Πανεπιστήμιο Πειραιώς
Τμήμα Πληροφορικής

Εργασία για το μάθημα "Δομές Δεδομένων"

Σύστημα Παρακολούθησης Ενεργών Πτήσεων

Δρόσου Ειρήνη, Π/06047
<eirinidrosou@gmail.com>

Κουζούπης Αντώνης, Π/06073
<kouzoupis.ant@gmail.com>

Πειραιάς, Δευτέρα 11 Ιουλίου 2011

Περιεχόμενα

1 Εισαγωγή

Η εφαρμογή υλοποιεί ένα απλοποιημένο σύστημα προβολής και διαχείρισης των ενεργών πτήσεων. Υλοποιήθηκε στα πλαίσια της δεύτερης εργασίας του μαθήματος *Δομές Δεδομένων*. Υλοποιεί κάποιες λειτουργίες πάνω σε δυαδικό δένδρο αναζήτησης. Η εφαρμογή είναι γραμμένη στη γλώσσα προγραμματισμού Java και οι δομές που χρησιμοποιήθηκαν δεν είναι οι built-in κλάσεις της γλώσσας αλλά υλοποιημένες από εμάς. Δυστυχώς λόγω ακαδημαϊκών υποχρεώσεων δεν προλάβαμε να υλοποιήσουμε το τέταρτο μέρος της εργασίας. Η εφαρμογή δεν χρησιμοποιεί κάποιο είδος persistance καθώς δεν χρειαζόταν από την εργασία.

2 Σχεδίαση

Η εφαρμογή είναι χωρισμένη σε τρία βασικά λειτουργικά κομμάτια. Τα τρία directories–packages της εφαρμογής είναι το business, το entities και το structure. Κάθε ένα package περιέχει κώδικα "ανεξάρτητο" από τα άλλα packages. Παρόλα αυτά χρειάζονται και τα τρία για να λειτουργήσει η εφαρμογή. Το πρώτο πακέτο, το business, περιέχει τον πηγαίο κώδικα για το business logic. Τυπώνει το μενού, συλλέγει τις πληροφορίες από το χρήστης, μετατρέπει τις raw πληροφορίες σε μορφή χρήσιμη και καλεί τις κατάλληλες μεθόδους για λειτουργίες πάνω στο δυαδικό δένδρο αναζήτησης. Στο δεύτερο πακέτο, το entities, υπάρχει ο κώδικας για την υλοποίηση των οντοτήτων του συστήματος. Στη συγκεκριμένη περίπτωση είναι οι πτήσεις. Τέλος στο πακέτο structures υπάρχουν οι κλάσεις των δομών δεδομένων που χρησιμοποιήσαμε οι οποίες προφανώς υλοποιούν τις δομές αλλά και κάποιες λειτουργίες πάνω σε αυτές.

Αναλυτικές πληροφορίες υπάρχουν και στο *JavaDoc* που συνοδεύει την εργασία αλλά και στον πηγαίο κώδικα.

2.1 business package

Το συγκεκριμένο πακέτο περιέχει τις κλάσεις *Business*, *Main* και *Printer*.

Η κλάση *Business* περιέχει τρεις βασικές μεθόδους. Η μία είναι για να φορτώνει κάποιες demo πτήσεις στο σύστημα, η δεύτερη προετοιμάζει τα raw δεδομένα που εισάγει ο χρήστης και καλεί τις κατάλληλες μεθόδους για την προσθήκη μιας νέας πτήσης και τέλος η τρίτη μέθοδος χρησιμοποιείται για την αναζήτηση των ενεργών πτήσεων σε ένα συγκεκριμένο χρονικό διάστημα.

Η κλάση *Main* είναι το entry point της εφαρμογής. Μέσω μεθόδων της κλάσης *Printer* τυπώνει το κεντρικό μενού, τα υπο–μενού και συλλέγει τις πληροφορίες από τον χρήστη.

Τελευταία είναι η κλάση *Printer* που αποτελείται από μεθόδους που προορίζονται αποκλειστικά στην εκτύπωση των διάφορων μενού και "ερωτήσεων" προς το χρήστη.

2.2 entities package

Το πακέτο *entities* περιέχει την κλάση *Flights*.

Η κλάση *Flights* κρατάει όλες τις λεπτομέρειες για μία πτήση. Ο πληροφορίες που κρατάει είναι ο κωδικός πτήσης, η ώρα αναχώρησης και η ώρα άφιξης. Ουσιαστικά ένα instance της κλάσης αυτής αντιπροσωπεύει μία πτήση. Επίσης έχει διάφορες άλλες μεθόδους για να θέτουμε και να διαβάζουμε τις τιμές των παραπάνω μεταβλητών (getters & setters). Τέλος υπάρχει μία μέθοδος για την εκτύπωση των λεπτομερειών μιας πτήσης.

2.3 structure package

Το πακέτο αυτό περιέχει τις κλάσεις *BinarySearchTree* και *SimplyLinkedList*.

Η κλάση *BinarySearchTree* υλοποιεί αρχικά ένα Δυαδικό Δένδρο Αναζήτησης καθώς και κάποιες επιπλέον λειτουργίες. Κάθε κόμβος του δένδρου αποθηκεύει instances της κλάσης *Flights*, δηλαδή αποθηκεύει πτήσεις. Κάθε κόμβος επίσης έχει ένα πεδίο **MaxArr** το οποίο κρατάει τη μέγιστη ώρα άφιξης του υποδένδρου του. Στην κλάση αυτή υπάρχει και η κλάση *Node* που αντιπροσωπεύει τους κόμβους του δένδρου. Κρατάει πληροφορίες για τον αριστερό κόμβο–παιδί, τον δεξιό κόμβο–παιδί, το πεδίο **MaxArr** και φυσικά τα δεδομένα. Έχει μεθόδους για την εισαγωγή, διαγραφή, αναζήτηση ενός κόμβου και εκτύπωση σε ενδοδιάταξη του δένδρου.

Η κλάση *SimplyLinkedList* υλοποιεί μία μονά συνδεδεμένη λίστα που επίσης χρησιμοποιεί Generics. Αποτελείται και αυτή από δύο κλάσεις, η μία είναι η *SNode* που κρατάει διάφορες πληροφορίες για ένα κόμβο όπως τη τιμή του κόμβου και τον επόμενο κόμβο. Έχει επίσης κάποιες μεθόδους για την προσπέλαση των στοιχείων αυτών. Η κλάση *SimplyLinkedList* υλοποιεί κάποιες λειτουργίες της μονά συνδεδεμένης λίστας όπως την προσθήκη ενός κόμβου είτε στην αρχή, είτε στο τέλος είτε κάπου ενδιάμεσα, τη διαγραφή ενός κόμβου, μέθοδο για την παίρνουμε το μέγεθος της λίστας, μέθοδο για να βλέπουμε αν η λίστα είναι άδεια και μία μέθοδο που τυπώνει τις τιμές όλων των κόμβων της λίστας. Η λίστα χρησιμοποιείται ως στοίβα (stack) για τις λειτουργίες της εισαγωγής και διαγραφής.

2.4 Δομή Εφαρμογής

Όπως αναφέραμε η εφαρμογή κρατάει κάποιες πτήσεις σε μία δομή δυαδικού δένδρου αναζήτησης και εκτελούμε κάποιες λειτουργίες σε αυτές. Μία πτήση χαρακτηρίζεται από τον κωδικό πτήσης, την ημερομηνία αναχώρησης και την ημερομηνία άφιξης. Στο δένδρο η ταξινόμηση γίνεται με βάση την ημερομηνία αναχώρησης. Επίσης κάθε κόμβος στο δένδρο έχει ένα ειδικό field με όνομα **MaxArr**. Αυτό το πεδίο κρατάει την μέγιστη ημερομηνία άφιξης του δένδρου κάτω από αυτό, δηλαδή του υποδένδρου του. Αυτό το πεδίο βοηθάει για διάφορες αναζητήσεις ώστε να μπορούμε να αποφασίζουμε να μην επισκεπτόμαστε κάποια υποδένδρα. Με αυτό τον τρόπο γλιτώνουμε συγκρίσεις. Προφανώς όταν γίνεται κάποια εισαγωγή ή διαγραφή, οι κόμβοι του δένδρου θα πρέπει να επανατοποθετηθούν στη σωστή θέση και να υπολογιστεί ξανά το πεδίο **MaxArr**.

3 Επεξήγηση Λειτουργίας

Η εφαρμογή είναι γραμμένη σε Java οπότε για να τρέξει ή να γίνει compile χρειαζόμαστε το Java Runtime Enviroment. Τα εκτελέσιμα είναι αρχειοθετημένα σε ένα archive τύπου jar για ποιο εύκολη εκτέλεση. Για να ξεκινήσουμε την εφαρμογή αρκεί να πάμε στον κατάλογο που είναι το αρχείο *DataStruct2.jar* και να εκτελέσουμε *java -jar DataStruct2.jar*. Αυτό θα μας εμφανίσει το κεντρικό μενού της εφαρμογής απ' όπου μπορούμε να κάνουμε τις επιλογές μας. Το κεντρικό μενού θα εμφανίζεται μετά από κάθε λειτουργία εκτός αν επιλέγουμε την Έξοδο από την εφαρμογή. Το entry point του συστήματος είναι η κλάση Main που φορτώνει κάποιες demo πτήσεις.

