



3^η Εργασία

Διαχείριση Σύνθετων Δεδομένων
27/05/2023 – Εαρινό Εξάμηνο 2023

By

ΤΟΔΡΙ ΑΓΓΕΛΟ (ΑΜ. 3090)
Καθηγητής: Ν. Μαμουλής

ΠΕΡΙΕΧΟΜΕΝΑ

Top-k Queries– Λίγα λόγια σχετικά με αυτά.....	2
Σημείωση για τον Αξιολογητή.....	2
Γνωστά σφάλματα – Known bugs/issues.....	2
Επεξήγηση προγράμματος/σχολιασμός & Στιγμιότυπα οθόνης.....	3
Μεθόδοι προγράμματος.....	3
1. Μέθοδος main – Βασική μέθοδος εκτέλεσης προγράμματος.....	3
2. Μέθοδος getKfromInput – Ανάκτηση παραμέτρου k απο command line .	4
3. Μέθοδος readRandomScores – Ανάγνωση αρχείου rnd.txt.....	5
4. Μέθοδος bruteForceTopK – Χρήση brute force για υπολογισμό top scores.....	6
5. Μέθοδος printOutput – Εκτύπωση output στην κονσόλα.....	8
6. Μέθοδος computeTokK – Υπολογισμός top k αποτελεσμάτων	9
Κλάσεις προγράμματος.....	12
Κλάση Restaurant	12
Πραγματικά εκτελέσιμα της προγραμματιστικής άσκησης.....	13

Στόχος της εργασίας είναι η ανάπτυξη αλγορίθμου για ερωτήσεις κορυφαίων κ (top-k queries), οι οποίες συναθροίζουν ατομικά σκορ αντικειμένων από τρεις πηγές. Τα αντικείμενα μπορεί να είναι εστιατόρια, η κάθε πηγή ένας ιστότοπος κρίσεων εστιατορίων (π.χ. google, TripAdvisor, yelp), και το ατομικό σκορ ενός αντικειμένου σε μία πηγή ο μέσος όρος των βαθμολογιών (ratings) που πήρε το εστιατόριο στο συγκεκριμένο ιστότοπο.

Η υλοποίηση της προγραμματιστικής εργασίας έγινε με την γλώσσα προγραμματισμού **Java**.


TOP-K QUERIES– ΛΙΓΑ ΛΟΓΙΑ ΣΧΕΤΙΚΑ ΜΕ ΑΥΤΑ

Τα ερωτήματα top-k αναφέρονται σε μια τεχνική που χρησιμοποιείται στην ανάκτηση πληροφοριών και στην αναζήτηση δεδομένων. Αντιπροσωπεύουν τα πιο σημαντικά ή δημοφιλή ερωτήματα που υποβάλλονται από ένα σύνολο ερωτημάτων, με βάση ένα συγκεκριμένο κριτήριο σημαντικότητας ή δημοτικότητας.

Τα ερωτήματα top-k είναι χρήσιμα για διάφορους λόγους. Πρώτον, μπορούν να χρησιμοποιηθούν για τη βελτιστοποίηση της ανάκτησης πληροφοριών και της απόδοσης των συστημάτων αναζήτησης. Αναλύοντας τα Top-k ερωτήματα, μπορούμε να εντοπίσουμε τις πιο συχνές αναζητήσεις και να βελτιστοποιήσουμε τις αντίστοιχες απαντήσεις ώστε να παρέχουμε ταχύτερα και ακριβέστερα αποτελέσματα.

Συνοπτικά, τα ερωτήματα Top-k μας επιτρέπουν να ιεραρχούμε και να βελτιστοποιούμε τα αποτελέσματα αναζήτησης, να κατανοούμε τη συμπεριφορά των χρηστών και να παρέχουμε εξατομικευμένες και αποτελεσματικές εμπειρίες αναζήτησης.

ΣΗΜΕΙΩΣΗ ΓΙΑ ΤΟΝ ΑΞΙΟΛΟΓΗΤΗ

 **READ ME FISRT:** Για δικιά σας διευκόλυνση ο κώδικας της προγραμματιστικής εργασίας καλύπτεται πλήρως και εξηγείται αναλυτικά στις επόμενες σελίδες με τη χρήση στιγμιotypων οθόνης και σχολίων. **Πρακτικά μπορείτε εάν το επιθυμείτε να παραλείψετε την αξιολόγηση του αρχείου .java,** καθώς τα στιγμιότυπα οθόνης και τα επεξηγηματικά σχόλια καλύπτουν πλήρως τον κώδικα. Αυτό το PDF δημιουργήθηκε με μεγάλη προσοχή και προσπάθεια για να δοθεί έμφαση στις λεπτομέρειες. Για του λόγου το αληθές μπορείτε να το εξακριβώσετε αυτό και οι ίδιοι. Καλή ανάγνωση/αξιολόγηση!

