Πανεπιστήμιο Πειραιώς
Τμήμα Πληροφορικής

Εργασία για το μάθημα “Δομές Δεδομένων”

Σύστημα Κράτησης Αεροπορικών Εισιτηρίων

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

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

Πειραιάς, Πέμπτη 5 Μαϊου 2011

Περιεχόμενα

1 Εισαγωγή **3**

2 Σχεδίαση **3**
2.1 business package 3
2.2 entities package 4
2.3 structures package 4
2.4 Δομή Εφαρμογής 5

3 Επεξήγηση Λειτουργίας **5**
3.1 View Available Flights 5
3.2 Add Flight 5
3.3 Remove Flight 5
3.4 Add Passenger 6
3.5 List Passengers 6
3.6 Delete Passenger 6
3.7 Exit 7

4 Περιεχόμενα CD **7**

5 Πηγαίος Κώδικας **7**
5.1 business 8
5.1.1 FlightsBusiness.java 8
5.1.2 Main.java 15
5.1.3 PassengerBusiness.java 19
5.1.4 Utilities.java 22
5.2 entities 23
5.2.1 Flight.java 23
5.2.2 Passenger.java 26
5.3 structures 28
5.3.1 DoublyLinkedList.java 28
5.3.2 FifoQueue.java 32
5.3.3 SimplyLinkedList.java 36

1 Εισαγωγή

Η εφαρμογή υλοποιεί ένα απλοποιημένο σύστημα κράτησης αεροπορικών εισιτηρίων Είναι γραμμένο στη γλώσσα προγραμματισμού Java. Οι δομές δεδομένων που χρησιμοποιήθηκαν δεν είναι οι έτοιμες δομές που προσφέρει η γλώσσα. Η εφαρμογή υποστηρίζει την προβολή, εισαγωγή και διαγραφή μιας πτήσης και τις αντίστοιχες για τις κρατήσεις των επιβατών. Δεν χρησιμοποιεί κάποια τεχνολογία για persistance καθώς δεν απαιτούνταν από την εργασία.

2 Σχεδίαση

Ο κώδικας της εφαρμογής είναι χωρισμένος σε τρία βασικά κομμάτια. Το κάθε κομμάτι περιέχει κώδικα "ανεξάρτητο" από τα άλλα δύο. Παρόλα αυτά δεν είναι κανένα αυτόνομο. Τα τρία directories–packages είναι τα *business, entities* και *structures*. Το πρώτο package περιέχει όλο το business logic της εφαρμογής. Δηλαδή περιέχει κλάσεις και μεθόδους που προσθέτει κάποιος μια νέα πτήση, νέο επιβάτη κτλ. Το δεύτερο πακέτο αποτελείται από τις δύο οντότητες της εφαρμογής, τους επιβάτες και τις πτήσεις. Οι κλάσεις αυτές είναι υπεύθυνες για την αποθήκευση κρίσιμων πληροφοριών όπως Όνομα, Επίθετο κτλ για τους επιβάτες και αριθμό πτήσης, ώρα αναχώρησης κτλ για τις πτήσεις. Το τρίτο και τελευταίο κομμάτι έχει τις δομές δεδομένων που χρησιμοποιούμε στην εφαρμογή. Συγκεκριμένα έχουμε υλοποιήσει μονά συνδεδεμένη λίστα, διπλά συνδεδεμένη λίστα και ουρά αναμονής. Αναλυτικές πληροφορίες σχετικά με την υλοποίηση και την εφαρμογή τους υπάρχουν στα σχόλια του κώδικα καθώς και στο Javadoc της εφαρμογής.

2.1 business package

Το πακέτο αυτό περιέχει τις κλάσεις *FlightsBusiness, Main, PassengerBusiness* και *Utilities*.

Η πρώτη κλάση έχει μεθόδους για τη δουλεία που γίνεται στη μεριά των πτήσεων. Συγκεκριμένα διαχειρίζεται τη προ–φόρτωση κάποιων πτήσεων στο σύστημα, την προβολή όλων των πτήσεων που είναι αποθηκευμένες στην εφαρμογή, την προσθήκη νέων πτήσεων, τη διαγραφή πτήσεων, τη κράτηση πτήσεων από τους χρήστες, τον έλεγχο της διαθεσιμότητας μιας πτήσης και τον έλεγχο των στοιχείων μιας κράτησης.