3.1 Add Flight

Η πρώτη επιλογή είναι για τη προσθήκη μιας πτήσης στην εφαρμογή. Επιλέγοντας λοιπόν την πρώτη επιλογή, μας εμφανίζεται ένα prompt για να εισάγουμε τα χαρακτηριστικά της πτήσης. Αφού τα εισάγουμε, η ημερομηνία αναχώρησης και ημερομηνία άφιξης μετατρέπονται σε χρόνο EPOCH. Έπειτα δημιουργείται ένα νέο instance της κλάσης *Flights* και καλείται η μέθοδος *add* της κλάσης *BinarySearchTree*. Αυτή η μέθοδος αποφασίζει ποιο είναι το σωστό σημείο στο δένδρο για να μπει και ρυθμίζει κατάλληλα τους δείκτες προς το αριστερό και δεξιό παιδί. Σε ένα stack κρατάει τους κόμβους που έκανε traverse μέχρι να βρει τη σωστή θέση ώστε μετά να ελέγξει αν χρειάζεται επανυπολογισμός του πεδίου **MaxArr** σε κάθε ένα κόμβο από το stack.

3.2 List Flights

Δεύτερη επιλογή είναι για την προβολή όλων των πτήσεων. Η επιλογή αυτή καλεί τη μέθοδο *toString* της κλάσης *BinarySearchTree* και αφού μετατρέψει το δένδρο σε μία λίστα με ενδοδιάταξη, τυπώνει όλα τα περιεχόμενα.

3.3 Delete Flight

Με την επιλογή *Delete Flight* μπορεί ο χρήστης να διαγράψει μία πτήση από το σύστημα. Προφανώς αφού διαγραφεί ένας κόμβος, το δυαδικό δένδρο αναζήτησης θα πρέπει να έχει τη σωστή δομή και το πεδίο **MaxArr** να έχει τη σωστή τιμή σε όλους τους κόμβους. Αρχικά ο χρήστης δίνει ένα κωδικό πτήσης, καλείται η μέθοδος *delFlight* της κλάσης *BinarySearchTree* όπου βρίσκει σε ποιο κόμβο αντιστοιχεί ο συγκεκριμένος κωδικός πτήσης. Έπειτα καλείται η μέθοδος *remove* της ίδιας κλάσης όπου διαγράφει τον κόμβο και αποφασίζει ποιος κόμβος θα πάρει τη θέση του στο δένδρο. Επίσης κρατάει σε μία λίστα όλους τους κόμβους που έχουν γίνει traversed μέχρι τον κόμβο προς διαγραφεί έτσι ώστε μετά να ελέγξει αν χρειάζεται αλλαγή το πεδίο **MaxArr**.

3.4 Search within a period

Επόμενη επιλογή είναι η αναζήτηση πτήσεων σε εξέλιξη σε ένα συγκεκριμένο χρονικό διάστημα. Ο χρήστης δίνει την αρχή και το τέλος του χρονικού διαστήματος και επιστρέφονται οι πτήσεις που άρχισαν στο διάστημα αυτό ή ολοκληρώθηκαν στο διάστημα αυτό. Αφού ο χρήστης δώσει την αρχή και το τέλος καλείται η μέθοδος *searchFlight* της κλάσης *Business*, μετατρέπει τις ημερομηνίες σε χρόνους EPOCH και καλεί τη μέθοδο *searchPeriod* της κλάσης *BinarySearchTree*. Η μέθοδος αυτή επιστρέφει μία λίστα με όλες τις πτήσεις που είναι σε εξέλιξη στη συγκεκριμένη χρονική περίοδο και τελικά τις τυπώνει.

3.5 Exit

Τελευταία επιλογή είναι η *Exit* με την οποία μπορούμε να τερματίσουμε και να βγούμε από την εφαρμογή.

4 Περιεχόμενα CD

Στο CD που συνοδεύει το παρόν εγχειρίδιο υπάρχουν τα παρακάτω:

src Ο πηγαίος κώδικας της εφαρμογής. Περιέχει τρεις καταλόγους που αντιστοιχούν στα τρία πακέτα. Το κάθε πακέτο έχει τον κώδικα από τις κλάσεις του.

doc Το Javadoc της εφαρμογής. Περιγραφή της λειτουργίας κάθε μεθόδου της εφαρμογής. Μέσα στον κατάλογο αυτό υπάρχει το αρχείο *index.html* το οποίο το ανοίγετε με ένα browser. (Σε ορισμένους browsers ίσως χρειαστεί να θέσετε με το χέρι την κωδικοποίηση σε UTF-8)

Documentation.pdf Το παρόν εγχειρίδιο.

DataStruct2.jar jar archive το οποίο περιέχει τον εκτελέσιμο κώδικα.

README Αρχείο με πληροφορίες για την εκτέλεση της εφαρμογής.

5 Πηγαίος Κώδικας

Ακολουθεί ο πηγαίος κώδικας της εφαρμογής χωρισμένος σε κατηγορίες σύμφωνα με τα packages. Δυστυχώς το LaTeX έχει πρόβλημα στο να εμφανίζει τα σχόλια στα ελληνικά.

5.1 business

5.1.1 Business.java

```
1  package business;
2
3  import entities.Flights;
4  import structure.BinarySearchTree;
5  import structure.SimplyLinkedList;
6
7  import java.util.GregorianCalendar;
8  import java.util.Date;
9  /**
10  * Κλάσηγιατηνκατάλληληδιαμόρφωσητωνδεδομένωνπρινσταλούνστιςμεθόδους
11  * τουδυαδικούδένδρουαναζήτησης . Επίσηςπεριέχειβοηθητικέςμεθόδους .
12  */
13 public class Business {
14         //Field πουδηλώνειποιοείναιτοδυαδικόδένδροαναζήτησηςγιατη
15         //συγκεκριμένη κλάση
16         private BinarySearchTree<Flights> bst;
17
18         /**
19          * Ορίζειτο field bst.
20          *
21          * @param bst Τοδυαδικόδένδροαναζήτησηςπουχρησιμοποιείηεφαρμογή .
22          */
23         public void setBST(BinarySearchTree<Flights> bst){
24                 this.bst=bst;
25         }
26         /**
27          * Βοηθητικήμέθοδοςγιατηπροσθήκη demo πτήσεωνστοσύστημα .
28          * @see BinarySearchTree#add(Flights)
29          * @see Flights#Flights(String, Date, Date)
30          */
31         public void loadFlights(){
32                 //Μετατροπή ημερομηνίαςτηςμορφής YYYY:MM:DD:HH:MM σεκατάλληλη
33                 //μορφή γιαδιάφορεςδιαδικασίες .
34                 Date departureTime=new GregorianCalendar(2011,05,20,10,00).getTime();
35                 Date arrivalTime=new GregorianCalendar(2011,05,20,11,00).getTime();
36                 //Δημιουργία ενόςνέου instance τηςκλάσης Flights πουαντιπροσωπεύει
37                 //μία πτήση.
38                 Flights newFlight=new Flights("a",departureTime,arrivalTime);
39                 //Προσθήκη τηςπτήσηςστοδυαδικόδένδροαναζήτησης .
40                 bst.add(newFlight);
41
42                 departureTime=new GregorianCalendar(2011,05,20,9,00).getTime();
43                 arrivalTime=new GregorianCalendar(2011,05,20,10,00).getTime();
44                 newFlight=new Flights("b",departureTime,arrivalTime);
45                 bst.add(newFlight);
46
47                 departureTime=new GregorianCalendar(2011,05,20,11,00).getTime();
48                 arrivalTime=new GregorianCalendar(2011,05,20,12,00).getTime();
49                 newFlight=new Flights("c",departureTime,arrivalTime);
50                 bst.add(newFlight);
51
52                 departureTime=new GregorianCalendar(2011,05,20,8,00).getTime();
53                 arrivalTime=new GregorianCalendar(2011,05,20,9,00).getTime();
54                 newFlight=new Flights("d",departureTime,arrivalTime);
55                 bst.add(newFlight);
56
```

```
57                departureTime=new GregorianCalendar(2011,05,20,9,10).getTime();
58                arrivalTime=new GregorianCalendar(2011,05,20,10,10).getTime();
59                newFlight=new Flights("e",departureTime,arrivalTime);
60                bst.add(newFlight);
61
62                departureTime=new GregorianCalendar(2011,05,20,10,50).getTime();
63                arrivalTime=new GregorianCalendar(2011,05,20,11,50).getTime();
64                newFlight=new Flights("f",departureTime,arrivalTime);
65                bst.add(newFlight);
66
67                departureTime=new GregorianCalendar(2011,05,20,11,10).getTime();
68                arrivalTime=new GregorianCalendar(2011,05,20,12,10).getTime();
69                newFlight=new Flights("g",departureTime,arrivalTime);
70                bst.add(newFlight);
71
72                departureTime=new GregorianCalendar(2010,05,20,9,00).getTime();
73                arrivalTime=new GregorianCalendar(2010,05,20,10,00).getTime();
74                newFlight=new Flights("h",departureTime,arrivalTime);
75                bst.add(newFlight);
76        }
77        /**
78         * Μέθοδοςγιατηπροσθήκημιαςνέαςπτήσηςστοσύστημα          .
79         *
80         * @param flightCode Οκωδικόςπτήσης   .
81         * @param depTime Ηώρααναχώρησης   .
82         * @param arrTime Ηώραάφιξης   .
83         * @see Flights#Flights(String, Date, Date)
84         * @see BinarySearchTree#add(Flights)
85         */
86        public void addFlight(String flightCode, String depTime, String arrTime){
87                //Χωρίζει σε tokens τηνημερομηνίααναχώρισηςκαιάφιξηςσύμφωναμε
88                //συγκεκριμένο οριοθέτη.
89                String[] tmpDep=depTime.split("[:]");
90                String[] tmpArr=arrTime.split("[:]");
91                //Μετατροπή τηςημερομηνίαςσεκατάλληλημορφήγιαεπεξεργασία       .
92                Date departureTime=new GregorianCalendar(Integer.parseInt(tmpDep[0]),
93                                Integer.parseInt(tmpDep[1])-1, Integer.parseInt(tmpDep[2]),
94                                Integer.parseInt(tmpDep[3]), Integer.parseInt(tmpDep[4])).
95                                getTime();
96                Date arrivalTime=new GregorianCalendar(Integer.parseInt(tmpArr[0]),
97                                Integer.parseInt(tmpArr[1])-1, Integer.parseInt(tmpArr[2]),
98                                Integer.parseInt(tmpArr[3]), Integer.parseInt(tmpArr[4])).
99                                getTime();
100               //Δημιουργία ενόςνέου  instance τηςκλάσης  Flights πουαντιπροσωπεύει
101               //μία πτήση.
102               Flights newFlight=new Flights(flightCode,departureTime,arrivalTime);
103               //Προσθήκη τηςνέαςπτήσηςστοδυαδικόδένδροαναζήτησης       .
104               bst.add(newFlight);
105       }
106       /**
107        * Μέθοδοςγιατηναναζήτησημιαςπτήσηςστοδυαδικόδένδροαναζήτησης          .
108        *
109        * @param startTime Αρχήτηςχρονικήςπεριόδου   .
110        * @param finishTime Τέλοστηςχρονικήςπεριόδου   .
111        * @see BinarySearchTree#searchPeriod(Date, Date)
112        * @see SimplyLinkedList#toString()
113        */
114       public void searchFlight(String startTime, String finishTime){
115               //Χωρίζει σε tokens τηνημερομηνίααναχώρισηςκαιάφιξηςσύμφωναμε
116               //συγκεκριμένο οριοθέτη.
117               String[] tmpSt=startTime.split("[:]");
118               String[] tmpFi=finishTime.split("[:]");
119               //Μετατροπή τηςημερομηνίαςσεκατάλληλημορφήγιαεπεξεργασία       .
120               Date startT=new GregorianCalendar(Integer.parseInt(tmpSt[0]),
121                               Integer.parseInt(tmpSt[1])-1, Integer.parseInt(tmpSt[2]),
122                               Integer.parseInt(tmpSt[3]), Integer.parseInt(tmpSt[4])).getTime();
123               Date finishT=new GregorianCalendar(Integer.parseInt(tmpFi[0]),
124                               Integer.parseInt(tmpFi[1])-1,Integer.parseInt(tmpFi[2]),
```

```
125                                     Integer.parseInt(tmpFi[3]), Integer.parseInt(tmpFi[4])).getTime();
126                     //Καλεί τηκατάλληλημέθοδοτουδυαδικούδένδρουαναζήτησης .
127                     //Επιστρέφει μίαλίσταμετιςπτήσεις .
128                     SimplyLinkedList<Flights> periodS=bst.searchPeriod(startT, finishT);
129                     //Εκτύπωση τηςπαραπάνωλίστας . Καλείταιμέθοδος toString
130                     System.out.println(periodS);
131             }
132     }
```