ΓΝΩΣΤΑ ΣΦΑΛΜΑΤΑ: KNOWN BUGS

Κάτα την computeK διατρέχουμε συνολικά όλες τις εγγραφές στα 2 αρχεία seq1.txt και seq2.txt. **Δεν τα αποθηκεύουμε στην μνήμη όμως (ευτυχώς).** Η έξυπνη σάρωση δεν γίνεται με τον βέλτιστο τρόπο. Ενδεικτικό output για k = 5:

Number of sequential accesses = **200000**

Top k objects:

50905: 14.84

85861: 14.76

22652: 14.74

75232: 14.74

20132: 14.74

ΕΠΕΞΗΓΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ/ ΣΧΟΛΙΑΣΜΟΣ & ΣΤΙΓΜΙΟΤΥΠΑ ΟΘΟΝΗΣ

ΜΕΘΟΔΟΙ ΠΡΟΓΡΑΜΜΑΤΟΣ

ΜΕΘΟΔΟΣ MAIN - ΒΑΣΙΚΗ ΜΕΘΟΔΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
public static void main(String[] args) {  
    // -----> 1 // You can either get k from user input or assign a default value  
    //int k = getKFromInput(args);  
  
    // Set the value of k (number of top scores to compute)  
    int k = 5;  
  
    try {  
        // Read random scores from a file into a map  
        Map<Integer, Double> randomScores = readRandomScores(RANDOM_FILE_TXT_FILE_LOCATION);  
  
        // Compute the top k scores using the random scores map and specified file locations  
        computeTopK(randomScores, SEQ1_TXT_FILE_LOCATION, SEQ2_TXT_FILE_LOCATION, k);  
  
        // -----> 2 Alternatively, you can compute the top k scores using brute force approach  
        // bruteForceTopK(randomScores, SEQ1_TXT_FILE_LOCATION, SEQ2_TXT_FILE_LOCATION, k);  
    } catch (IOException e) {  
        System.out.println("An error occurred while reading the input files.");  
        e.printStackTrace();  
    }  
}
```

Εικόνα 1- Μέθοδος main

Σχολιασμός:

- Ορισμός της τιμής του k, η οποία αντιπροσωπεύει τον αριθμό των κορυφαίων βαθμολογιών που θα υπολογιστούν. Το k έχει την προεπιλεγμένη τιμή 5. Ωστόσο, μπορείτε να ξεσχολιάσετε την παραπάνω γραμμή (int k = getKFromInput(args)) και να υλοποιήσετε τη μέθοδο getKFromInput για να λάβετε το k από την είσοδο του χρήστη.
- Μέσα σε ένα block try-catch, ο κώδικας διαβάζει τυχαία σκορ από ένα αρχείο σε έναν χάρτη χρησιμοποιώντας τη μέθοδο readRandomScores. Το RANDOM_FILE_TXT_FILE_LOCATION είναι η θέση του αρχείου που περιέχει τα τυχαία σκορ.
- Στη συνέχεια, ο κώδικας καλεί τη μέθοδο computeTopK, περνώντας το map randomScores, τα SEQ1_TXT_FILE_LOCATION, SEQ2_TXT_FILE_LOCATION και k ως παραμέτρους. Αυτή η μέθοδος υπολογίζει τις κορυφαίες k βαθμολογίες με βάση τις τυχαίες βαθμολογίες και τις βαθμολογίες από τα seq1.txt και seq2.txt.
- Ο σχολιασμένος κώδικας περιλαμβάνει μια εναλλακτική προσέγγιση για τον υπολογισμό των κορυφαίων k βαθμολογιών, χρησιμοποιώντας τη μέθοδο bruteForceTopK. Μπορείτε να ξεσχολιάσετε αυτή τη γραμμή αν θέλετε να χρησιμοποιήσετε την προσέγγιση ωμής βίας για τον υπολογισμό των κορυφαίων k βαθμολογιών. **Υλοποιήθηκε για σκοπούς επαλήθευσης.**

ΜΕΘΟΔΟΣ GET K FROM INPUT – ΑΝΑΚΤΗΣΗ ΠΑΡΑΜΕΤΡΟΥ ΑΠΟ COMMAND LINE

