20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO: URK22AI1038

|  |  |
| --- | --- |
| **Ex. No. 8** | **MULTITHREADING** |
| **Date of Exercise** | 05-10-2023 |

# Aim :

To write a java program which takes input as huge array of numbers in which array is split into n sub-arrays and n threads apply a bubble sort on each of the n sub-arrays.

# Procedure :

* 1. Start the program.
  2. Create the requested classes.
  3. Split the n sub arrays and n threads.
  4. Apply bubble sort on each of n sub array.
  5. Create another sort on each of n sub array.
  6. Give the input and get the output.
  7. Stop the program.

# Program :

import java.util.concurrent.\*; import java.util.Arrays;

public class ParallelBubbleSort {

public static void main(String[] args) throws InterruptedException, ExecutionException { int[] inputArray = {9, 2, 5, 1, 7, 8, 3, 4, 6};

int n = 3;

int subArraySize = inputArray.length / n;

ExecutorService executor = Executors.newFixedThreadPool(n);

1

20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO:URK22AI1050

Future<int[]>[] futures = new Future[n]; for (int i = 0; i < n; i++) {

int startIndex = i \* subArraySize;

int endIndex = (i == n - 1) ? inputArray.length : (i + 1) \* subArraySize; int[] subArray = Arrays.copyOfRange(inputArray, startIndex, endIndex); Callable<int[]> task = new BubbleSortTask(subArray);

futures[i] = executor.submit(task);

}

for (Future<int[]> future : futures) { future.get();

}

int[] sortedArray = mergeSortedArrays(futures); System.out.println(Arrays.toString(sortedArray)); executor.shutdown();

}

static class BubbleSortTask implements Callable<int[]> { private int[] subArray;

public BubbleSortTask(int[] subArray) { this.subArray = subArray;

}

@Override

public int[] call() { bubbleSort(subArray); return subArray;

}

}

static void bubbleSort(int[] arr) { int n = arr.length;

boolean swapped;

for (int i = 0; i < n - 1; i++) { swapped = false;

for (int j = 0; j < n - i - 1; j++) { if (arr[j] > arr[j + 1]) {

int temp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = temp;

2

20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO:URK22AI1050

swapped = true;

}

}

if (!swapped) { break;

}

}

}

static int[] mergeSortedArrays(Future<int[]>[] futures) throws InterruptedException, ExecutionException {

int n = futures.length;

int[][] sortedArrays = new int[n][];

for (int i = 0; i < n; i++) { sortedArrays[i] = futures[i].get();

}

return mergeArrays(sortedArrays);

}

static int[] mergeArrays(int[][] arrays) { int totalLength = 0;

for (int[] array : arrays) { totalLength += array.length;

}

int[] result = new int[totalLength]; int[] indices = new int[arrays.length];

for (int i = 0; i < totalLength; i++) { int minIndex = -1;

int minValue = Integer.MAX\_VALUE;

for (int j = 0; j < arrays.length; j++) {

if (indices[j] < arrays[j].length && arrays[j][indices[j]] < minValue) { minValue = arrays[j][indices[j]];

minIndex = j;

3

20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO:URK22AI1050

}

}

result[i] = minValue; indices[minIndex]++;

}

return result;

}

}

# Output Screenshot :



**Result :**

The above program has been successfully executed and verified.

4

20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO:URK22AI1050

# Aim :

To write a java program to check with the Airport to see if it has an available runway before it's able to take off or land using multi-threading.

# Procedure :

* 1. Start the program.
  2. Create all the required classes.
  3. Implement a mechanism for airplanes to request access to the runway from the airport
  4. The airport should grant access if there are available runways; otherwise, airplanes should wait.
  5. After completing their operation (take off or landing), airplanes should release the runway.
  6. Stop the programe

# Program :

import java.util.concurrent.Semaphore; public class AirportSimulation {

private static final int NUM\_RUNWAYS = 3;

private static Semaphore runwaySemaphore = new Semaphore(NUM\_RUNWAYS);

public static void main(String[] args) {

Thread takeOffThread = new Thread(new Airplane("Take Off")); Thread landThread = new Thread(new Airplane("Land"));

takeOffThread.start(); landThread.start();

}

5

20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO:URK22AI1050

static class Airplane implements Runnable { private String operation;

public Airplane(String operation) { this.operation = operation;

}

@Override

public void run() { while (true) {

try {

System.out.println("Airplane requesting permission to " + operation); runwaySemaphore.acquire(); // Request permission to use a runway

System.out.println("Airplane is " + operation); Thread.sleep(2000); // Simulate the operation

System.out.println("Airplane has completed " + operation); runwaySemaphore.release(); // Release the runway

} catch (InterruptedException e) { e.printStackTrace();

}

}

}

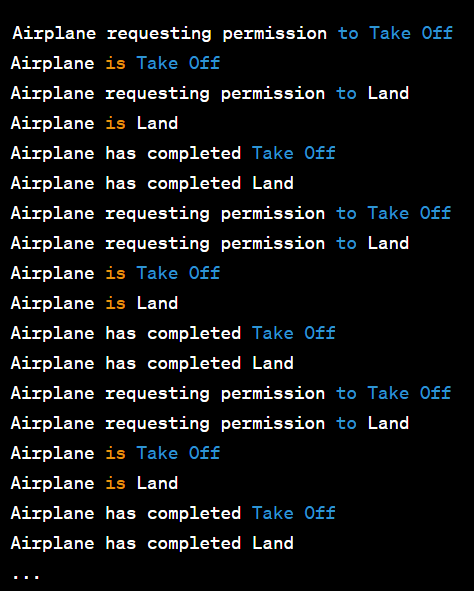
}

}

6

20CS2035L–Object Oriented Programming (Lab)–B11 REG.NO:URK22AI1050

# Output Screenshot :



**Result :**

The above program has been successfully executed and verified.

7