5.1.2 Main.java

```
1   package business;
2
3   import structure.BinarySearchTree;
4   import entities.Flights;
5
6   import java.util.Scanner;
7
8   /**
9    * Κεντρικήκλάσητηςεφαρμογήςπουτυπώνειτομενού , συλλέγειτιςπληροφορίες
10   * απότοχρήστηκαικαλείτιςκατάλληλεςμεθόδους .
11   */
12  public class Main {
13
14          /**
15           * Κεντρικήμέθοδοςπουτυπώνειτομενου , συλλέγειτιςπληροφορίεςαπότο
16           * χρήστη, αρχικοποιείμεταβλήτεςκαικαλείτιςκατάλληλεςμεθόδους .
17           *
18           * @see BinarySearchTree#BinarySearchTree()
19           * @see Business#Business()
20           * @see Business#setBST(BinarySearchTree)
21           * @see Business#loadFlights()
22           * @see Business#addFlight(String, String, String)
23           * @see BinarySearchTree#toString()
24           * @see BinarySearchTree#delFlight(String)
25           * @see Business#searchFlight(String, String)
26           */
27          public static void main(String[] args) {
28                  //Αντικείμενο τύπου BinarySearchTree
29                  BinarySearchTree<Flights> bst=new BinarySearchTree<Flights>();
30                  //Instance τηςκλάσης Business
31                  Business bus=new Business();
32                  //Ορίζει στηκλάση Business ναχρησιμοποιείτοαντικείμενο bst
33                  //για δυαδικόδένδροαναζήτησης .
34                  bus.setBST(bst);
35                  //Φορτώνει ορισμένες demo πτήσεις.
36                  bus.loadFlights();
37                  //Όσο είναι true ημεταβλητή running, μετάαπόκάθελειτουργία
38                  //τυπώνεται καιτοκεντρικόμενού .
39                  boolean running=true;
40                  Scanner in=new Scanner(System.in);
41                  while(running){
42                          //Τυπώνει τοκεντρικόμενού .
43                          System.out.println(Printer.printMain());
44                          //Παίρνει τηνεπιλογήτουχρήστη .
45                          int choice=in.nextInt();
46                          switch(choice){
47                          //Περίπτωση γιατηπροσθήκημιαςνέαςπτήσηςστηνεφαρμογή .
48                          case 1:
49                                  Scanner inAdd=new Scanner(System.in);
50                                  //Εκτύπωση καισυλλογήπληροφοριώνγιατηνέαπτήση .
51                                  System.out.println(Printer.printAddCode());
52                                  String flightCode=inAdd.nextLine();
53                                  System.out.println(Printer.printaddDep());
54                                  String departureTime=inAdd.nextLine();
55                                  System.out.println(Printer.printaddArr());
```

```
56                                      String arrivalTime=inAdd.nextLine();
57                                      //Κλήση μεθόδουγιατηνοριστικήπροσθήκητηςπτήσηςστην
58                                      //εφαρμογή
59                                      bus.addFlight(flightCode, departureTime, arrivalTime);
60                                      break;
61                              //Περίπτωση γιατηπροβολήόλωντωνπτήσεων .
62                              case 2:
63                                      //Εκτύπωση τουδυαδικούδένδρουαναζήτησης . Καλείταιημέθοδος
64                                      //toString
65                                      System.out.println(bst);
66                                      break;
67                              //Περίπτωση γιατηδιαγραφήμιαςπτήσης
68                              case 3:
69                                      Scanner delFl=new Scanner(System.in);
70                                      //Εκτύπωση τουμενούγιατηδιαγραφή .
71                                      System.out.println(Printer.printDel());
72                                      //Συλλογή τουκωδικούπτήσης .
73                                      String delFlightCode=delFl.nextLine();
74                                      //Κλήση τηςμεθόδουγιατηδιαγραφήτηςπτήσηςαπότοδένδρο .
75                                      bst.delFlight(delFlightCode);
76                                      break;
77                              //Περίπτωση γιατηναναζήτησητουδεύτερουερωτήματος .
78                              case 4:
79                                      Scanner inSea=new Scanner(System.in);
80                                      //Εκτύπωση καισυλλογήτωνχρονικώνστιγμώντηςαναζήτησης .
81                                      System.out.println(Printer.printSearchPerA());
82                                      String startTime=inSea.nextLine();
83                                      System.out.println(Printer.printSearchPerB());
84                                      String finishTime=inSea.nextLine();
85                                      //Κλήση τηςμεθόδουγιατηναναζήτηση
86                                      bus.searchFlight(startTime, finishTime);
87                                      break;
88                              //Περίπτωση γιατηνέξοδοαπότηνεφαρμογή .
89                              case 0:
90                                      running=false;
91                                      break;
92                              default:
93                                      running=false;
94                                      break;
95                              }
96                      }
97              }
98
99      }
```