```
no usages
private static int getKFromInput(String[] args) {
    // Check if the number of command-line arguments is less than 1
    if (args.length < 1) {
        System.out.println("Please provide a positive integer k as an argument.");
    }

    // Parse the first argument (k) into an integer and return it
    return Integer.parseInt(args[0]);
}
```

Εικόνα 2 - Μέθοδος getKFromInput

Σχολιασμός:

- Ελέγχεται αν ο αριθμός των παραμέτρων της γραμμής εντολών (args.length) είναι μικρότερος από 1, υποδεικνύοντας ότι δεν δόθηκε κανένα όρισμα.
- Εάν δεν έχουν δοθεί ορίσματα, εμφανίζει ένα μήνυμα στην κονσόλα, ζητώντας από τον χρήστη να δώσει έναν θετικό ακέραιο αριθμό k.
- Στη συνέχεια, η μέθοδος αναλύει το πρώτο όρισμα (args[0]) σε ακέραιο αριθμό χρησιμοποιώντας τη μέθοδο Integer.parseInt().
- Η μέθοδος επιστρέφει την αναλυμένη ακέραια τιμή του k.

ΜΕΘΟΔΟΣ READ RANDOM SCORES – ΑΝΑΓΝΩΣΗ ΑΡΧΕΙΟΥ RND.TXT

```
1 usage
private static Map<Integer, Double> readRandomScores(String filename) throws IOException {
    // Create a new HashMap to store the scores
    Map<Integer, Double> scores = new HashMap<>();

    // Use a try-with-resources block to automatically close the reader
    try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
        String line;
        // Read each line from the file
        while ((line = reader.readLine()) != null) {
            // Split the line by a space to separate the ID and score
            String[] parts = line.split(" ");
            // Parse the ID as an integer
            int id = Integer.parseInt(parts[0]);
            // Parse the score as a double
            double score = Double.parseDouble(parts[1]);
            // Add the ID and score to the map
            scores.put(id, score);
        }
    }

    // Return the map of scores
    return scores;
}
```

Εικόνα 3 - Μέθοδος readRandomScores

Σχολιασμός:

- Δημιουργεί ένα νέο HashMap με όνομα scores για την αποθήκευση των αποτελεσμάτων.
- Αξιοποίηση BufferedReader με χρήση try-with-resources για το καθορισμένο αρχείο.
- Διάβασμα κάθε γραμμής από το αρχείο χρησιμοποιώντας τη μέθοδο readLine του BufferedReader. Η επανάληψη συνεχίζεται μέχρι να μην υπάρχουν άλλες γραμμές προς ανάγνωση.
- Διαχωρισμός κάθε γραμμής με ένα κενό χρησιμοποιώντας τη μέθοδο split. Το πρώτο μέρος αντιπροσωπεύει το ID και το δεύτερο μέρος αντιπροσωπεύει τη βαθμολογία του εστιατορίου.
- Προσθήκη του ID και του σκορ στο map scores με χρήση της μεθόδου put.
- Μόλις ολοκληρωθεί η επεξεργασία όλων των γραμμών, η μέθοδος κλείνει τον bufferedReader και επιστρέφει το map με τα αποτελέσματα.

ΜΕΘΟΔΟΣ BRUTE FORCE TOP K – ΧΡΗΣΗ BRUTE FORCE ΓΙΑ ΥΠΟΛΟΓΙΣΜΟ TOP SCORES

```
private static void bruteForceTopK(Map<Integer, Double> randomScores, String seq1File, String seq2File, int k) throws IOException {
    // Create a new map to store the total scores
    Map<Integer, Double> totalScores = new HashMap<>(randomScores);

    try (BufferedReader seq1Reader = new BufferedReader(new FileReader(seq1File));
        BufferedReader seq2Reader = new BufferedReader(new FileReader(seq2File))) {

        // Read seq1.txt
        String seq1Line;
        while ((seq1Line = seq1Reader.readLine()) != null) {
            String[] parts = seq1Line.split(" ");
            int id = Integer.parseInt(parts[0]);
            double seq1Score = Double.parseDouble(parts[1]);

            if (totalScores.containsKey(id)) {
                double totalScore = seq1Score + totalScores.get(id);
                totalScores.put(id, totalScore);
            }
        }

