UNIVERSITI UTARA MALAYSIA
SECOND SEMESTER SESSION A242
STIWK 3014 REAL TIME PROGRAMMING
(GROUP A)

Exercise 13 - Recursive Task

Lecturer: Dr. Ruzita binti Ahmad

Name: Chong Mun Kei

Matric No: 298767

Github Link: https://github.com/Chong0508/RealTime.git

Source code

```
package Exercise12_RecursiveTask;

import java.util.Random;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;

public class FindMin extends RecursiveTask<Integer> {
   private static final long serialVersionUID = 1L;

   private int[] numbers;
   private int startIndex;
   private int endIndex;

   public FindMin(int[] numbers, int startIndex, int endIndex) {
       this.numbers = numbers;
       this.startIndex = startIndex;
       this.endIndex = endIndex;
   }

   @Override
   protected Integer compute() {
       int sliceLength = (endIndex - startIndex) + 1;

       if(sliceLength > 2) {
           int mid = startIndex + (sliceLength/2) - 1;
           // Split task into two subtasks
           FindMin lowerHalf = new FindMin(numbers, startIndex,
mid);
           FindMin upperHalf = new FindMin(numbers, mid + 1,
endIndex);

           // Fork the first subtask
           lowerHalf.fork(); // enables concurrent

           // Compute second subtask directly to improve
efficiency
           int upperResult = upperHalf.compute();

           // Join first subtask
           int lowerResult = lowerHalf.join(); // wait for the
lowerhalf result return its result

           return Math.min(lowerResult, upperResult);
       } else {
           // Direct comparison when 2 or fewer elements
           if(startIndex == endIndex) {
               return numbers[startIndex];
           }
           return Math.min(numbers[startIndex],
numbers[endIndex]);
       }
   }

   public static void main(String[] args) {
       int[] numbers = new int[100];
       Random random = new Random(System.currentTimeMillis());

       // Fill the array with random numbers between 0 and 99
       for(int i = 0; i < numbers.length; i++) {
           numbers[i] = random.nextInt(100);
       }

       // Display array contents (optional)
       System.out.println("Array: ");
       for(int num : numbers){
           System.out.println(num + " ");
       }
       System.out.println();

       // Use ForkJoinPool with available processors
       ForkJoinPool pool = new
ForkJoinPool(Runtime.getRuntime().availableProcessors());

       Integer min = pool.invoke(new FindMin(numbers, 0,
numbers.length-1));

       System.out.println("Minimum value in the array: " + min);
   }
}
```

Output

```
Array:

57

28

92

4

91

0

69

17

80

83

58

57

49

47

79

75

44

77

96
```

72

65

79

38

90

10

27

71

0

71

71

36

49

30

13

83

73

25

70

25

70

64

22

54

94

58

16

72

11

32

61

23

80

37

54

94

34

74

91

3

11

17

46

1

10

46

82

67

69

46

90

10

13

80

25

84

20

38

79

23

85

81

46

```
92

85

9

20

13

29

3

70

87

13

50

34

82

12

92

45

3

97


Minimum value in the array: 0
```

```
Process finished with exit code 0
```