5.1.3 Printer.java

```
1   package business;
2   /**
3    * Κλάσηγιατηνεκτύπωσητωνμενούτηςεφαρμογής .
4    */
5   public class Printer {
6           /**
7            * Μέθοδοςγιατηνεκτύπωσητουκεντρικούμενού .
8            *
9            * @return Τοκεντρικόμενούσε String
10           */
11          public static String printMain(){
12                  StringBuilder mainBuild=new StringBuilder();
13                  mainBuild.append("Please make your choice:");
14                  mainBuild.append("\n");
15                  mainBuild.append("1 — Add a flight");
16                  mainBuild.append("\n");
17                  mainBuild.append("2 — List flights");
18                  mainBuild.append("\n");
19                  mainBuild.append("3 — Delete flight");
```

```
20                  mainBuild.append("\n");
21                  mainBuild.append("4 — Search within a period");
22                  mainBuild.append("\n");
23                  mainBuild.append("5 — Flight after specified time");
24                  mainBuild.append("\n");
25                  mainBuild.append("0 — Exit");
26                  mainBuild.append("\n");
27
28                  return mainBuild.toString();
29          }
30          /**
31           * Μέθοδοςγιατηνεκτύπωσητουμενούγιατηπροσθήκηενόςνέουκωδικού
32           * πτήσης.
33           *
34           * @return Τομενούγιατηπροσθήκηενόςνέουκωδικούπτήσηςσε String.
35           */
36          public static String printAddCode(){
37                  StringBuilder addBuildC=new StringBuilder();
38                  addBuildC.append("Enter the flight code:");
39                  addBuildC.append("\n");
40
41                  return addBuildC.toString();
42          }
43          /**
44           * Μέθοδοςγιατηνεκτύπωσητουμενούγιατηπροσθήκημιαςνέαςημερομηνίας
45           * αναχώρησης.
46           *
47           * @return Τομενούγιατηπροσθήκημιαςνέαςημνιας / αναχώρησηςσε String.
48           */
49          public static String printaddDep(){
50                  StringBuilder addBuildD=new StringBuilder();
51                  addBuildD.append("Enter departure time: (YYYY:MM:DD:HH:MM)");
52                  addBuildD.append("\n");
53
54                  return addBuildD.toString();
55          }
56          /**
57           * Μέθοδοςγιατηνεκτύπωσητουμενούγιατηπροσθήκημιαςνέαςημερομηνίας
58           * άφιξης.
59           *
60           * @return Τομενούγιατηπροσθήκημιαςνέαςημνιας / άφιξηςσε String.
61           */
62          public static String printaddArr(){
63                  StringBuilder addBuildA=new StringBuilder();
64                  addBuildA.append("Enter arrival time: (YYYY:MM:DD:HH:MM)");
65                  addBuildA.append("\n");
66
67                  return addBuildA.toString();
68          }
69          /**
70           * Μέθοδοςγιατηνεκτύπωσητουμενούγιατηδιαγραφήμιαςπτήσης .
71           *
72           * @return Τομενούγιατηδιαγραφήμιαςπτήσηςσε String.
73           */
74          public static String printDel(){
75                  StringBuilder delBuild=new StringBuilder();
76                  delBuild.append("Enter flight's code:");
77                  delBuild.append("\n");
78
79                  return delBuild.toString();
80          }
81          /**
82           * Μέθοδοςγιατηνεκτύπωσητουμενούγιατηναναζήτησητουδεύτερου
83           * ερωτήματος.
84           *
85           * @return Τομενούγιατηνεκτύπωσητουμενούγιατηναναζήτησητου
86           * δεύτερουερωτήματος .
87           */
```

```
88          public static String printSearchPerA(){
89                  StringBuilder seBuild=new StringBuilder();
90                  seBuild.append("Enter start time: (YYYY:MM:DD:HH:MM)");
91                  seBuild.append("\n");
92
93                  return seBuild.toString();
94          }
95          /**
96           * Μέθοδοςγιατηνεκτύπωσητουμενούγιατηναναζήτησητουδεύτερου
97           * ερωτήματος.
98           *
99           * @return Τομενούγιατηνεκτύπωσητουμενούγιατηναναζήτησητου
100          * δεύτερουερωτήματος .
101          */
102         public static String printSearchPerB(){
103                 StringBuilder seBuild=new StringBuilder();
104                 seBuild.append("Enter finish time: (YYYY:MM:DD:HH:MM)");
105                 seBuild.append("\n");
106
107                 return seBuild.toString();
108         }
109 }
```

5.2 entities

5.2.1 Flights.java

```
1   package entities;
2
3   import java.util.Date;
4
5   /**
6    * Κλάσηπουκρατάειτιςπληροφορίεςτωνπτήσεων . Το instance τηςαντιπροσωπεύει
7    * μίαπτήση .
8    */
9   public class Flights {
10          //Ο κωδικόςπτήσης
11          private String flightCode;
12          //Η ημερομηνίαάφιξης
13          private Date arrivalTime;
14          //Η ημερομηνίααναχώρησης
15          private Date departureTime;
16
17          /**
18           * Constructor πουθέτειστα private fields συγκεκριμένεςτιμές .
19           *
20           * @param flightCode Οκωδικόςπτήσης .
21           * @param departureTime Ηώραναχώρησης .
22           * @param arrivalTime Ηώραάφιξης .
23           */
24          public Flights(String flightCode, Date departureTime, Date arrivalTime){
25                  this.flightCode=flightCode;
26                  this.arrivalTime=arrivalTime;
27                  this.departureTime=departureTime;
28          }
29          /**
30           * Επιστρέφειτονκωδικόπτήσης .
31           *
32           * @return Τονκωδικόπτήσης .
33           */
34          public String getFlightCode(){
35                  return flightCode;
36          }
37          /**
38           * Επιστρέφειτηνημερομηνίαάφιξης .
```

```
39          *
40          * @return Τηνημερομηνίαάφιξης .
41          */
42         public Date getArrivalTime(){
43                 return arrivalTime;
44         }
45         /**
46          * Επιστρέφειτηνημερομηνίααναχώρησης .
47          *
48          * @return Τηνημερομηνίααναχώρησης .
49          */
50         public Date getDepartureTime(){
51                 return departureTime;
52         }
53         /**
54          * Θέτειτονκωδικόπτήσης .
55          *
56          * @param flightCode Οκωδικόςπτήσης .
57          */
58         public void setFlightCode(String flightCode){
59                 this.flightCode=flightCode;
60         }
61         /**
62          * Θέτειτηνημερομηνίαάφιξης .
63          *
64          * @param arrivalTime Ημερομηνίαάφιξης .
65          */
66         public void setArrivalTime(Date arrivalTime){
67                 this.arrivalTime=arrivalTime;
68         }
69         /**
70          * Θέτειτηνημερομηνίααναχώρησης .
71          *
72          * @param departureTime Ημερομηνίααναχώρησης .
73          */
74         public void setDepartureTime(Date departureTime){
75                 this.departureTime=departureTime;
76         }
77         /**
78          * Επιστρέφειτηνημερομηνίααναχώρησηςσε EPOCH.
79          *
80          * @return Τηνημερομηνίααναχώρησηςσε EPOCH.
81          */
82         public long getDepTime(){
83                 return departureTime.getTime();
84         }
85         /**
86          * Επιστρέφειτηνημερομηνίαάφιξησε EPOCH.
87          *
88          * @return Τηνημερομηνίαάφιξησε EPOCH.
89          */
90         public long getArrTime(){
91                 return arrivalTime.getTime();
92         }
93         /**
94          * Επιστρέφειτιςλεπτομέρειεςτηςπτήσης .
95          *
96          * @return Τιςλεπτομέρειεςτηςπτήσης .
97          */
98         @Override
99         public String toString(){
100                StringBuilder sb=new StringBuilder();
101                sb.append("\n");
102                sb.append("Flight Code: ");
103                sb.append(flightCode).append("\n");
104                sb.append("Departure Time: ");
105                sb.append(departureTime).append("\n");
106                sb.append("Arrival Time: ");
```

```
107                     sb.append(arrivalTime).append("\n");
108
109                     return sb.toString();
110             }
111     }
```