        // Read seq2.txt
        String seq2Line;
        while ((seq2Line = seq2Reader.readLine()) != null) {
            String[] parts = seq2Line.split(" ");
            int id = Integer.parseInt(parts[0]);
            double seq2Score = Double.parseDouble(parts[1]);

            if (totalScores.containsKey(id)) {
                double totalScore = seq2Score + totalScores.get(id);
                totalScores.put(id, totalScore);
            }
        }
    }

    // Sort the restaurants based on total score in descending order
    List<Map.Entry<Integer, Double>> sortedEntries = new ArrayList<>(totalScores.entrySet());
    sortedEntries.sort(Map.Entry.comparingByValue(Comparator.reverseOrder()));

    // Print the top k restaurants
    for (int i = 0; i < Math.min(k, sortedEntries.size()); i++) {
        Map.Entry<Integer, Double> entry = sortedEntries.get(i);
        int id = entry.getKey();
        double totalScore = entry.getValue();

        // Format the total score to 2 decimal places
        DecimalFormatSymbols symbols = new DecimalFormatSymbols(Locale.getDefault());
        symbols.setDecimalSeparator('.'); // Set decimal separator to dot (.)
        DecimalFormat decimalFormat = new DecimalFormat("#.00", symbols); // Format to 2 decimal places with dot separator

        System.out.println(id + ": " + decimalFormat.format(totalScore));
    }
}
```

Εικόνα 4,5 - Μέθοδος *bruteForceTopK*

Σχολιασμός:

- Στον παραπάνω κώδικα, έχουμε μια μέθοδο με όνομα *bruteForceTopK* που λαμβάνει ως παραμέτρους ένα map των random scores, δύο διαδρομές αρχείων (*seq1File* και *seq2File*) και έναν ακέραιο *k*. Εκτελεί τα ακόλουθα βήματα:

- Δημιουργεί ένα νέο map με όνομα totalScores χρησιμοποιώντας/αντιγράφοντας το όρισμα της μεθόδου με όνομα randomScores.
- Χρησιμοποιεί try-with-resources για να ανοίξει και να κλείσει αυτόματα τα BufferedReader για το διάβασμα των seq1File και seq2File.
- Διάβασμα κάθε γραμμής από το seq1File και το seq2File αντίστοιχα και ενημέρωση των αντιστοιχών συνολικών σκορ στο map totalScores.
- Ταξινόμηση των καταχωρήσεων του map totalScores με βάση τη συνολική βαθμολογία σε φθίνουσα σειρά.
- Εκτύπωση των κορυφαίων k καταχωρήσεων από τον ταξινομημένο κατάλογο, εμφανίζοντας το αναγνωριστικό του εστιατορίου και τη μορφοποιημένη συνολική βαθμολογία.

ΜΕΘΟΔΟΣ PRINT OUTPUT – ΕΚΤΥΠΩΣΗ OUTPUT ΣΤΗΝ ΚΟΝΣΟΛΑ

```
private static void printOutput(PriorityQueue<Restaurant> topK, int seqAccessCount) {
    // Print the number of sequential accesses
    System.out.println("Number of sequential accesses = " + seqAccessCount);

    // Print the top k objects
    System.out.println("Top k objects:");

    // Convert the PriorityQueue to a List
    List<Restaurant> topKList = new ArrayList<>();
    while (!topK.isEmpty()) {
        Restaurant restaurant = topK.poll();
        topKList.add(restaurant);
    }

    // Reverse the List to get descending order
    Collections.reverse(topKList);

    // Format the scores to 2 decimal places
    DecimalFormatSymbols symbols = new DecimalFormatSymbols(Locale.getDefault());
    symbols.setDecimalSeparator('.'); // Set decimal separator to dot (.)
    DecimalFormat decimalFormat = new DecimalFormat("#.00", symbols); // Format to 2 decimal places with dot separator

    // Print the ID and formatted score of each restaurant
    for (Restaurant restaurant : topKList) {
        String formattedScore = decimalFormat.format(restaurant.getTotalScore());
        System.out.println(restaurant.getId() + ": " + formattedScore);
    }
}
```

Εικόνα 4 - Μέθοδος printOutput

Σχολιασμός:

- Εκτύπωση του αριθμού των διαδοχικών προσπελάσεων.
- Εκτύπωση της επικεφαλίδας για τα κορυφαία k αντικείμενα.
- Μετατροπή της PriorityQueue topK σε μια λίστα με το όνομα topKList.
- Αντιστροφή της σειράς των στοιχείων στην topKList για να εκτυπωθούν με φθίνουσα σειρά.
- Ορισμός εντός DecimalFormat αντικειμένου για την μορφοποίηση των αποτελεσμάτων των βαθμολογιών των εστιατορίων με 2 δεκαδικά ψηφία.
- Εκτύπωση των IDs και των μορφοποιημένων βαθμολογιών.

ΜΕΘΟΔΟΣ COMPUTE TOP K – ΥΠΟΛΟΓΙΣΜΟΣ ΤΩΠ K RESULTS

