Σκοπός της εργασίας αυτής είναι η εξοικείωση σας με την υλοποίηση απλών δομών δεδομένων και αλγορίθμων που χειρίζονται αυτές τις δομές. Στην συγκεκριμένη εργασία καλείστε να υλοποιήσετε έναν πίνακα κατακερματισμού με χρήση ανοικτής διευθυνσιοδότησης (open addressing) και καθολικού κατακερματισμού (universal hashing) σε γλώσσα Java.

Ανοικτή διευθυνσιοδότηση Υλοποιήστε έναν πίνακα κατακερματισμού με την χρήση της τεχνικής ”ανοικτής διευθυνσιοδότησης” και ”γραμμικής διερεύνησης” (linear probing). Υλοποιήστε με γραμμική διερεύνηση την αναζήτηση, την εισαγωγή και την διαγραφή. Στην ανοικτή διευθυνσιοδότηση με γραμμική διερεύνηση, κάθε θέση του πίνακα αποθηκεύει ένα μόνο στοιχείο. Όταν μία θέση είναι ήδη κατειλημμένη τότε αναζητάμε προς τα δεξιά την πρώτη διαθέσιμη θέση. Η αναζήτηση γίνεται κυκλικά στον πίνακα (modulo αριθμητική).

Καθολικός Κατακερματισμός - Μέθοδος του πίνακα Στην άσκηση αυτή πρέπει να υλοποιήσετε την μέθοδο του πίνακα για την κατασκευή μίας συνάρτησης καθολικού κατακερματισμού. Θυμηθείτε πως στην μέθοδο του πίνακα η συνάρτηση κατακερματισμού είναι ουσιαστικά ένας πίνακας με διαστάσεις $b \times u$ όπου u είναι αριθμός των bits των κλειδιών της εισόδου και b είναι ο αριθμός των bits των κλειδιών της εξόδου. Ταυτόχρονα η αριθμητική είναι modulo 2. Θεωρήστε πως το κλειδί εισόδου είναι ο ακέραιος που επιστρέφει η συνάντηση hashCode και άρα είναι $u = 32$ bits. Για ευκολία φροντίστε ο πίνακας κατακερματισμού σας να έχει μέγεθος μόνο δυνάμεις του 2 και άρα να έχει μέγεθος 2^b .
Κάθε φορά που αλλάζει το μέγεθος του πίνακα, δηλαδή αυξομειώνεται το b, θα πρέπει να κατασκευάζετε καινούριο πίνακα μεγέθους $b \times u$ με τυχαία 0 και 1. Διάβασμα των bits ενός ακεραίου στην Java μπορεί να πραγματοποιηθεί είτε χρησιμοποιώντας μάσκες και bitwise τελεστές ή με την χρήση πιο υψηλού επιπέδου APIs όπως για παράδειγμα την κλάση BitSet.

Δυναμικός Πίνακας Για την υποστήριξη οποιουδήποτε αριθμού αντικειμένων, η τεχνική είναι παρόμοια με τις άλλες δομές που έχουμε μελετήσει. Όταν ο αριθμός των αντικειμένων που έχουμε αποθηκευμένα μέσα στον πίνακα κατακερματισμού πλησιάζει το μέγεθος του πίνακα, τότε διπλασιάζουμε το μέγεθος του πίνακα. Προσοχή πως κατά την διάρκεια αλλαγής του μεγέθους του πίνακα πρέπει να κάνετε επανακερματισμό (rehashing) όλων των αντικειμένων. Αντίθετα όταν ο αριθμός των αντικειμένων γίνει $< 25\%$ της χωρητικότητας, υποδιπλασιάζουμε τον πίνακα και κάνουμε πάλι απανακερματισμό. Σημαντικό είναι πως η αλλαγή μεγέθους του πίνακα σημαίνει ταυτόχρονα πως πρέπει να διαλέξουμε καινούρια συνάρτηση κατακερματισμού.

Testing Κατά την διάρκεια του εργαστηρίου είδαμε πως μπορώ να γράψω ένα unit test. Γράψτε 3 unit tests που να δοκιμάζουν τα όρια της υλοποίησης σας.

Συχνότητες λέξεων σε αρχεία Για επίδειξη της λειτουργικότητας της υλοποίησης σας χρησιμοποιείστε το παράδειγμα του εργαστηρίου, δηλαδή ένα πρόγραμμα που χρησιμοποιώντας ένα hashtable μετράει συχνότητες λέξεων σε αρχεία κειμένου. Δώστε μία main με αυτή την λειτουργικότητα και δοκιμάστε τον κώδικα σας σε μεγάλα κείμενα. Περιγράψτε την δουλειά σας στο παραδοτέο.

Παραδοτέα Η άσκηση έχει ένα παραδοτέο που αποτελείται από 3 μέρη:

- Ο πηγαίος κώδικας ο οποίος θα πρέπει να μεταγλωττίζεται πολύ εύκολα χρησιμοποιώντας το maven. Δεν θα γίνουν δεκτές λύσεις που χρειάζονται εξωτερικά προγράμματα ή που δεν μεταγλωττίζονται επιτυχώς σε περιβάλλον Linux. Μπορείτε να χρησιμοποιήσετε τον σκελετό των projects από το εργαστήριο.
- Επιτρέπεται η χρήση μόνο Java έκδοσης 11 ή 17. Καμία άλλη έκδοση δεν γίνεται αποδεκτή.
- Μαζί με τον πηγαίο κώδικα θα πρέπει να υπάρχει και ένα αρχείο README το οποίο να περιγράφει την διαδικασία μεταγλώττισης και εκτέλεσης.
- Τέλος θα πρέπει να υπάρχει και ένα αρχείο report.pdf το οποίο να περιγράφει αναλυτικά την δουλειά σας, να εξηγεί τον κώδικα σας, και να περιέχει παραδείγματα εκτέλεσης του κώδικας σας. Προσοχή η βαθμολόγηση δεν γίνεται μόνο με βάση την λειτουργικότητα αλλά και με βάση την ποιότητα του κώδικα. Επιπρόσθετα σημαντικό ρόλο παίζει και η αναφορά.