**Αναφορά Κώδικα ΠΛΗ202 1η Άσκηση**

**Λεοντής Παναγιώτης 2018030099**

Η main() βρίσκεται στη κλάση Console.

Αρχικά ανοίγω το αρχείο με ένα RandomAccessFile στη μέθοδο openAndCreateList() της κλάσης *Console* και προσθέτω κάθε γραμμή ως έναν νέο αντικείμενο της κλάσης Node στη λίστα μου.

Έχω κάνει εισαγωγή του jar StandardInputRead από το εργαστηριακό ύλικο του Δομημένου Προγραμματισμού ώστε να διαβάζω τις προτάσεις που εισάγει ο χρήστης με το αντικείμενο reader.

**Για το 1ο μέρος της άσκησης.**

Για όλες τις λειτουργικότητες έχω ορίσει την κλάση *DoublyLinkedList* η οποία υλοποιεί μια διπλά συνδεδεμένη λίστα και επιπλέον περιέχει τις μεθόδους που απαιτούνται για τις επιθυμητές λειτουργίες

Ορίζω ένα αντικείμενο της κλάσης *Node*, globalPointer ο οποίος θα είναι η τρέχουσα θέση στη λίστα μου όσο εκτελείται το πρόγραμμα.

Για μεταφορά στην πρώτη γραμμή όρισα τον globalPointer να είναι ίσος με το head της λίστας μου *DoublyLinkedList* από τη μέθοδο getHead().

Αντίστοιχα για την μεταφορά στη τελευταία γραμμή μετακινώ τον globalPointer μέχρι το επόμενο node της λίστας να είναι null, δηλαδή βρισκόμαστε στην τελευταία θέση της λίστας.

Για να μετακινηθεί ο globalPointer μια γραμμή πάνω όρισα μια μέθοδο μέσα στην κλάση *DoublyLinkedList* η οποία παίρνει ως όρισμα ένα Node και επιστρέφει το προηγούμενό του με τη μέθοδο getPrevious() έπειτα από τους κατάλληλους ελέγχους. Έτσι το Node αυτό επιστρέφεται στον globalPointer.

Για να μετακινηθεί ο globalPointer μια γραμμή κάτω όρισα μια μέθοδο μέσα στην κλάση *DoublyLinkedList* η οποία παίρνει ως όρισμα ένα Node και επιστρέφει το επόμενό του με τη μέθοδο getNext() έπειτα από τους κατάλληλους ελέγχους. Έτσι το *Node* αυτό επιστρέφεται στον globalPointer.

Για να προσθέσω πρόταση στην επόμενη σειρά καθώς και στην προηγούμενη όρισα μία μέθοδο μέσα στη *DoublyLinkedList, addAfter() και addBefore()* για την κάθε περίπτωση αντίστοιχα. Αυτές παίρνουν ως όρισμα την πρόταση του χρήστη καθώς και τον globalPointer ώστε να γνωρίζουμε που θα τοποθετηθεί η πρόταση. Αναλόγως με τη θέση του globalPointer(αρχή, μέση, τέλος) ρυθμίζω κατάλληλα τις συνδέσεις μεταξύ των κόμβων και τοποθετώ τον νέο κόμβο στην επιθυμητή θέση.

Για την διαγραφή κόμβου έχω ορίσει τη μέθοδο deleteNode() μέσα στην *DoyblyLinkedList* η οποία παίρνει ως όρισμα τον globalPointer ώστε να γνωρίζουμε ποιον κόμβο να διαγράψουμε. Ανάλογα με την θέση του globalPointer (αρχή μέση τέλος) ρυθμίζω τις σχέσεις μεταξύ των κόμβων και τον μετακινώ είτε μια θέση πίσω είτε μία θέση μπροστά ώστε να βρίσκεται σε κάποιο κόμβο της λίστας και όχι στον διαγραμμένο.

Για την αρίθμηση σειρών ορίζω μία Boolean μεταβλητή enable μέσα στη main η οποία είναι false by default. Όταν επιλεχθεί η λειτουργία κάνω τον έλεγχο για την τιμή της enable και αναλόγως εκτελώ την μέθοδο printList() της *DoublyLinkedList*.

Για την εκτύπωση γραμμής καλώ την μέθοδο printNode() της *DoublyLinkedList* η οποία παίρνει ως όρισμα την Boolean enable και τον globalPointer για να γνωρίζει ποια γραμμή θα εκτυπώσει και για να γίνει εκτύπωση με αρίθμηση σειρών ή χωρίς.