Η δεύτερη κλάση αρχικοποιεί τις κατάλληλες δομές, τυπώνει το κεντρικό μενού και αναλόγως την επιλογή καλεί τις κατάλληλες μεθόδους. Οι διαθέσιμες επιλογές του χρήστη είναι η προβολή των διαθέσιμων πτήσεων, η προσθήκη μιας νέας πτήσης στο σύστημα, η διαγραφή μιας πτήσης, η προσθήκη–κράτηση μιας κράτησης από τον χρήστη, η προβολή όλων των επιβατών που έχουν καταχωρηθεί στην εφαρμογή, η διαγραφή ενός επιβάτη και της κράτησής του και η έξοδος από την εφαρμογή.

Τρίτη κλάση είναι η *PassengerBusiness* η όποια είναι υπεύθυνη για τις λειτουργίες που αφορούν τους χρήστες. Χρησιμοποιείται για να εκτύπωση όλων των επιβατών στο σύστημα, την εκτύπωση των κατάλληλων "ερωτήσεων" κατά την εισαγωγή ενός νέου επιβάτη, για την εισαγωγή ενός νέου επιβάτη–κράτησης και για την διαγραφή μιας κράτησης.

Τέλος η κλάση *Utilities* περιέχει δύο βοηθητικές μεθόδους. Η πρώτη είναι μία υλοποίηση της συνάρτησης κατακερματισμού md5 και η άλλη είναι η μετατροπή μιας συμβολοσειράς σε δεκαεξαδική μορφή. Η τελευταία χρησιμοποιείται από την μέθοδο που υλοποιεί το md5 hash.

2.2 entities package

Το πακέτο *entities* περιέχει τις κλάσεις *Flight* και *Passenger*.

Η κλάση *Flight* είναι υπεύθυνη για την αποθήκευση όλων των στοιχείων μιας πτήσης. Συγκεκριμένα τα στοιχεία που κρατάει είναι ο κωδικός πτήσης, την αφετηρία, τον προορισμό, την ώρα αναχώρησης, την ώρα άφιξης, τη τιμή των εισιτηρίων, τον τύπο του αεροσκάφους, των συνολικό αριθμό θέσεων στο αεροπλάνο, τις διαθέσιμες θέσεις στο αεροπλάνο, τους επιβάτες που έχουν κάνει κράτηση για τη συγκεκριμένη πτήση και υπάρχει θέση στο αεροπλάνο και τους επιβάτες οι οποίοι έχουν κάνει κράτηση για τη συγκεκριμένη πτήση αλλά δεν υπάρχουν διαθέσιμες θέσεις οπότε μπαίνουν σε λίστα αναμονής. Οι μέθοδοι που υπάρχουν είναι για να θέτουμε μία νέα τιμή στις παραπάνω μεταβλητές και για να παίρνουμε τις τιμές των παραπάνω μεταβλητών (getters & setters). Επίσης υπάρχει και μία μέθοδος για την εκτύπωση όλων των παραπάνω λεπτομερειών.

Η κλάση αυτή είναι υπεύθυνη για την αποθήκευση των λεπτομερειών ενός χρήστη και της κράτησής του. Τα στοιχεία που κρατάει είναι το επίθετο, το όνομα, τον αριθμό ταυτότητας, την εθνικότητα, τη διεύθυνση, το τηλέφωνο, το μοναδικό αριθμό κράτησης του χρήστη, την κατάσταση της κράτησής του και μία λίστα με τους κωδικούς πτήσης των δρομολογίων που έχει κάνει κράτηση. Οι μέθοδοι που υπάρχουν στη κλάση αυτή είναι για να ορίζουμε μία νέα τιμή στα παραπάνω στοιχεία και για να παίρνουμε την υπάρχουσα τιμή τους. Υπάρχει και μία μέθοδος για την εκτύπωση όλων των παραπάνω στοιχείων.

2.3 structures package

Το πακέτο αυτό περιέχει τις κλάσεις *DoublyLinkedList, FifoQueue* και *SimplyLinkedList*.