```
private static void computeTopK(Map<Integer, Double> randomScores, String seq1File, String seq2File, int k) throws IOException {
    // Create a PriorityQueue to store the top k restaurants based on total score
    PriorityQueue<Restaurant> topK = new PriorityQueue<>(Comparator.comparingDouble(Restaurant::getTotalScore));

    // Variables for tracking scores and access count
    double lastSeq1Score = 0.0;
    double lastSeq2Score = 0.0;
    int seqAccessCount = 0;

    // Initialize a map to store the lower bounds and seen status of each item
    Map<Integer, Restaurant> lowerBounds = new HashMap<>();

    try (BufferedReader seq1Reader = new BufferedReader(new FileReader(seq1File));
        BufferedReader seq2Reader = new BufferedReader(new FileReader(seq2File))) {

        String seq1Line;
        String seq2Line;
        boolean seq1Finished = false;
        boolean seq2Finished = false;

        while (!seq1Finished || !seq2Finished) {

            // Read from seq1.txt if it's not finished
            if (!seq1Finished) {
                seq1Line = seq1Reader.readLine();
                if (seq1Line != null) {
                    // Process seq1Line
                    String[] parts = seq1Line.split(" ");
                    int id = Integer.parseInt(parts[0]);
                    double seq1Score = Double.parseDouble(parts[1]);

                    if (!randomScores.containsKey(id))
                        continue;

                    // Update lower bounds and add the item to topK
                    double lowerBound;
                    if (!lowerBounds.containsKey(id)) {
                        lowerBound = seq1Score + randomScores.get(id);
                        Restaurant restaurant = new Restaurant(id, lowerBound);
                        lowerBounds.put(id, restaurant);
                        topK.offer(restaurant); // Add the item to topK
                    } else {
                        Restaurant restaurant = lowerBounds.get(id);
                        lowerBound = restaurant.getLowerBound() + seq1Score;
                        restaurant.setTotalScore(restaurant.getTotalScore() + seq1Score);
                        restaurant.setLowerBound(lowerBound);
                    }

                    lastSeq1Score = seq1Score;

                    // Increment seqAccessCount only when reading from seq1.txt
                    seqAccessCount++;
                } else {
                    seq1Finished = true;
                }
            }
        }
    }
```

```

// Read from seq2.txt if it's not finished
if (!seq2Finished) {
    seq2Line = seq2Reader.readLine();
    if (seq2Line != null) {
        // Process seq2Line
        String[] parts = seq2Line.split(" ");
        int id = Integer.parseInt(parts[0]);
        double seq2Score = Double.parseDouble(parts[1]);

        if (!randomScores.containsKey(id))
            continue;

        double lowerBound;
        if (!lowerBounds.containsKey(id)) {
            // Update lower bounds and add the item to topK
            lowerBound = seq2Score + randomScores.get(id);
            Restaurant restaurant = new Restaurant(id, lowerBound);
            lowerBounds.put(id, restaurant);
            topK.offer(restaurant); // Add the item to topK
        } else {
            // Update scores and access count
            Restaurant restaurant = lowerBounds.get(id);
            lowerBound = restaurant.getLowerBound() + seq2Score;
            restaurant.setTotalScore(restaurant.getTotalScore() + seq2Score);
            restaurant.setLowerBound(lowerBound);
        }

        lastSeq2Score = seq2Score;

        // Increment seqAccessCount only when reading from seq2.txt
        seqAccessCount++;
    } else {
        seq2Finished = true;
    }
}

// Termination condition
double threshold = lastSeq1Score + lastSeq2Score + 5.0;
while (!topK.isEmpty() && topK.peek().getTotalScore() < threshold) {
    Restaurant restaurant = topK.poll();
    // Adjust topK based on the threshold and upper/lower bounds
    if (!topK.isEmpty() && restaurant.getUpperBound() >= topK.peek().getLowerBound()) {
        topK.offer(restaurant);
        break;
    }
}

// Remove excess elements from topK if necessary
if (topK.size() > k)
    topK.poll();
}