Για έξοδο χωρίς αποθήκευση απλώς τερματίζω το πρόγραμμα.

Για γράψιμο αρχείου στο δίσκο χρησιμοποιώ τη μέθοδο saveFile() στην *Console* η οποία παίρνει ως όρισμα το RandomAccessFile με το οποίο ανοίξαμε το αρχείο στην αρχή του προγράμματος . Διασχίζω τη λίστα μου και σε ένα buffer τοποθετώ το String του κάθε κόμβου. Γράφω στο αρχείο το String έπειτα γράφω τον χαρακτήρα νέας σειράς και πάω στον επόμενο κόμβο. Τελικά πανωγράφουμε στο προηγούμενο αρχείο την τρέχουσα λίστα.

Για έξοδο και σώσιμο απλώς εκτελώ την προηγούμενη λειτουργία και έπειτα τερματίζω το πρόγραμμα.

Για να εμφανίσω τον αριθμό της τρέχουσας γραμμής καλώ την μέθοδο findNodeNumber() της *DoublyLinkedList* η οποία παίρνει ως όρισμα τον globalPointer. Μέσα σε αυτή ορίζω μια μεταβλητή και διασχίζω τη λίστα βήμα βήμα μέχρι να βρω τον κόμβο που ισούται με το globalPointer .Για κάθε μετακίνηση στη λίστα αυξάνω τη μεταβλητή κατά 1. Τελικά προκύπτει ο αριθμός της γραμμής.

Για να εμφανίσω συνολικές γραμμές και χαρακτήρες καλώ την μέθοδο της *DoublyLinkedList*, totalCharAndLines().Οι γραμμές είναι το size της λίστας και για τους χαρακτήρες παίρνω κάθε κόμβο έπειτα το String του και με τη μέθοδο .length() προσθέτω τους συνολικούς χαρακτήρες.

**<u>Για το 2$^{ο}$ μέρος της άσκησης.</u>**

Αρχικά στη μέθοδο createTupleList() της Console δημιουργώ μια λίστα από αντικείμενα της κλάσης *Tuple.* Από την υπάρχουσα λίστα παίρνω για την κάθε γραμμή κάθε λέξη ξεχωριστά και την γραμμή που βρίσκεται.

Για την δημιουργία αρχείου δεικτοδότησης χρησιμοποίησα τον κώδικα που προσφέρεται μέσα στην εκφώνηση της άσκησης «Εγγραφή/Ανάγνωση από δίσκο ανά σελίδες». Χρησιμοποιώ τη μέθοδο stringToBytes() της κλάσης *FileAccessor* που παίρνει σαν όρισμα την λίστα με τα *Tuples*. Διασχίζοντας τη λίστα παίρνω τα στοιχεία από κάθε κόμβο (*Tuple*) και δημιουργώ ζευγάρια των 24 byte (λέξη 20 byte, αριθμός γραμμής 4 byte). Προσθέτω στον buffer ζεύγη μέχρι να φτάσει το μέγιστο χώρο(ρυθμίσαμε 128 bytes) και όταν γεμίσει γράφω την πρώτη σελίδα στο αρχείο. Αδειάζω το buffer και επαναλαμβάνω μέχρι να διασχίσω όλη τη λίστα με τα *Tuples*.

Για την εκτύπωση του κάθε *Tuple*(λέξη, αριθμός γραμμής), χρησιμοποίησα τον κώδικα που προσφέρεται μέσα στην εκφώνηση της άσκησης «Εγγραφή/Ανάγνωση από δίσκο ανά σελίδες». Γεμίζω ένα byteArray με τα περιεχόμενα μιας σελίδας. Μέσα στην μέθοδο byteToString() της *FileAccessor* δημιουργώ μια νέα ArrayList από *Tuple* και το κάθε *Tuple* φτιάχνεται ύστερα από επεξεργασία των bytes ώστε να πάρω την λέξη και τον αριθμό σειράς.