Η κλάση *DoublyLinkedList* υλοποιεί τη διπλά συνδεδεμένη λίστα που χρησιμοποιείται στην εφαρμογή. Χρησιμοποιεί τα Generics της Java για τον προσδιορισμό του τύπου των στοιχείων που θα κρατάει. Αποτελείται από δύο κλάσεις, την *DNode* που κρατάει τις πληροφορίες για κάθε κόμβο όπως την τιμή του κόμβου, τον επόμενο και τον προηγούμενο κόμβο καθώς και κάποιες μεθόδους για την προσπέλαση των παραπάνω στοιχείων. Η δεύτερη κλάση είναι η *DoublyLinkedList* που υλοποιεί κάποιες λειτουργίες της διπλά συνδεδεμένης λίστας όπως την διαγραφή ενός κόμβου ενώ παίρνουμε και την τιμή του, την προσθήκη σε κάποια συγκεκριμένη θέση στη λίστα, την προσθήκη ενός κόμβου στην αρχή, στο τέλος, μία μέθοδος για να παίρνουμε το μέγεθος της λίστας, μία για να ξέρουμε αν η λίστα είναι άδεια και μία για να τυπώνουμε όλα τους κόμβους της λίστας.

Η κλάση *FifoQueue* υλοποιεί μία FIFO (First In First Out) ουρά αναμονής βασισμένη σε μία απλά συνδεδεμένη λίστα. Χρησιμοποιεί επίσης τα Generics της Java. Έχει δύο κλάσεις, η μία είναι η *QNode* όπου κρατάει τα στοιχεία ενός κόμβου όπως την τιμή του κόμβου και τον επόμενο κόμβο. Επίσης έχει κάποιες μεθόδους για την προσπέλαση των στοιχείων αυτών. Τέλος η κλάση *FifoQueue* περιέχει μεθόδους για τις βασικές λειτουργίες της ουράς όπως είναι η εισαγωγή στο τέλος, η εξαγωγή από την αρχή, η διαγραφή ενός στοιχείου στο ενδιάμεσο της ουράς, μέθοδος για να παίρνουμε το μέγεθος της ουράς, για να βλέπουμε αν είναι άδεια και μία μέθοδος για να τυπώνει τις τιμές των κόμβων της ουράς αναμονής.

Τέλος 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.

2.4 Δομή Εφαρμογής

Οι δύο οντότητες της εφαρμογής είναι οι πτήσεις και οι επιβάτες–κρατήσεις. Τα χαρακτηριστικά στοιχεία των πτήσεων κρατούνται στην κλάση *Flight*, δηλαδή κάθε πτήση είναι ένα instance της κλάσης αυτής. Κάθε πτήση, εκτός από τα στοιχεία που αναφέρονται παραπάνω, έχει μία λίστα μονά συνδεδεμένη με τους κωδικούς κράτησης και μία ουρά αναμονής με τους κωδικούς κράτησης που ναι μεν έχουν κάνει κράτηση αλλά επειδή δεν υπήρχε διαθέσιμη θέση, μπήκαν σε ουρά αναμονής. Όλες οι πτήσεις, δηλαδή τα instances της κλάσης Flight κρατούνται σε μία διπλά συνδεδεμένη λίστα.

Για κάθε επιβάτη που κάνει επιτυχώς μία κράτηση, κρατούνται τα προσωπικά του στοιχεία καθώς και μία λίστα με τους κωδικούς πτήσεων που έχει κρατήσει θέση. Η λίστα αυτή είναι μία μονά συνδεδεμένη λίστα. Επίσης για κάθε επιβάτη–κράτηση δημιουργείται ένα μοναδικό "κλειδί". Όλοι οι επιβάτες, δηλαδή instances της κλάσης *Passenger* κρατούνται σε μία διπλά συνδεδεμένη λίστα.

3 Επεξήγηση Λειτουργίας

Για να τρέξει η εφαρμογή χρειάζεται το runtime enviroment της Java. Τα binaries είναι αρχειοθετημένα σε ένα archive τύπου jar. Οπότε για να εκτελέσουμε την εφαρμογή αρκεί να πάμε στον κατάλογο που βρίσκεται το αρχείο *DataStruct.jar* και εκτελούμε *java -jar DataStruct.jar*. Εμφανίζεται ένα μενού με τις διαθέσιμες λειτουργίες. Το μενού θα εμφανίζεται έπειτα από κάθε ενέργεια μέχρι να δώσουμε την επιλογή εξόδου.

Επίσης η κλάση *Main* δημιουργεί τις λίστες με τους επιβάτες και τις πτήσεις ώστε να είναι διαθέσιμες στην εφαρμογή.