5.3 structure

5.3.1 BinarySearchTree.java

```
1   package structure;
2
3   import entities.*;
4
5   import java.util.Date;
6
7   /**
8    * Κλάσηπουαντιπροσωπεύειένακόμβοστοδυαδικόδένδροαναζήτησης          .
9    */
10  class Node{
11          //Αριστερός κόμβοςπαιδί–
12          private Node leftNode;
13          //Δεξιός κόμβοςπαιδί—
14          private Node rightNode;
15          //Τα δεδομέναπουκρατάειοκόμβος
16          private Flights data;
17          private long MaxArr;
18
19          /**
20           * Επιστρέφειτοναριστερόκόμβοπαιδί    –.
21           *
22           * @return Τοναριστερόκόμβοπαιδί  –.
23           */
24          public Node getLeftNode(){
25                  return leftNode;
26          }
27          /**
28           * Επιστρέφειτονδεξιόκόμβοπαιδί    –.
29           *
30           * @return Τονδεξιόκόμβοπαιδί  –.
31           */
32          public Node getRightNode(){
33                  return rightNode;
34          }
35          /**
36           * Επιστρέφειταδεδομένατουκόμβου     .
37           *
38           * @return Ταδεδομένατουκόμβου    .
39           */
40          public Flights getData(){
41                  return data;
42          }
43          /**
44           * Επιστρέφειτο field MaxArr.
45           *
46           * @return To field MaxArr.
47           */
48          public long getMaxArr(){
49                  return MaxArr;
50          }
51          /**
52           * Ορίζειτοναριστερόκόμβοπαιδί    –.
53           *
54           * @param leftNode Οαριστερόςκόμβοςπαιδί  –.
55           */
```

```
56          public void setLeftNode(Node leftNode){
57                  this.leftNode=leftNode;
58          }
59          /**
60           * Ορίζειτονδεξιόκόμβοπαιδί  –.
61           *
62           * @param rightNode Οδεξιόςκόμβοςπαιδί  –.
63           */
64          public void setRightNode(Node rightNode){
65                  this.rightNode=rightNode;
66          }
67          /**
68           * Ορίζειταδεδομέναπουθακρατάειοκόμβος       .
69           *
70           * @param data Ταδεδομένατουκόμβου   .
71           */
72          public void setData(Flights data){
73                  this.data=data;
74          }
75          /**
76           * Ορίζειτηντιμήτου    field MaxArr.
77           *
78           * @param MaxArr Ητιμήτου   field MaxArr
79           */
80          public void setMaxArr(long MaxArr){
81                  this.MaxArr=MaxArr;
82          }
83          /**
84           * Constructor πουαρχικοποιείτα   fields.
85           *
86           * @param data Ταδεδομέναπουθακρατάειοκόμβος       .
87           */
88          Node(Flights data){
89                  this.leftNode=null;
90                  this.rightNode=null;
91                  this.data=data;
92                  this.MaxArr=0L;
93          }
94          /**
95           * oCnstructor πουδημιουργείένανκενόκόμβο    .
96           */
97          Node(){
98                  this.leftNode=null;
99                  this.rightNode=null;
100                 this.data=null;
101                 this.MaxArr=0L;
102         }
103         /**
104          * Constructor πουαντιγράφειένακόμβοσεέναακαινούργιο       .
105          *
106          * @param node Οκόμβοςπουθααντιγραφεί    .
107          */
108         Node(Node node){
109                 this.leftNode=node.getLeftNode();
110                 this.rightNode=node.getRightNode();
111                 this.data=node.getData();
112                 this.MaxArr=node.getMaxArr();
113         }
114         /**
115          * Βρίσκειταδεδομέναμετηνελάχιστητιμή       .
116          *
117          * @return Ταδεδομέναμετηνελάχιστητιμή      .
118          */
119         private Flights minValue(){
120                 if(leftNode==null)
121                         return data;
122                 else return leftNode.minValue();
123         }
```

```
124         /**
125          * Διαγράφειένακόμβοαπότοδένδρο      .
126          *
127          * @param data Ταδεδομέναπουθαπρέπειναδιαγραφούναπότοδένδρο          .
128          * @param parent Οπατέραςτουκόμβουπουεξετάζεταισεκάθε          recursion.
129          * @param travDelPath Ηλίσταμετουςκόμβουςπουέχουνγίνει          traversed
130          * μέχριναβρεθείοκόμβοςπουμαςενδιαφέρει      .
131          * @return true ανδιαγραφείεπιτυχώς  , αλλιώς false.
132          */
133         public boolean remove(Flights data, Node parent, SimplyLinkedList<Node> travDelPath){
134                 //Προσθέτει τονεαυτότουο  ( κόμβος) στηλίσταμετους    traversed κόμβους
135                 //μέχρι ναβρεθείαυτόςπουμαςενδιαφέρει      .
136                 travDelPath.addHead(this);
137                 //Συγκρίνουμε ταδεδομένατουκόμβουπουμαςενδιαφέρεικαιτουτρέχοντα
138                 //κόμβου πουγίνεταιτο   recursion.
139                 long compareR=data.getDepTime()-this.data.getDepTime();
140                 //Αν ηδιαφοράείναιμεγαλύτερηαπότομηδένκοιτάμεστοδεξιό
141                 //υποδένδρο
142                 if(compareR>0){
143                         //Αν οδεξιόςκόμβοςδενείναι      null
144                         if(rightNode!=null)
145                                 //Ξανακαλούμε τηνίδιαμέθοδο   (recursion)
146                                 return rightNode.remove(data, this, travDelPath);
147                         else
148                                 return false;
149                 //Αν ηδιαφοράείναιμικρότερηαπότομηδένκοιτάμεστοαριστερό
150                 //υποδένδρο
151                 }else if(compareR<0){
152                         //Αν οαριστερόςκόμβοςδενείναι      null
153                         if(leftNode!=null)
154                                 //Ξανακαλούμε τηνίδιαμέθοδο   (recursion)
155                                 return leftNode.remove(data, this, travDelPath);
156                         else
157                                 return false;
158                 //Αν ηδιαφοράείναιίσημετομηδέν       , βρίκαμετονκόμβοπουμας
159                 //ενδιαφέρει
160                 }else{
161                         //Αν ταπαιδιάτουκόμβουδενείναι      null
162                         if(leftNode!=null && rightNode!=null){
163                                 //Στα δεδομένατουκόμβουπροςδιαγραφή     , βάζουμεταδεδομένα
164                                 //με τηνελάχιστητιμή
165                                 this.data=rightNode.minValue();
166                                 //Διαγράφουμε τονκόμβοαπότονοποίοαντιγράψαμεταδεδομένα
167                                 //παραπάνω.
168                                 rightNode.remove(this.data, this, travDelPath);
169                         //Αν οκόμβοςπουμαςενδιαφέρειτοαριστερόπαίδιτουπατέρατου
170                         }else if(parent.leftNode==this){
171                                 //Αν τοαριστερόπαιδίτουκόμβουπουμαςενδιαφέρειδενείναι
172                                 //null, τότεγιααριστερόπαιδίτουπατέραβάζουμεαυτό       , αλλιώς
173                                 //βάζουμε τοδεξιό .
174                                 parent.leftNode=(leftNode!=null)?leftNode:rightNode;
175                         }else if(parent.rightNode==this){
176                                 parent.rightNode=(leftNode!=null)?leftNode:rightNode;
177                         }
178                         return true;
179                 }
180         }
181 }
182
183 /**
184  * Κλάσηπουαντιπροσωπεύειτοδυαδικόδένδροαναζήτησηςτηςεφαρμογής      .
185  *
186  * @param <E> Οτύποςδεδομένωνπουθακρατάειτοδυαδικόδένδροαναζήτησης      .
187  */
188 public class BinarySearchTree<E>{
189         Node root;
190         int size=0;
191         Node resultNode=null;
```

```
192         //Λίστα πουκρατάμετουςκόμβουςπουέχουνγίνει traversed κατάτηνεισαγωγή
193         //ενός νέουκόμβουστοδένδρο .
194         SimplyLinkedList<Node> travAddPath;
195         //Λίστα μετααποτελέσματατηςαναζήτησης .
196         SimplyLinkedList<Flights> periodS;
197         //Λίστα πουκρατάμετουςκόμβουςπουέχουνγίνει traversed κατάτηδιαγραφή
198         //ενός κόμβουαπότοδένδρο .
199         SimplyLinkedList<Node> travDelPath;
200
201         /**
202          * Προσθέτειένανέοκόμβοστηλίστα .
203          *
204          * @param data Ταδεδομέναπουθακρατάειονέοςκόμβος .
205          * @see Node#Node(Flights)
206          * @see Node#setMaxArr(long)
207          * @see Flights#getArrTime()
208          * @see SimplyLinkedList#SimplyLinkedList()
209          * @see BinarySearchTree#add(Node, Flights)
210          * @see SimplyLinkedList#getLength()
211          * @see SimplyLinkedList#removeHead()
212          * @see Node#getMaxArr()
213          */
214         public void add(Flights data){
215             //Αν δενυπάρχειήδηκόμβοςστοδένδροκαιταδεδομέναδενείναι null
216             if(root==null && data!=null){
217                 //Φτιάχνουμε έναν έοκόμβοκαιτονθέτουμεωςρίζατουδένδρου .
218                 root=new Node(data);
219                 //Ορίζουμε τοπεδίο MaxArr
220                 root.setMaxArr(data.getArrTime());
221                 size++;
222             //Αλλιώς ανταδεδομέναδενείναι null
223             }else if(data!=null){
224                 //Δημιουργούμε ένα instance μιαςμονάσυνδεδεμένηςλίσταπου
225                 //τη χρησιμοποιούμεως stack
226                 travAddPath=new SimplyLinkedList<Node>();
227                 root=add(root,data);
228
229                 //Η ώραάφιξηςτουκαινούργιουκόμβου
230                 long curArr=data.getArrTime();
231                 //Όσο υπάρχουνκόμβοιστηλίστα
232                 while(travAddPath.getLength()!=0){
233                     Node index=travAddPath.removeHead();
234                     long indexMax=index.getMaxArr();
235                     //Αν τοπεδίο MaxArr τουκαινούργιουκόμβουείναιμεγαλύτερο