Για τη σειριακή αναζήτηση καλώ τη μέθοδο searchSerial() της *FileAccessor* η οποία παίρνει ως όρισμα την λέξη που αναζητούμε και το σύνολο των σελίδων των 128 bytes που έχει το αρχείο μας. Φτιάχνω μια λίστα από *Tuples* διαβάζοντας κάθε φορά μία σελίδα. Ανακτώ την λέξη των 20 bytes και τον αριθμό των 4 bytes από το κάθε *Tuple* και για το μήκος της λίστας αναζητώ εάν η λέξη ισούται με την είσοδο του χρήστη. Για κάθε νέα σελίδα που διαβάζω αυξάνω κατά 1 τις προσβάσεις στο δίσκο και γράφω και την γραμμή που βρίσκεται η λέξη. Η διαδικασία επαναλαμβάνεται με την επόμενη σελίδα μέχρι να τις διαβάσω όλες. Επιπλέον έβαλα μια συνθήκη η οποία συγκρίνει την τελευταία λέξη της σελίδας με την είσοδο του χρήστη. Αν η είσοδος του χρήστη προηγείται αλφαβητικά της τελευταίας λέξης της σελίδας η δυαδική αναζήτηση σταματάει. Έτσι έχουμε λιγότερα Disk Accesses και το πρόγραμμα γίνεται πιο γρήγορο.

Για τη δυαδική αναζήτηση χρησιμοποιώ τρεις αναδρομικές μεθόδους της *FileAccessor*. Με την searchBinary() παίρνω την μεσαία σελίδα και συγκρίνω με την πρώτη και την τελευταία λέξη για να ξαναχωρίσω το αρχείο στην μέση από αριστερά η δεξιά. Όταν φτάσω στην σελίδα όπου πρέπει να βρίσκεται η λέξη συγκρίνω με κάθε λέξη της σελίδας και αποθηκεύω την γραμμή της λέξης εάν ισούται με αυτή του χρήστη. Αν η πρώτη ή η  τελευταία λέξη είναι ίση με αυτή του χρήστη ορίζω δύο Boolean μεταβλητές ώστε να γνωρίζω αν πρέπει να ψάξω πιο πριν ή πιο μετά. Αυτό γίνεται αναδρομικά και στην ουσία σειριακά με τις μεθόδους readPreviousPage() και readNextPage() αντίστοιχα. Στην searchBinary() έχω βάλει συνθήκη firstPage+1==lastPage για την λέξη του χρήστη όπου θα φτάσω σε τελικό σημείο την δυαδική αναζήτηση. Όταν φτάσω στο απροχώρητο την διαίρεση των συνολικών σελίδων η παραπάνω συνθήκη θα μου επιτρέψει να αποφύγω μία ατέρμονη αναδρομική κλήση.

Στην επόμενη σελίδα παρατίθενται οι μετρήσεις. Για την παραγωγή των 30 τυχαίων Strings χρησιμοποίησα τον σύνδεσμο που μας παρείχατε στις σημειώσεις και συγκεκριμένα τη κλάση *RandomString* από την μέθοδο 1. Ως μέγεθος του τυχαίου String χρησιμοποιώ 10 χαρακτήρες.

https://www.geeksforgeeks.org/generate-random-string-of-given-size-in-java/

Ο κώδικας για την τυχαία δοκιμή βρίσκεται μέσα στις επιλογές s (serial) και b (binary) του switch menu αντίστοιχα. Έχω προσθέσει μία συνθήκη ώστε να επιλέγετε την επιθυμητή λειτουργικότητα(30 τυχαίες δοκιμές ή αναζήτηση συγκεκριμένης λέξης).

| | Μέγεθος | Σελίδες | Rectors | laboratories | Technical | Venetian | 30 Random |
|---|---|---|---|---|---|---|---|
| testfile1 | 16640 | 130 | Serial: 38 Binary: 7 | Serial: 92 Binary: 6 | Serial: 46 Binary: 8 | Serial: 53 Binary:6 | Serial: 45 Binary: 6 |
| testfile_x2 | 33152 | 259 | Serial: 76 Binary: 8 | Serial: 184 Binary: 8 | Serial: 91 Binary:11 | Serial: 106 Binary: 7 | Serial: 110 Binary: 7 |
| testfile_x5 | 82688 | 646 | Serial: 189 Binary: 11 | Serial: 459 Binary: 12 | Serial:228 Binary: 22 | Serial: 265 Binary:10 | Serial: 226 Binary: 10 |
| testfile_x10 | 165376 | 1292 | Serial: 377 Binary: 12 | Serial: 917 Binary: 12 | Serial: 455 Binary: 38 | Serial: 529 Binary: 14 | Serial: 570 Binary: 11 |
| testfile_1000 | 16537600 | 129200 | Serial: 37601 Binary: 210 | Serial: 91601 Binary: 210 | Serial: 45401 Binary: 3206 | Serial: 52801 Binary: 806 | Serial: 57734 Binary: 17 |