3.1 View Available Flights

Με την επιλογή αυτή, τυπώνεται όλη η λίστα με τις πτήσεις που είναι καταχωρημένες στην εφαρμογή. Συγκεκριμένα επιλέγοντας την πρώτη επιλογή, καλείται η μέθοδος *listFlights* της κλάσης *FlightsBusiness*. Αυτή με τη σειρά της καλεί την μέθοδο *toString* της κλάσης *DoublyLinkedList* όπου είναι η λίστα που κρατάει τις πτήσεις. Τέλος από αυτή καλείται η *toString* της κλάσης *Flight* που τυπώνει τα χαρακτηριστικά μιας πτήσης.

3.2 Add Flight

Με την δεύτερη επιλογή μπορούμε να προσθέσουμε πτήσεις στο σύστημα. Καλείται η μέθοδος *addFlight* της κλάσης *FlightsBusiness* όπου ρωτάει τον χρήστη ποια θα είναι τα χαρακτηριστικά της νέας πτήσης. Ο κωδικός πτήσης μπορεί να είναι είτε κεφαλαία είτε πεζά. Για την καταχώρηση της ημερομηνίας αναχώρησης και άφιξης χρησιμοποιείται η κλάση *Gregorian Calendar*, η κλάση *Date* και η μέθοδος *getTime* της Java που μετατρέπουν την ημερομηνία σε milliseconds από το EPOCH. Έπειτα δημιουργείται ένα νέο αντικείμενο τύπου *Flight* με τις παραπάνω τιμές και το βάζουμε στη λίστα με τις υπόλοιπες διαθέσιμες πτήσεις μέσω της μεθόδου *addTail* της κλάσης *DoublyLinkedList*.

3.3 Remove Flight

Με την επιλογή αυτή μπορούμε να διαγράψουμε μία διαθέσιμη πτήση. Επιλέγοντάς την, καλείται η μέθοδος *removeFlight* της κλάσης *FlightBusiness*. Μας ρωτάει τον κωδικό πτή-

σης και ψάχνει για το συγκεκριμένο αντικείμενο στη λίστα με τις διαθέσιμες πτήσεις. Αφού το βρει, το διαγράφει.

3.4 Add Passenger

Η επιλογή Add Passenger προσθέτει έναν επιβάτη και μία κράτηση στο σύστημα. Αρχικά καλείται η μέθοδος *askPassenger* της κλάσης *PassengerBusiness* όπου ρωτάει τον χρήστη για τα προσωπικά του στοιχεία και τους κωδικούς πτήσεων των πτώσεων που θέλει να κλείσει. Οι κωδικοί πτήσεων μπορεί να είναι είτε κεφαλαία είτε πεζά. Στη συνέχεια καλείται η μέθοδος *validateFlights* της κλάσης *FlightsBusiness* όπου ελέγχει αν μία κράτηση είναι έγκυρη Αν ο χρήστης έχει κάνει κράτηση μιας πτήσης χωρίς ενδιάμεση στάση τότε προφανώς είναι έγκυρη Σε αντίθετη περίπτωση θα πρέπει ο προορισμός της πρώτης να είναι ίδιος με την αφετηρία της επόμενης και η ώρα άφιξης της πρώτης να είναι πριν την ημερομηνία αναχώρησης της επόμενης. Αυτός ο έλεγχος γίνεται για όσες πτήσεις έχει δώσει ο χρήστης. Αν οι κράτηση βγει άκυρη τότε εκτυπώνεται κατάλληλο μήνυμα και ο χρήστης βγαίνει πάλι στο κεντρικό μενού όπου μπορεί να ξαναπροσπαθήσει.

