Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Spring, 2023), B.Sc. in CSE (Day)

LAB REPORT NO: 10
Course Title: Object Oriented Programing Lab
Course Code: CSE 202 Section: DE

Student Details

Name	ID
Md. Moshiur Rahman	221902324

Submission Date : 31/05/2023
Course Teacher's Name : **Dr. Muhammad Aminur Rahaman**

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

1. TITLE OF THE LAB EXPERIMENT:

Search for a number N inside a randomly generated array of size 200 using 5 threads.

2. OBJECTIVES

- ✓ Understand and implement java thread
- ✓ Apply thread in different use cases

3. ALGORITHM

Step-1 : Start

Step-2 : Define the constant values:

ARRAY_SIZE = 200 (size of the randomly generated array)

THREAD_COUNT = 5 (number of threads)

Step-3 : Define the main method:

Generate a random array of size ARRAY_SIZE using the generateRandomArray method

Generate a random target number using the generateRandomNumber method

Print the generated array and the target number

Create an ExecutorService with a fixed thread pool size of THREAD_COUNT

Divide the array into equal segments for each thread to search:

Calculate the start and end index for each segment

Submit each segment to the ExecutorService as a separate task:

In each task, call the searchNumber method to search for the target number within the assigned segment of the array

If a match is found, print the number and the corresponding index

Shut down the ExecutorService and wait for the threads to finish their execution

Step-4 : Define the generateRandomArray method:

Create an empty integer array of size ARRAY_SIZE

Generate random numbers between 0 and 999 and assign them to the array elements.

Return the generated array

Step-5 : Define the generateRandomNumber method:

Generate a random number between 0 and 999

Return the generated number

Step-6: Define the searchNumber method:

Accept the array, target number, start index, and end index as parameters

Iterate over the assigned segment of the array from the start index to the end index

If a number in the segment matches the target number, print the number and its index

Step-7: END

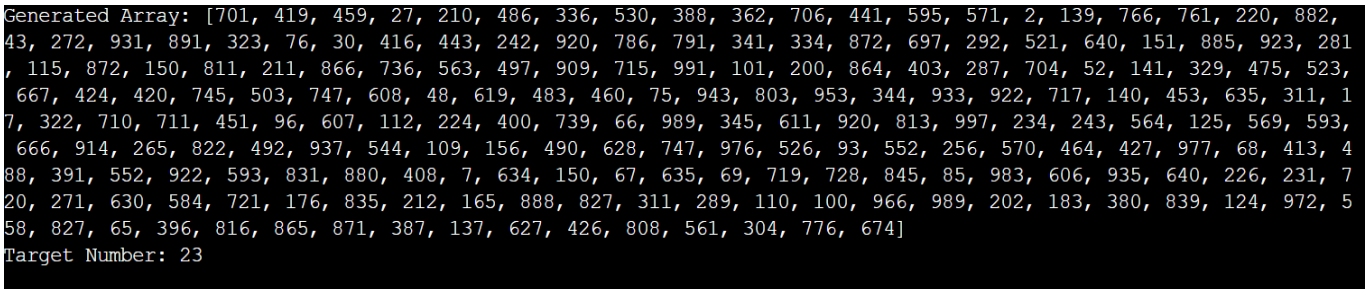
4. IMPLEMENTATION

```
import java.util.Arrays;
import java.util.Random;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class NumberSearch {

    private static final int ARRAY_SIZE = 200;
    private static final int THREAD_COUNT = 5;

    public static void main(String[] args) {
        int[] array = generateRandomArray(ARRAY_SIZE);
        int targetNumber = generateRandomNumber();
        System.out.println("Generated Array: " + Arrays.toString(array));
        System.out.println("Target Number: " + targetNumber);

        ExecutorService executor = Executors.newFixedThreadPool(THREAD_COUNT);
        int startIndex = 0;
        int endIndex = ARRAY_SIZE / THREAD_COUNT;

        for (int i = 0; i < THREAD_COUNT; i++) {
            int finalStartIndex = startIndex;
            int finalEndIndex = endIndex;
            executor.submit(() -> searchNumber(array, targetNumber, finalStartIndex,
finalEndIndex));
```

```
            startIndex = endIndex;
            endIndex += ARRAY_SIZE / THREAD_COUNT;
        }

        executor.shutdown();
        try {
            executor.awaitTermination(1, TimeUnit.MINUTES);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    private static int[] generateRandomArray(int size) {
        int[] array = new int[size];
        Random random = new Random();
        for (int i = 0; i < size; i++) {
            array[i] = random.nextInt(1000); // Random numbers between 0 and 999
        }
        return array;
    }

    private static int generateRandomNumber() {
        Random random = new Random();
        return random.nextInt(1000); // Random number between 0 and 999
    }

    private static void searchNumber(int[] array, int target, int startIndex, int endIndex) {
```

```
        for (int i = startIndex; i < endIndex; i++) {
            if (array[i] == target) {
                System.out.println("Number " + target + " found at index " + i);
            }
        }
    }
}
```

5. TEST RESULT



6. ANALYSIS & DISCUSSION

The program combines multithreading, parallelism, and randomness to efficiently search for a given number within a large array. By distributing the search task among multiple threads, it maximizes the utilization of system resources and potentially improves the search performance.