 1 minute to answer as many questions as possible. The game offers three difficulty levels: easy, medium, and hard. Each difficulty level increases the complexity of the questions by incorporating additional operations and larger numbers. This game is both entertaining and educational, helping users improve their mental math skills under time pressure.

Algorithms Used:

Generate Random Arithmetic Question

This algorithm creates a random math question based on the selected difficulty level.

```java
public static String generateQuestion(String difficulty, Random random) {
    int num1, num2;
    char operator = '+';

    switch (difficulty) {
        case "easy":
            num1 = random.nextInt(bound:15) + 1;
            num2 = random.nextInt(bound:15) + 1;
            operator = '+';
            break;

        case "medium":
            num1 = random.nextInt(bound:50) + 1;
            num2 = random.nextInt(bound:50) + 1;
            operator = random.nextBoolean() ? '+' : '-';
            break;

        case "hard":
            num1 = random.nextInt(bound:50) + 1;
            num2 = operator == '*' ? random.nextInt(bound:12) + 1 : random.nextInt(bound:50) + 1;
            operator = switch (random.nextInt(bound:3)) {
                case 0 -> '+';
                case 1 -> '-';
                default -> '*';
            };
            break;

        default:
            throw new IllegalStateException("Unexpected value: " + difficulty);
    }

    return num1 + " " + operator + " " + num2;
}
```

This function generates two random numbers and a random operator (+, -, or *) based on the selected difficulty.

- **Easy:** Generates numbers between 1 and 15 with only addition (+).

- **Medium:** Generates numbers between 1 and 50 with addition (+) or subtraction (-).
- **Hard:** Generates numbers up to 50 and randomly chooses between addition (+), subtraction (-), or multiplication (*).

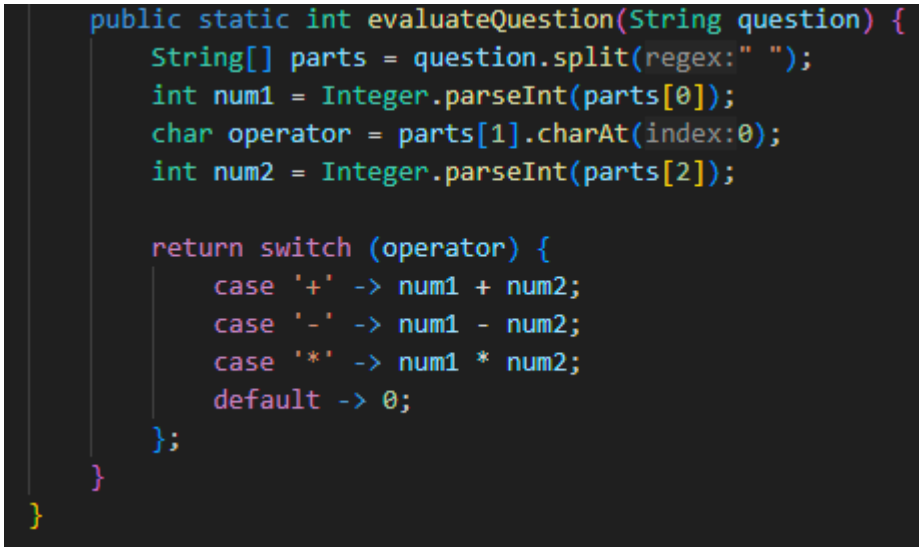
Big O Time Complexity:

- **Time Complexity:** *O(1)O(1)*O(1) – The function generates numbers and selects operators in constant time.
- **Space Complexity:** *O(1)O(1)*O(1) – Only a few variables are created, regardless of input size.

Evaluate Math Question

This algorithm evaluates the generated math question to determine the

correct answer.