// Call the printOutput method to display the top k restaurants and the number of sequential accesses
printOutput(topK, seqAccessCount);
}

```

Εικόνα 5,8,9,10 - 4 εικόνες που συνδυάζουν την μέθοδο *computeTopK*

Σχολιασμός:

- Σε αυτόν τον κώδικα, έχουμε μια μέθοδο με όνομα `computeTopK` που δέχεται ως παραμέτρους ένα `map` με τα `randomScores`, δύο διαδρομές αρχείων `seq1File` και `seq2File` και έναν ακέραιο `k`. Εκτελεί τα ακόλουθα βήματα:
- Δημιουργεί μια `PriorityQueue` με όνομα `topK` για την αποθήκευση των κορυφαίων `k` εστιατορίων με βάση τη συνολική βαθμολογία τους.
- Αρχικοποιεί τις μεταβλητές για την παρακολούθηση των βαθμολογιών και τον αριθμό των προσβάσεων.
- Αρχικοποιεί ένα `map` με όνομα `lowerBounds` για την αποθήκευση των κατώτερων ορίων και της κατάστασης θέασης (εάν έχουν σαρωθεί ήδη κατά το σκανάρισμα) κάθε στοιχείου.
- Διαβάζει τα αρχεία εισόδου `seq1File` και `seq2File` χρησιμοποιώντας `BufferedReader`.
- Επεξεργάζεται τις γραμμές από τα αρχεία εισόδου και ενημερώνει τα κατώτερα όρια των εστιατορίων, τα προσθέτει στο `topK` και ενημερώνει τα σκορ και τον αριθμό προσβάσεων.
- Εφαρμόζει μια συνθήκη τερματισμού με βάση ένα κατώφλι (`threshold`) και προσαρμόζει το `topK` με βάση το κατώφλι (`threshold`) και τα άνω/κάτω όρια.
- Αφαιρεί τα πλεονάζοντα στοιχεία από το `topK` εάν το μέγεθος υπερβαίνει το `k`.
- Καλεί τη μέθοδο `printOutput` για να εμφανίσει τα κορυφαία `k` εστιατόρια και τον αριθμό των διαδοχικών προσβάσεων.

Σκοπός αυτής της μεθόδου είναι να υπολογίσει τα κορυφαία `k` εστιατόρια με βάση τα παρεχόμενα αποτελέσματα και την είσοδο

ΚΛΑΣΕΙΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

ΚΛΑΣΗ RESTAURANT

```
private static class Restaurant {  
    2 usages  
    private final int id;  
    4 usages  
    private double lowerBound;  
    4 usages  
    private double totalScore;  
  
    2 usages  
    public Restaurant(int id, double lowerBound) {...}  
  
    public int getId() { return id; }  
  
    3 usages  
    public double getLowerBound() { return lowerBound; }  
  
    2 usages  
    public void setLowerBound(double lowerBound) { this.lowerBound = lowerBound; }  
  
    5 usages  
    public double getTotalScore() { return totalScore; }  
  
    2 usages  
    public void setTotalScore(double totalScore) { this.totalScore = totalScore; }  
  
    1 usage  
    public double getUpperBound() { return lowerBound + totalScore; }  
}
```

Εικόνα 11 - Μοναδική κλάση προγράμματος Restaurant

Κλάση/μοντέλο (model) που κουβαλάει την πληροφορία για τα αναγνωριστικά και τα στοιχεία αναφορικά με τις βαθμολογίες των εστιατορίων.

ΠΡΑΓΜΑΤΙΚΑ ΕΚΤΕΛΕΣΙΜΑ ΤΗΣ ΑΣΚΗΣΗΣ

Number of sequential accesses = 200000

Top k objects:

50905: 14.84

$k=1$

Number of sequential accesses = 200000

Top k objects:

50905: 14.84

85861: 14.76

22652: 14.74

75232: 14.74

20132: 14.74

$k=5$

Number of sequential accesses = 200000

Top k objects:

50905: 14.84

85861: 14.76

22652: 14.74

75232: 14.74

20132: 14.74

21824: 14.70

9041: 14.66

97866: 14.65

83759: 14.64

96407: 14.58

35055: 14.57

78315: 14.54

594: 14.54

16564: 14.54

33288: 14.52

79330: 14.51

11283: 14.49

4885: 14.47

9745: 14.45

55165: 14.44

$k=20$

**ΣΑΣ ΕΥΧΑΡΙΣΤΩ ΘΕΡΜΑ ΓΙΑ ΤΟΝ ΧΡΟΝΟ ΚΑΙ ΤΗΝ ΠΡΟΣΟΧΗ
ΣΑΣ!!!**