Σε αντίθετη περίπτωση, καλείται η μέθοδος *addPassenger* της κλάσης *PassengerBusiness*. Η μέθοδος αυτή προσθέτει στη λίστα με τους επιβάτες τον νέο επιβάτη. Επίσης δημιουργεί ένα μοναδικό κλειδί για αυτή την κράτηση ώστε να μπορεί αργότερα να τη διαγράψει, αλλά βοηθάει και στην λειτουργία της εφαρμογής. Το κλειδί είναι το md5sum του αριθμού ταυτότητας του χρήστη και των κωδικών πτήσεων που έχει κάνει κράτηση. Ακολουθεί η μέθοδος *checkAvailability* της κλάσης *FlightsBusiness* με όρισμα το αντικείμενο μιας πτήσης που αντιστοιχεί στον κωδικό πτήσης που έχει δώσει ο χρήστης. Με αυτό τον τρόπο ελέγχεται αν σε κάθε πτήση υπάρχει διαθέσιμη θέση σε κάθε πτήση. Αν έστω και σε μία πτήση δεν υπάρχει διαθέσιμη θέση τότε ο χρήστης μπαίνει στις ουρές αναμονής όλων των πτήσεων. Στη συνέχει καλείται η *bookFlight* της κλάσης *FlightsBusiness* με όρισμα τους κωδικούς πτήσεων που έχει δώσει ο χρήστης, το μοναδικό κλειδί της κράτησης και μία επιλογή με διαφορετική τιμή αναλόγως την διαθεσιμότητα των θέσεων. Όπως αναφέραμε να παραπάνω, αν υπάρχουν διαθέσιμες θέσεις σε όλες τις πτήσεις, τότε ο κωδικός της κράτησης αποθηκεύεται στη λίστα επιβίβασης κάθε πτήσης και μειώνονται οι διαθέσιμες θέσεις. Αν δεν υπάρχουν διαθέσιμες θέσεις τουλάχιστον σε μία πτήση, τότε ο κωδικός της κράτησης αποθηκεύεται στις ουρές αναμονής κάθε πτήσης. Τελειώνοντας μέσω της μεθόδου *setStatus* της κλάσης *Passenger* θέτουμε την διαθεσιμότητα της κράτησης.

3.5 List Passengers

Με την επιλογή List Passengers μπορούμε να δούμε όλους τους επιβάτες που έχουν κάνει κράτηση μέσω της εφαρμογής. Με την επιλογή αυτή καλείται η μέθοδος *listPassengers* της κλάσης *PassengerBusiness*. Αυτή καλεί την μέθοδο *toString* της κλάσης *DoublyLinkedList* όπου τυπώνει την τιμή κάθε αντικειμένου που είναι αποθηκευμένο στη λίστα. Αυτή με τη σειρά της καλεί την μέθοδο *toString* της κλάσης *Passenger* όπου τελικά τυπώνει τα χαρακτηριστικά μιας κράτησης.

3.6 Delete Passenger

Με αυτή την επιλογή ένας επιβάτης μπορεί να διαγράψει την κράτηση που έχει κάνει. Μία κράτηση μπορεί να βρίσκεται σε δύο δυνατές καταστάσεις. Μπορεί να βρίσκεται στις λίστες αναμονής των πτήσεων ή μπορεί να βρίσκεται στις λίστες επιβίβασης των πτήσεων. Αναλόγως την κατάσταση, ακολουθείται διαφορετική λογική και διαδικασία κατά την διαγραφή.

Αν μία κράτηση βρίσκεται σε κατάσταση pending, δηλαδή ο επιβάτες είναι στις ουρές αναμονής των πτήσεων του, τότε αρκεί απλά να τον διαγράψουμε από τις παραπάνω λίστες αναμονής. Οπότε για κάποιο επιβάτη σε αυτή την κατηγορία, παίρνουμε τη λίστα *bookedFlightsList* που κρατάει τις πτήσεις που έχει κάνει κράτηση ο χρήστης και διαγράφουμε από αυτές τον κωδικό κράτησης του (του επιβάτη). Τέλος διαγράφουμε τον επιβάτη και από τη λίστα με τους επιβάτες.