236                     //από τοαντίστοιχοπεδίοτουκόμβουαπότηλίστα , τότετο
237                     //αντικαθιστούμε.
238                     if(curArr>indexMax){
239                         index.setMaxArr(curArr);
240                     }else{
241                         break;
242                     }
243                 }
244             }
245         }
246
247         /**
248          * Μέθοδοςπουπραγματοποιείόλεςτιςαπαραίτητεςενέργειεςγιανα
249          * τοποθετηθείστησωστήθέσηέναςνέοςκόμβος .
250          *
251          * @param index Οτρέχωνκόμβος – οκόμβοςπουκάνειτιςσυγκρίσεις .
252          * @param data Ταδεδομέναπουθαποθηκεύσειονέοςκόμβος .
253          * @return Τοντρέχωνκόμβο .
254          * @see Node#Node(Node)
255          * @see Node#Node(Flights)
256          * @see SimplyLinkedList#addHead(Object)
257          * @see Node#getData()
258          * @see Flights#getDepTime()
259          * @see Node#getLeftNode()
```

```
260             * @see Node#setLeftNode(Node)
261             * @see Node#setMaxArr(long)
262             * @see Node#getRightNode()
263             * @see Node#setRightNode(Node)
264             */
265            private Node add(Node index,Flights data){
266                    Node indexNode=new Node(index);
267                    //Προσθέτει τοντρέχωνκόμβοστηλίσταμετους traversed κόμβους
268                    travAddPath.addHead(indexNode);
269                    long compareR=indexNode.getData().getDepTime()-data.getDepTime();
270                    //Αν ηδιαφοράείναιμηδέντότεοκόμβοςυπάρχειήδη .
271                    if(compareR==0)
272                            return indexNode;
273                    //Αν ηδιαφοράείναιμεγαλύτερηαπότομηδέν
274                    if(compareR>0){
275                            //Αν τοαριστερόπαιδίδενείναι null
276                            if(indexNode.getLeftNode()!=null){
277                                    //Καλείται πάλιηίδιαμέθοδοςμε index node τοαριστερό
278                                    //παιδί τουτρέχοντακόμβου
279                                    indexNode.setLeftNode(add(indexNode.getLeftNode(),data));
280                            //Αν τοαριστερόπαιδίείναι null
281                            }else{
282                                    //Δημιουργεί ένανέοκόμβοκαιτονορίζειωςαριστερόπαιδί
283                                    indexNode.setLeftNode(new Node(data));
284                                    //Ορίζει τοπεδίο MaxArr
285                                    indexNode.getLeftNode().setMaxArr(data.getArrTime());
286                                    //Αυξάνει τομέγεθοςτηςλίστας
287                                    size++;
288                            }
289                    //Αν ηδιαφοράείναιμικρότερηαπότομηδέν , ακολουθείταιαντίστοιχη
290                    //διαδικασία μετηνπαραπάνω .
291                    }else if(compareR<0){
292                            if(indexNode.getRightNode()!=null){
293                                    indexNode.setRightNode(add(indexNode.getRightNode(),data));
294                            }else{
295                                    indexNode.setRightNode(new Node(data));
296                                    //Set MaxArr
297                                    indexNode.getRightNode().setMaxArr(data.getArrTime());
298                                    size++;
299                            }
300                    }
301                    return indexNode;
302            }
303            /**
304             * Δοθέντοςμιαςημερομηνίαςαναχώρησηςβρίσκεισεποιοκόμβοανήκει .
305             *
306             * @param index Ημερομηνίααναχώρησηςπροςαναζήτηση .
307             * @return Τονκόμβοστονοποίοανήκειτο index.
308             * @see Node#getData()
309             * @see Flights#getDepTime()
310             * @see Node#getLeftNode()
311             * @see Node#getRightNode()
312             */
313            public Node getNode(long index){
314                    if(root==null)
315                            return null;
316                    Node indexNode=root;
317                    long compareR;
318                    //Παίρνουμε τηδιαφοράτηςημερομηνίαςαναχώρησηςτουτρέχοντακόμβου
319                    //με τηνημερομηνίααναχώρησηςπουψάχνουμε
320                    //Όσο ηδιαφοράαυτήδενείναιμηδέν
321                    while((compareR=indexNode.getData().getDepTime()-index)!=0){
322                            //Αν ηδιαφοράείναιμεγαλύτερηαπότομηδέν
323                            if(compareR>0){
324                                    //Αν τοαριστερόπαιδίδενείναι null
325                                    if(indexNode.getLeftNode()!=null){
326                                            //Θέτουμε ωςτρέχοντακόμβοτοαριστερόπαιδί
327                                            indexNode=indexNode.getLeftNode();
```

```
328                             }else{
329                                     return null;
330                             }
331                     //Αν ηδιαφοράείναιμικρότερηαπότομηδέν        , ακολουθούμεαντίστοιχη
332                     //διαδικασία
333                     }else if(compareR<0){
334                             if(indexNode.getRightNode()!=null){
335                                     indexNode=indexNode.getRightNode();
336                             }else{
337                                     return null;
338                             }
339                     }
340             }
341             return indexNode;
342     }
343     /**
344      * Διαγράφειένακόμβοαπότηλίστα      .
345      *
346      * @param data Ταδεδομέναπροςδιαγραφή    .
347      * @return true ανδιαγραφείσωστά   , αλλιώς false.
348      * @see SimplyLinkedList#SimplyLinkedList()
349      * @see Node#getData()
350      * @see Flights#getDepTime()
351      * @see Node#Node(Flights)
352      * @see Flights#Flights(String, Date, Date)
353      * @see Node#setLeftNode(Node)
354      * @see Node#remove(Flights, Node, SimplyLinkedList)
355      * @see Node#getLeftNode()
356      * @see BinarySearchTree#getNode(long)
357      * @see Node#getRightNode()
358      * @see Node#getMaxArr()
359      * @see SimplyLinkedList#addHead(Object)
360      * @see SimplyLinkedList#removeHead()
361      * @see SimplyLinkedList#getFirstNode()
362      * @see SNode#getValue()
363      * @see SimplyLinkedList#getLength()
364      * @see Node#setMaxArr(long)
365      */
366     public boolean remove(Flights data){
367             if(root==null){
368                     return false;
369             }else{
370                     //Δημιουργούμε μιαλίσταπουθακρατάειτους      traversed κόμβους
371                     //μέχρι ναφτάσουμεστονκόμβοπουμαςενδιαφέρει
372                     travDelPath=new SimplyLinkedList<Node>();
373                     //Αν οκόμβοςπουμαςενδιαφέρειείναιηρίζα
374                     if(root.getData().getDepTime()==data.getDepTime()){
375                             //Δημιουργούμε ένακενόκόμβο
376                             Node dummyNode=new Node(new Flights("",new Date(0),new Date(0)));
377                             //Ορίζουμε τηρίζαωςαριστερόπαιδίτουκενούκόμβου
378                             dummyNode.setLeftNode(root);
379                             //Διαγράφουμε τονκόμβο
380                             boolean removeRes=root.remove(data, dummyNode, travDelPath);
381                             //Ως ρίζαορίζουμετονέοαριστερόκόμβοπαιδί     -
382                             root=dummyNode.getLeftNode();
383                             return removeRes;
384                     //Αν οκόμβοςπουμαςενδιαφέρειδενείναιηρίζα
385                     }else{
386                             //Βρίσκουμε ποιοςκόμβοςείναιαυτόςπουχαρακτηρίζεταιαπό
387                             //τη συγκεκριμένηώραναναχώρησης
388                             Node delNode=getNode(data.getDepTime());
389                             long tmpMaxArr=0L;
390                             long tmpMaxLArr=0L;
391                             long tmpMaxRArr=0L;
392                             //Παίρνουμε τηντιμήτουπεδίου    MaxArr τωνπαιδιώντου   , ανδεν
393                             //είναι null
394                             if(delNode.getLeftNode()!=null)
395                                     tmpMaxLArr=delNode.getLeftNode().getMaxArr();
```

```
396                             if(delNode.getRightNode()!=null)
397                                     tmpMaxRArr=delNode.getRightNode().getMaxArr();
398                             //Παίρνουμε τημεγαλύτερητιμή
399                             tmpMaxArr=(tmpMaxLArr>=tmpMaxRArr)?tmpMaxLArr:tmpMaxRArr;
400
401                             //Προσθέτουμε στο stack τηρίζα
402                             travDelPath.addHead(root);
403                             //Διαγράφουμε τοκόμβο .
404                             boolean removeRes=root.remove(data, null, travDelPath);
405
406                             //Παίρνουμε τονπατέρατουκόμβουπουμαςενδιαφέρει .
407                             //Είναι οδεύτεροςκόμβοςστο stack
408                             travDelPath.removeHead();
409                             Node parentNode=travDelPath.getFirstNode().getValue();
410                             boolean switchMaxArr=false;
411
412                             if(tmpMaxArr>parentNode.getMaxArr())
413                                     switchMaxArr=true;
414
415                             //Αν το tmpMaxArr είναιμεγαλύτεροαπότοαντίστοιχοπεδίο
416                             //του πατέρατου , τότεγιακάθεένακόμβοστο stack
417                             if(switchMaxArr){
418                                     //Ο πρώτοςκόμβοστου stack θαέχειπάντα MaxArr