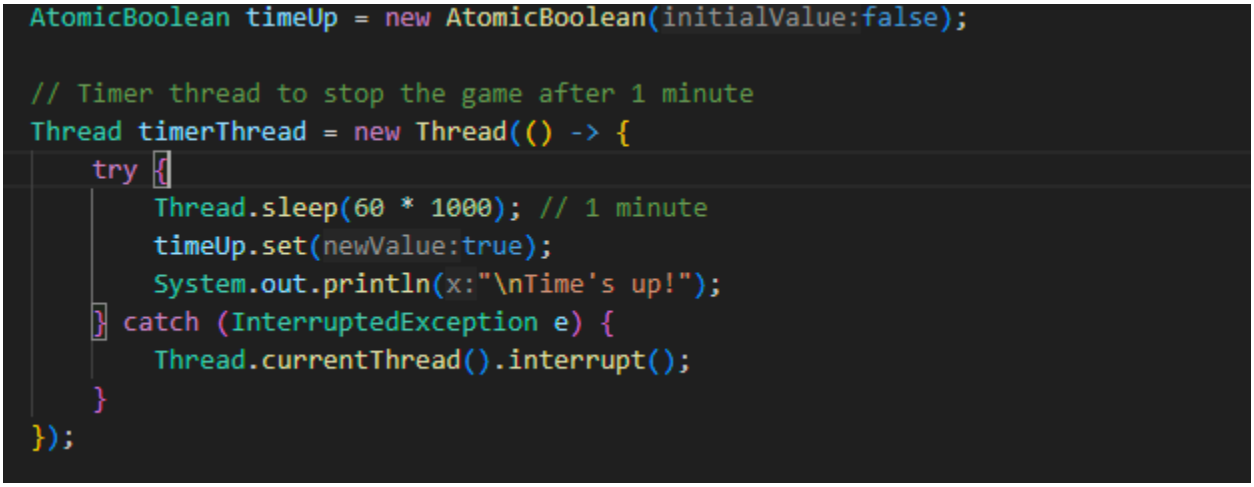
The function splits the question string (e.g., `"12 + 5"`) into its components: two numbers and an operator. It then evaluates the expression based on the operator.

Big O Time Complexity:

- **Time Complexity:** *O(1)O(1)*O(1) – Parsing the string and performing a basic arithmetic operation takes constant time.
- **Space Complexity:** *O(1)O(1)*O(1) – A fixed number of variables are used.

Timer Using a Separate Thread

This algorithm uses a separate thread to implement a 1-minute timer.



```
AtomicBoolean timeUp = new AtomicBoolean(initialValue:false);

// Timer thread to stop the game after 1 minute
Thread timerThread = new Thread(() -> {
    try {
        Thread.sleep(60 * 1000); // 1 minute
        timeUp.set(newValue:true);
        System.out.println(x:"\nTime's up!");
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
});
```

This function uses a Thread to create a timer that runs independently of the main game loop. After 60 seconds, the timeUp flag is set to true, and the game ends.

Big O Time Complexity:

- **Time Complexity:** *O(1)O(1)*O(1) – The timer sleeps for a fixed duration (60 seconds).
- **Space Complexity:** *O(1)O(1)*O(1) – Minimal memory is used for the timer thread and the AtomicBoolean.

Data Structures Used

Arrays

Usage:

In the evaluateQuestion method, the question string is split into parts using a String array to extract the numbers and operator.

Why Chosen:

Arrays provide a simple and efficient way to store fixed-length collections of elements. Splitting the string into a String array is a straightforward way to parse the question.

AtomicBoolean

Usage:

Used to track if the timer has expired (`timeUp` flag).

Why Chosen:

`AtomicBoolean` is thread-safe, making it suitable for scenarios where multiple threads need to read/write to the same variable.

Thread

Usage:

A `Thread` is used to run the timer independently of the main game loop.

Why Chosen:

Threads allow concurrent execution, enabling the timer to count down while the game loop continues to run.

OVERALL EXPLANATION SUMMARY REPORT

This summary is more of a personally report on how I made this game. So First of all, I don't really pay attention to exactly what data structures I am going to be using or algorithms I am going to make. As most Programmers work I believe, I think we all start with the idea first then use whatever we need to get the idea to become a reality. I wanted to do something some what unique that no other student

would do or think of. I didn't want to copy something. Then it clicked into my head when I remembered in elementary school that they had this math game that used to test you. Thats where this inspiration came from. I wanted to make a simple math test for the average adult that would show them a score on what their math skill are at. Firstly, I wanted to implement a difficulty selector that varies the questions. I had this idea where I wanted easy to be only addition and the two numbers would only be from 1-15. Then medium would have the addition of subraction while also keeping addition. This time the number would vary from 1-50. Hard is the same as medium except it adds multiplication as the final sign. I also wanted to be timed as well to make it challenging for the player making the player want to engage more and more to earn a better score and beat their previous score. There were a couple of issues that I had run into when trying to make this game. One of them was the time. The timer would not end the game when 1 minute was up and would only end after what ever question you were on was answered. To fix this I added separate function where when the time is up, it will tell and then what ever the next input is won't count for that question and your score will be shown. There are a lot things I think I can improve for this game. I think it has potential. One is that I would like it to poke fun at the player if they scored bad and give them a message say something like you are a math loser if you score a 1-5. Also I would like it to praise the player if they got a good score saying something like you are a math god or you are a math wizard.