Αν μία κράτηση βρίσκεται σε κατάσταση boarding, δηλαδή ο επιβάτης βρίσκεται στις λίστες επιβίβασης των πτήσεων που έχει κάνει κράτηση, τότε ακολουθείται διαφορετική διαδικασία. Η λογική είναι ότι όταν διαγράφεται ένας χρήστης από μία λίστα επιβίβασης, κοιτάμε στην ουρά αναμονής της αντίστοιχης πτήσης, αν υπάρχει επιβάτης του οποίου η κατάσταση κράτησης μπορεί να γίνει boarding έπειτα από τη διαγραφή του πρώτου, δηλαδή σε όλες τις πτήσεις που έχει κάνει κράτηση υπάρχουν διαθέσιμες θέσεις, τότε τον βάζουμε στις λίστες επιβίβασης των πτήσεων του. Αν δεν βρεθεί τέτοιος επιβάτης μένει κενή μία θέση στην πτήση. Επομένως από τον επιβάτη προς διαγραφή, παίρνουμε τη λίστα με τους κωδικούς πτήσεών του και παίρνουμε την πρώτη πτήση. Από την πρώτη πτήση παίρνουμε τη λίστα αναμονής της. Για κάθε έναν επιβάτη στη λίστα αναμονής ελέγχουμε αν κάθε μία πτήση του έχει διαθέσιμες θέσεις έπειτα από τη διαγραφή του πρώτου επιβάτη. Αν βρεθεί τέτοιος επιβάτης, τον διαγράφουμε από τις ουρές αναμονής των πτήσεων που έχει κάνει κράτηση και τον τοποθετούμε στις λίστες επιβίβασης αυτών των πτήσεων. Τελικά διαγράφουμε τον χρήστη και από τη λίστα με τους επιβάτες.

3.7 Exit

Με την επιλογή αυτή βγαίνουμε από την εφαρμογή. Επειδή δεν έχει υλοποιηθεί κάποιος τρόπος για μόνιμη αποθήκευση, έπειτα από την έξοδο, όποια καταχώρηση είχε γίνει χάνεται.

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.

Documentation.pdf Το παρόν εγχειρίδιο.

DataStruct.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 και εμφανίζει μερικά χωρίς κενά.