419                                     //μικρότερο του tmpMaxArr, εφόσοντο switchMaxArr είναι
420                                     //true
421                                     travDelPath.removeHead().setMaxArr(tmpMaxArr);
422                                     while(travDelPath.getLength()!=0){
423                                             //Αν τοαντίστοιχοπεδίοτουείναιμικρότεροαπό
424                                             //το tmpMaxArr τοαλλάζουμε .
425                                             Node tmpNode=travDelPath.removeHead();
426                                             if(tmpNode.getMaxArr()<tmpMaxArr)
427                                                     tmpNode.setMaxArr(tmpMaxArr);
428                                     }
429                             }
430
431                             return removeRes;
432                     }
433             }
434     }
435     /**
436      * Μετατρέπειτοδυαδικόδένδροαναζήτησηςσελίστασύμφωναμετην
437      * ενδοδιάταξη.
438      *
439      * @return Τοδένδροσελίστα .
440      * @see SimplyLinkedList#SimplyLinkedList()
441      * @see BinarySearchTree#treeToList(Node, SimplyLinkedList)
442      */
443     public SimplyLinkedList<Flights> toList(){
444             //Δημιουργεί μιανέαλίστα
445             SimplyLinkedList<Flights> theList=new SimplyLinkedList<Flights>();
446             treeToList(root,theList);
447             return theList;
448     }
449     /**
450      * Ενδοδιάταξηγιατοδυαδικόδένδροαναζήτησης .
451      * @param indexNode Οτρέχωνκόμβος .
452      * @param theList Ηλίσταστηνοποίααποθηκεύονταιοικόμβοιτουδένδρου .
453      * @see Node#getLeftNode()
454      * @see SimplyLinkedList#addTail(Object)
455      * @see Node#getData()
456      * @see Node#getRightNode()
457      */
458     private void treeToList(Node indexNode, SimplyLinkedList<Flights> theList){
459             //Αν οκόμβοςδενείναι null
460             if(indexNode!=null){
461                     //Αναδρομικό κάλεσματηςίδιαμεθόδου
462                     treeToList(indexNode.getLeftNode(), theList);
463                     //Προσθέτει τονκόμβοστοτέλοςτηςλίστας .
```

```
464                     theList.addTail(indexNode.getData());
465                     treeToList(indexNode.getRightNode(), theList);
466             }
467         }
468         /**
469          * Κάνειαναζήτησησεέναδυαδικόδένδροαναζήτησηςσύμφωναμετον
470          * κωδικόπτήσης .
471          * @param indexNode Οτρέχωνκόμβος .
472          * @param flightCode Οκωδικόςπτήσης .
473          * @see Node#getData()
474          * @see Flights#getFlightCode()
475          * @see Node#getLeftNode()
476          * @see Node#getRightNode()
477          */
478         public void searchNode(Node indexNode, String flightCode){
479             //Αν οτρέχωνκόμβοςδενείναι null
480             if(indexNode!=null){
481                 //Αν οκωδικόςπτήσηςτουτρέχοντακόμβουείναιίδιοςμετον
482                 //κωδικό πτήσηςπουψάχνουμε
483                 if(indexNode.getData().getFlightCode().equals(flightCode))
484                     //Στο field resultNode βάζουμετοντρέχοντακόμβο
485                     resultNode=indexNode;
486                 //Αναδρομικό κάλεσματηςμεθόδου
487                 searchNode(indexNode.getLeftNode(), flightCode);
488                 searchNode(indexNode.getRightNode(), flightCode);
489             }
490         }
491         /**
492          * Διαγράφειένακόμβοσύμφωναμετονκωδικόπτήσης .
493          *
494          * @param flightCode Οκωδικόςπτήσης .
495          * @see BinarySearchTree#searchNode(Node, String)
496          * @see Node#getData()
497          * @see BinarySearchTree#remove(Flights)
498          */
499         public void delFlight(String flightCode){
500             //Βρίσκει ποιοςκόμβοςέχειτοσυγκεκριμένοκωδικόπτήσης
501             searchNode(root,flightCode);
502             //Παίρνει ταδεδομένααπότοκόμβο
503             Flights delFlight=resultNode.getData();
504             //Διαγράφει τοκόμβομετασυγκεκριμέναδεδομέναααπότοδένδρο .
505             remove(delFlight);
506         }
507         /**
508          * Δοθέντοςενόςχρονικούδιαστήματοςεπιστρέφειτιςπτήσειςπουείναι
509          * σεεξέλιξηστοδιάστημααυτό .
510          *
511          * @param startTime Αρχήτουδιαστήματοςσεμορφή YYYY:MM:DD:HH:MM
512          * @param finishTime Τέλοςτουδιαστήματοςσεμορφή YYYY:MM:DD:HH:MM
513          * @return Μιαλίσταμετιςπτήσειςπουείναισεεξέλιξη .
514          * @see SimplyLinkedList#SimplyLinkedList()
515          * @see BinarySearchTree#searchPeriodArr(Node, Date, Date)
516          * @see BinarySearchTree#searchPeriodDep(Node, Date, Date)
517          */
518         public SimplyLinkedList<Flights> searchPeriod(Date startTime, Date finishTime){
519             //Δημιουργούμε μίανέαμονάσυνδεδεμένηλίστα
520             periodS=new SimplyLinkedList<Flights>();
521             searchPeriodDep(root, startTime, finishTime);
522             searchPeriodArr(root, startTime, finishTime);
523
524             return periodS;
525         }
526         /**
527          * Ψάχνειγιαπτήσειςστοδένδροπουέχουνημερομηνίααναχώρησηςμέσασεένα
528          * συγκεκριμένοδιάστημα .
529          *
530          * @param indexNode Οτρέχωνκόμβος .
531          * @param startTime Ηαρχήτουδιαστήματοςαναζήτησης .
```

```
532             * @param finishTime Τοτέλοςτουδιαστήματοςαναζήτησης      .
533             * @see Node#getData()
534             * @see Flights#getDepartureTime()
535             * @see SimplyLinkedList#addTail(Object)
536             */
537            private void searchPeriodDep(Node indexNode, Date startTime, Date finishTime){
538                    //Αν οτρέχωνκόμβοςδενείναι      null
539                    if(indexNode!=null){
540                            //Παίρνουμε τηνημ . αναχ. τουτρέχοντακόμβου
541                            long indexDep=indexNode.getData().getDepartureTime().getTime();
542                            //Αν ηημ . αναχ. τουτρέχοντακόμβουείναιμεταξύτουσυγκεκριμένου
543                            //διαστήματος
544                            if(indexDep>=startTime.getTime() && indexDep<=finishTime.getTime()){
545                                    //Προσθέτουμε τονκόμβοστηλίστα
546                                    periodS.addTail(indexNode.getData());
547                            }
548
549                            //Αν ηημ . αναχ. τουτρέχοντακόμβουείναιμεγαλύτεραπότοτέλος
550                            //του διαστήματος, τότεκάνουμεαναζήτησητουαριστερούυποδένδρου
551                            if(indexDep>=finishTime.getTime()){
552                                    searchPeriodDep(indexNode.getLeftNode(), startTime, finishTime);
553                            //Αν ηημ . αναχ. είναιμικρότερητουτέλουςτουδιαστήματοςκαι
554                            //μεγαλύτερη ήίσηαπότηναρχήτότεσυνεχίζουμετηναναζήτηση
555                            //στο υπόλοιποδένδρο
556                            }else if(indexDep<finishTime.getTime() && indexDep>=startTime.getTime()){
557                                    searchPeriodDep(indexNode.getLeftNode(), startTime, finishTime);
558                                    searchPeriodDep(indexNode.getRightNode(), startTime, finishTime);
559                            }
560                    }
561            }
562            /**
563             * Ψάχνειγιαπτήσειςστοδένδροπουέχουνημερομηνίαάφιξηςμέσασεένα
564             * συγκεκριμένοδιάστημα .
565             *
566             * @param indexNode Οτρέχωνκόμβος   .
567             * @param startTime Ηαρχήτουδιαστήματος    .
568             * @param finishTime Τοτέλοςτουδιαστήματος    .
569             * @see Node#getData()
570             * @see Flights#getArrivalTime()
571             * @see SimplyLinkedList#contains(Object)
572             * @see SimplyLinkedList#addTail(Object)
573             */
574            private void searchPeriodArr(Node indexNode, Date startTime, Date finishTime){
575                    //Αν οτρέχωνκόμβοςδενείναι      null
576                    if(indexNode!=null){
577                            //Παίρνουμε τηνημ . αφ. τουτρέχοντακόμβου
578                            float indexArr=indexNode.getData().getArrivalTime().getTime();
579                            //Αν ηημ . αφ. τουτρέχοντακόμβουείναιμεταξύτουδιαστήματος
580                            if(indexArr>=startTime.getTime() && indexArr<=finishTime.getTime()){
581                                    //Αν δενυπάρχειήδηστηλίστα
582                                    if(!periodS.contains(indexNode.getData()))
583                                            //Προσθέτουμε τοντρέχοντακόμβοστηλίστα
584                                            periodS.addTail(indexNode.getData());
585                            }
586                            //Καλούμε αναδρομικάτημέθοδογιαόλοτοδένδρο
587                            searchPeriodArr(indexNode.getLeftNode(), startTime, finishTime);
588                            searchPeriodArr(indexNode.getRightNode(), startTime, finishTime);
589                    }
590            }
591            /**
592             * Μέθοδοςπουτυπώνειτοδυαδικόδένδροσεενδοδιάταξη
593             *
594             * @return Τοδένδροσεενδοδιάταξη    .
595             * @see BinarySearchTree#toList()
596             * @see SimplyLinkedList#toString()
597             */
598            @Override
599            public String toString(){
```

```
600                     SimplyLinkedList<Flights> theList=toList();
601                     StringBuilder listBuild=new StringBuilder();
602                     listBuild.append(theList);
603
604                     return listBuild.toString();
605             }
606     }
```