5.1 business

5.1.1 FlightsBusiness.java

```
1  package business;
2
3  import java.util.Scanner;
4  import java.util.Date;
5  import java.util.GregorianCalendar;
6
7  import entities.Flight;
8  import entities.Passenger;
9  import structures.DoublyLinkedList;
10 import structures.FifoQueue;
11 import structures.SimplyLinkedList;
12
13 /**
14  * Κλάσημεμεθόδουςγιατηνδιαχείρισητωνπτήσεων
15  * καιότιέχεισχέσημεαυτές .
16  */
17 public class FlightsBusiness {
18         private DoublyLinkedList<Flight> flights;
19         /**
20          * Ορίζειτηλίσταμετιςδιαθέσιμεςπτήσειςγιατην
21          * κλάσηαυτή .
22          *
23          * @param flights Ηλίσταμετιςδιαθέσιμεςκλάσεις .
24          */
25         public void setFlightsList(DoublyLinkedList<Flight> flights){
26                 this.flights=flights;
27         }
28         /**
29          * Παίρνουμετηλίσταμετιςδιαθέσιμεςπτήσεις .
30          *
31          * @return Τηλίσταμετιςδιαθέσιμεςπτήσεις .
32          */
33         public DoublyLinkedList<Flight> getFlightsList(){
34                 return flights;
35         }
36         /**
37          * Μέθοδοςγιατηνφόρτωσηστοσύστημαμερικώνπτήσεων .
38          * Κάνειχρήσητων GregorianCalendar και Date της Java
39          * γιατηναναπαράστασητωνημερομηνιών .
40          * @see DoublyLinkedList#addTail(Object)
41          * @see Flight#Flight(String, String, String, Date, Date, double, String,
42          *  int, int)
43          */
44         public void loadFlights(){
45                 Date departureDate=new GregorianCalendar(2011,04,15,18,15).getTime();
46                 Date arrivalDate=new GregorianCalendar(2011,04,15,20,30).getTime();
47                 //Δημιουργείται ένανέο instance τηςκλάσης Flight γιατην
48                 //αποθήκευση μιαςνέαςπτήσης
49                 Flight flight=new Flight("EZY8567", "Athens", "London", departureDate,
50                                 arrivalDate, 180.50, "Airbus␣320", 100, 5);
51                 //Προσθήκη τηςπαραπάνωπτήσηςστηλίσταμε
52                 //τις διαθέσιμεςπτήσεις
53                 flights.addTail(flight);
54
55                 departureDate=new GregorianCalendar(2011, 04, 16, 15, 00).getTime();
56                 arrivalDate=new GregorianCalendar(2011, 04, 16, 17, 30).getTime();
57                 flight=new Flight("ABC1234", "London", "Dublin", departureDate,
58                                 arrivalDate, 100, "Airbus123", 50, 10);
59                 flights.addTail(flight);
60
61                 departureDate=new GregorianCalendar(2011, 04, 16, 20, 10).getTime();
62                 arrivalDate=new GregorianCalendar(2011, 05, 1, 13, 40).getTime();
63                 flight=new Flight("DEF5678", "Dublin", "Alexandria", departureDate,
```

```
64                              arrivalDate, 200.20, "Airbus 450", 200, 8);
65              flights.addTail(flight);
66
67              departureDate=new GregorianCalendar(2011, 06, 1, 8, 2).getTime();
68              arrivalDate=new GregorianCalendar(2011, 06, 1, 13, 40).getTime();
69              flight=new Flight("ERT1234", "Athens", "Rome", departureDate,
70                              arrivalDate, 200.20, "Airbus 450", 100, 20);
71              flights.addTail(flight);
72      }
73      /**
74       * Τυπώνειόλεςτιςδιαθέσιμεςπτήσεις     .
75       * @see Flight#toString()
76       * @see DoublyLinkedList#toString()
77       */
78      public void listFlights(){
79              //Χρησιμοποιεί τιςμεθόδους  toString γιανατυπώσει
80              //τις πτήσεις
81              System.out.println(flights);
82      }
83      /**
84       * Παίρνειόλαταααπαραίτηταστοιχείααπότον
85       * χρήστηκαιπροσθέτειμίανέαπτήσηστηνεφαρμογή       .
86       * @see Flight#Flight(String, String, String, Date, Date, double, String,
87       *  int, int)
88       * @see DoublyLinkedList#addTail(Object)
89       */
90      public void addFlight(){
91              Scanner in=new Scanner(System.in);
92              //Μαζεύει τααπαραίτηταστοιχείααπό
93              //τον χρήστη
94              System.out.println("Flight Code:");
95              String flightCode=in.nextLine();
96              System.out.println("Starting Point:");
97              String startingPoint=in.nextLine();
98              System.out.println("Destination:");
99              String destination=in.nextLine();
100             System.out.println("Departure Time (yyyy:mm:dd:hh:mm)");
101             String depTime=in.nextLine();
102             System.out.println("Arrival Time (yyyy:mm:dd:hh:mm)");
103             String arTime=in.nextLine();
104             System.out.println("Plane Type:");
105             String planeType=in.nextLine();
106             System.out.println("Total Seats:");
107             int totalSeats=in.nextInt();
108             System.out.println("Available Seats:");
109             int availableSeats=in.nextInt();
110             System.out.println("Ticket Price:");
111             double ticketPrice= in.nextDouble();
112             //Χωρίζει τοέτοςμήναημέραώρα ,,, καιλεπτά
113             //με διαχωριστικότο ':'
114             String[] tmpDTime=depTime.split("[:]");
115             String[] tmpATime=arTime.split("[:]");
116             Date departureTime=new GregorianCalendar(Integer.parseInt(tmpDTime[0]),