5.3.2 SimplyLinkedList.java

```
1   package structure;
2
3   /**
4    * Κλάσηπουαντιπροσωπεύειένακόμβοστημονάσυνδεδεμένηλίστα .
5    * Κρατάειπληροφορίεςγιατηντιμήτουκόμβουκαιγιατονεπόμενοκόμβο .
6    *
7    * @param <E> Οτύποςτωνδεδομένωνπουθααποθηκεύονταιστονκόμβο .
8    */
9   class SNode<E>{
10          private E value;
11          private SNode<E> nextNode;
12
13          /**
14           * Constructor μετονοποίοορίζουμετηντιμήτουκόμβου .
15           *
16           * @param value Ητιμήτουκόμβου .
17           */
18          SNode(E value){
19                  this.value=value;
20          }
21
22          /**
23           * Ορίζουμετονεπόμενοκόμβοστηλίστα .
24           *
25           * @param nextNode Οεπόμενοςκόμβοςστηλίστα .
26           */
27          void setNextNode(SNode<E> nextNode){
28                  this.nextNode=nextNode;
29          }
30
31          /**
32           * Παίρνουμετηντιμήτουκόμβου .
33           *
34           * @return Τηντιμήτουκόμβου .
35           */
36          E getValue(){
37                  return value;
38          }
39
40          /**
41           * Παίρνουμετονεπόμενοκόμβοαπότηλίστα .
42           *
43           * @return Τονεπόμενοκόμβοαπότηλίστα .
44           */
45          SNode<E> getNextNode(){
46                  return nextNode;
47          }
48  }
49
50  /**
51   * Κλάσηπουυλοποιείτημονάσυνδεδεμένηλίστα .
52   *
53   * @param <E> Οτύποςδεδομένωνπουθαχειρίζεταιηλίστα .
54   */
55  public class SimplyLinkedList<E> {
56          //Ο πρώτοςκόμβος
```

```
57          private SNode<E> head;
58          //Το μέγεθοςτηςλίστας
59          private int length;
60
61          /**
62           * Constructor μετονοποίοαρχικοποιούμετονπρώτοκόμβο
63           * σε null καιτομέγεθοςτηςλίστας σεμηδέν .
64           */
65          public SimplyLinkedList(){
66                  head=null;
67                  length=0;
68          }
69
70          /**
71           * Τουδίνουμεέναδείκτηκαιπαίρνουμετονσυγκεκριμένοκόμβοχωρίς
72           * νατονδιαγράψουμεαπότηλίστα .
73           *
74           * @param index Οδείκτηςτουκόμβουπουμαςενδιαφέρει .
75           * @return Οκόμβοςπουμαςενδιαφέρεισύμφωναμετοδείκτη .
76           * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
77           * μικρότεροςτουμηδένήμεγαλύτεροςτουμεγέθουςτηςλίστας .
78           * @see SNode#getNextNode()
79           */
80          public SNode<E> getNode(int index) throws IndexOutOfBoundsException{
81                  if(index<0 || index>length){
82                          System.err.println("Index out of bounds");
83                          throw new IndexOutOfBoundsException();
84                  }else{
85                          SNode<E> cursor=head;
86                          for(int i=0;i<index;i++)
87                                  cursor=cursor.getNextNode();
88
89                          return cursor;
90                  }
91          }
92
93          /**
94           * Προσθέτειένανέοκόμβοστοτέλοςτηςλίστας .
95           *
96           * @param value Ητιμήτουνέουκόμβου .
97           * @see SNode#SNode(Object)
98           * @see SNode#setNextNode(SNode)
99           * @see SimplyLinkedList#getNode(int)
100          */
101         public void addTail(E value){
102                 if(length==0){
103                         //Αν τομέγεθοςείναιμηδέντότε
104                         //ο νέοςκόμβοςγίνεταικαιηαρχήτηςλίστας
105                         head=new SNode<E>(value);
106                         head.setNextNode(null);
107                 }else{
108                         SNode<E> newNode=new SNode<E>(value);
109                         SNode<E> cursor=getNode(length-1);
110                         newNode.setNextNode(null);
111                         cursor.setNextNode(newNode);
112                 }
113                 length++;
114         }
115
116         /**
117          * Προσθέτειένανέοκόμβοστηναρχήτηςλίστας .
118          *
119          * @param value Ητιμήτουνέουκόμβου .
120          * @see SNode#SNode(Object)
121          * @see SNode#setNextNode(SNode)
122          */
123         public void addHead(E value){
124                 if(length==0){
```

```
125                     //Αν τομέγεθοςείναιμηδέντότε
126                     //ο νέοςκόμβοςγίνεταικαιηαρχήτηςλίστας
127                     head=new SNode<E>(value);
128                     head.setNextNode(null);
129             }else{
130                     SNode<E> newNode=new SNode<E>(value);
131                     newNode.setNextNode(head);
132                     head=newNode;
133             }
134             length++;
135     }
136
137     /**
138      * Προσθέτειένανέοκόμβοστοενδιάμεσοτηςλίστας
139      * σύμφωναμετονδείκτη .
140      * @param index Ηθέσηπουθέλουμεναβάλουμετονκόμβο .
141      * @param value Ητιμήτουνέουκόμβου .
142      * @see SimplyLinkedList#addHead(Object)
143      * @see SimplyLinkedList#addTail(Object)
144      * @see SimplyLinkedList#getNode(int)
145      * @see SNode#SNode(Object)
146      * @see SNode#setNextNode(SNode)
147      * @see SNode#getNextNode()
148      */
149     public void add(int index, E value){
150             if(index==0){
151                     addHead(value);
152             }else if(index==length-1){
153                     addTail(value);
154             }else{
155                     SNode<E> newNode=new SNode<E>(value);
156                     SNode<E> cursor=getNode(index-1);
157                     newNode.setNextNode(cursor.getNextNode());
158                     cursor.setNextNode(newNode);
159                     length++;
160             }
161     }
162
163     /**
164      * Παίρνουμετηντιμήτουπρώτουκόμβουαπότηλίστακαιτοναφαιρούμε .
165      *
166      * @return Τηντιμήτουπρώτουκόμβου .
167      * @throws IndexOutOfBoundsException Σεπερίπτωσηπουηλίσταείναιάδεια .
168      * @see SNode#getValue()
169      * @see SNode#getNextNode()
170      */
171     public E removeHead() throws IndexOutOfBoundsException{
172             if(length==0){
173                     System.err.println("List is empty");
174                     throw new IndexOutOfBoundsException();
175             }else{
176                     E value=head.getValue();
177                     head=head.getNextNode();
178                     length--;
179
180                     return value;
181             }
182     }
183
184     /**
185      * Παίρνουμετηντιμήενόςσυγκεκριμένουκόμβουχωρίςνατον
186      * αφαιρέσουμεαπότηλίστα .
187      *
188      * @param index Ηθέσητουκόμβουπουμαςενδιαφέρει .
189      * @return Τηντιμήτουκόμβουπουμαςενδιαφέρει .
190      * @see SimplyLinkedList#getNode(int)
191      * @see SNode#getValue()
192      */
```

```
193         public E getNodeValue(int index){
194                 SNode<E> cursor=getNode(index);
195 
196                 return cursor.getValue();
197         }
198 
199         /**
200          * Παίρνουμετηντιμήτουτελευταίουκόμβουστηλίστα
201          * καιτονδιαγράφουμε .
202          *
203          * @return Τηντιμήτουτελευταίουκόμβουστηλίστα .
204          * @throws IndexOutOfBoundsException Σεπερίπτωσηπουηλίστα
205          * είναιάδεια .
206          * @see SimplyLinkedList#getNode(int)
207          * @see SNode#setNextNode(SNode)
208          * @see SNode#getNextNode()
209          * @see SNode#getValue()
210          */
211         public E removeTail() throws IndexOutOfBoundsException{
212                 if(length==0){
213                         System.err.println("List is empty");
214                         throw new IndexOutOfBoundsException();
215                 }else{
216                         SNode<E> tailNode=getNode(length-2);
217                         E value=tailNode.getNextNode().getValue();
218                         tailNode.setNextNode(null);
219                         length--;
220 
221                         return value;
222                 }
223         }
224 
225         /**
226          * Παίρνουμετηντιμήενόςκόμβουπουμαςενδιαφέρεισύμφωναμε
227          * τονδείκτηκαιτοναφαιρούμεαπότηλίστα .
228          *
229          * @param index Ηθέσητουκόμβουπουμαςενδιαφέρει .
230          * @return Τηντιμήτουκόμβουπουμαςενδιαφέρει .
231          * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
232          * μικρότεροςτουμηδένήμεγαλύτεροςτουμεγέθουςτηςλίστας .
233          * @see SimplyLinkedList#removeHead()
234          * @see SimplyLinkedList#removeTail()
235          * @see SimplyLinkedList#getNode(int)
236          * @see SNode#getNextNode()
237          * @see SNode#getValue()
238          * @see SNode#setNextNode(SNode)
239          */
240         public E removeNode(int index) throws IndexOutOfBoundsException{
241                 if(index<0 || index>length){
242                         System.err.println("Index out of bounds");
243                         throw new IndexOutOfBoundsException();
244                 }else if(index==0){
245                         return removeHead();
246                 }else if(index==length-1){
247                         return removeTail();
248                 }else{
249                         SNode<E> tmpNode=getNode(index-1);
250                         E value=tmpNode.getNextNode().getValue();
251                         tmpNode.setNextNode(tmpNode.getNextNode().getNextNode());
252                         length--;
253 
254                         return value;
255                 }
256         }
257 
258         /**
259          * Παίρνουμετονπρώτοκόμβοαπότηλίσταχωρίςνατονδιαγράψουμε .
260          *
```

```
261             * @return Τονπρώτοκόμβοαπότηλίστα      .
262             * @see SimplyLinkedList#getNode(int)
263             */
264            public SNode<E> getFirstNode(){
265                    return getNode(0);
266            }
267
268            /**
269             * Παίρνουμετοντελευταίοκόμβοαπότηλίσταχωρίςνατονδιαγράψουμε              .
270             *
271             * @return Τοντελευταίοκόμβοαπότηλίστα      .
272             * @see SimplyLinkedList#getNode(int)
273             */
274            public SNode<E> getLastNode(){
275                    return getNode(length-1);
276            }
277
278            /**
279             * Παίρνουμετομέγεθοστηςλίστας     .
280             *
281             * @return Τομέγεθοστηςλίστας    .
282             */
283            public int getLength(){
284                    return length;
285            }
286
287            /**
288             * Ελέγχουμεανηλίσταείναιάδεια      .
289             *
290             * @return true ανηλίσταείναιάδεια     , false ανηλίσταέχεικόμβους     .
291             */
292            public boolean isEmpty(){
293                    return length==0?true:false;
294            }
295
296            public boolean contains(E index){
297                    SNode<E> tmpNode=head;
298                    boolean exists=false;
299                    while(tmpNode.getNextNode()!=null){
300                            if(tmpNode.getValue().equals(index))
301                                    exists=true;
302                            tmpNode=tmpNode.getNextNode();
303                    }
304
305                    return exists;
306            }
307            /**
308             * Επεστρέφειτιςτιμέςκάθεκόμβουστηλίστα       .
309             *
310             * @return Τιςτιμέςκάθεκόμβουστηλίστα     .
311             * @see SNode#getValue()
312             * @see SNode#getNextNode()
313             */
314            @Override
315            public String toString(){
316                    if(length==0){
317                            return "List is empty";
318                    }else{
319                            StringBuilder sb=new StringBuilder();
320                            sb.append("\n");
321                            SNode<E> tmpNode=head;
322                            while(tmpNode.getNextNode()!=null){
323                                    sb.append(tmpNode.getValue()).append(" - ");
324                                    tmpNode=tmpNode.getNextNode();
325                            }
326                            sb.append(tmpNode.getValue());
327                            sb.append("\n");
328
```

```
329                             return sb.toString();
330                     }
331             }
332     }
```