117                             Integer.parseInt(tmpDTime[1])-1, Integer.parseInt(tmpDTime[2]),
118                             Integer.parseInt(tmpDTime[3]), Integer.parseInt(tmpDTime[4]))
119             .getTime();
120             Date arrivalTime=new GregorianCalendar(Integer.parseInt(tmpATime[0]),
121                             Integer.parseInt(tmpATime[1])-1, Integer.parseInt(tmpATime[2]),
122                             Integer.parseInt(tmpATime[3]), Integer.parseInt(tmpATime[4]))
123             .getTime();
124             //Μετατρέπει τονκωδικόπτήσηςσεκεφαλαία
125             //για διευκόλυνσητουχρήστη
126             flightCode=flightCode.toUpperCase();
127             //Δημιουργείται ένανέο  instance τηςκλάσης  Flight γιατην
128             //αποθήκευση τηςπτήσης
129             Flight newFlight=new Flight(flightCode, startingPoint, destination,
130                             departureTime, arrivalTime, ticketPrice, planeType, totalSeats,
131                             availableSeats);
```

```
132             //Προσθήκη της πτήσης στην υπάρχουσα λίστα με τις διαθέσιμες πτήσεις
133             flights.addTail(newFlight);
134         }
135         /**
136          * Παίρνει απότον χρήστη τον κωδικό πτήσης
137          * και διαγράφει τη συγκεκριμένη πτήση.
138          * @see FlightsBusiness#searchForFlightCode(String)
139          * @see DoublyLinkedList#removeNode(int)
140          */
141         public void removeFlight(){
142             System.out.println("Give flight code:");
143             Scanner in=new Scanner(System.in);
144             //Διαβάζει τον κωδικό πτήσης από τον χρήστη
145             String flightCode=in.nextLine();
146             //Μετατρέπει τον κωδικό πτήσης σε κεφαλαία
147             //για την διευκόλυνση του χρήστη
148             flightCode=flightCode.toUpperCase();
149             //Ψάχνει στη λίστα με τις διαθέσιμες πτήσεις
150             //σε ποια θέση είναι η πτήση με τον συγκεκριμένο
151             //κωδικό πτήσης
152             int index=searchForFlightCode(flightCode);
153             //Διαγράφει την παραπάνω πτήση
154             flights.removeNode(index);
155         }
156         /**
157          * Δημιουργία μιας νέας κράτησης. Μία κράτηση μπορεί να μπει
158          * σε κάποια λίστα επιβίβασης ή σε κάποια ουρά αναμονής
159          * αναλόγως το κλειδί <i>available.</i>
160          *
161          * @param bookingCode Ο μοναδικός κωδικός κράτησης.
162          * @param flightCode Ο κωδικός πτήσης.
163          * @param available Διαθεσιμότητα πτήσης. Ανείναι <i>true</i> η
164          * κράτηση γίνεται σε κάποια λίστα επιβίβασης. Ανείναι <i>false,</i>
165          * η κράτηση γίνεται σε κάποια ουρά αναμονής.
166          * @return 1 αν η κράτηση έγινε σε κάποια λίστα επιβίβασης και
167          * 0 αν η κράτηση έγινε σε κάποια ουρά αναμονής.
168          * @see FlightsBusiness#searchForFlightCode(String)
169          * @see DoublyLinkedList#getNodeValue(int)
170          * @see Flight#getAvailableSeats()
171          * @see Flight#getBoardedPass()
172          * @see Flight#setAvailableSeats(int)
173          * @see Flight#getWaitingPass()
174          * @see FifoQueue#enqueue(Object)
175          * @see SimplyLinkedList#addTail(Object)
176          */
177         public int bookFlight(String bookingCode, String flightCode,
178                 boolean available){
179             //Ψάχνει στη λίστα με τις διαθέσιμες πτήσεις
180             //σε ποια θέση είναι η πτήση με τον συγκεκριμένο
181             //κωδικό πτήσης
182             int index=searchForFlightCode(flightCode);
183             //Παίρνει από τη λίστα τον συγκεκριμένο κόμβο
184             Flight flight=flights.getNodeValue(index);
185             //Οι διαθέσιμες θέσεις της συγκεκριμένης πτήσης
186             int availableSeats=flight.getAvailableSeats();
187             if(available){
188                 //Αν το <i>available</i> είναι <i>true</i>
189                 //Από τη συγκεκριμένη πτήση, παίρνουμε τη λίστα
190                 //επιβίβασης και προσθέτουμε τον κωδικό της κράτησης
191                 flight.getBoardedPass().addTail(bookingCode);
192                 //Μειώνουμε κατά ένα τις διαθέσιμες θέσεις της
193                 //συγκεκριμένης πτήσης
194                 flight.setAvailableSeats(--availableSeats);
195                 return 1;
196             }else{
197                 //Αν το <i>available</i> είναι <i>false</i>
198                 //Παίρνουμε την ουρά αναμονής της συγκεκριμένης
199
```

```
200                     //πτήσης καιπροσθέτουμετονκωδικότηςκράτησης
201                     flight.getWaitingPass().enqueue(bookingCode);
202                     return 0;
203             }
204         }
205         /**
206          * Ελέγχειανυπάρχουνδιαθέσιμεςθέσειςσεμίαπτήση .
207          *
208          * @param flightCode Οκωδικόςπτήσης .
209          * @return true ανυπάρχουνδιαθέσιμεςθέσεις , false
210          * ανδενυπάρχουν .
211          * @see FlightsBusiness#searchForFlightCode(String)
212          * @see DoublyLinkedList#getNodeValue(int)
213          * @see Flight#getAvailableSeats()
214          */
215         public boolean checkAvailability(String flightCode){
216             //Ψάχνει στηλίσταμετιςδιαθέσιμεςπτήσεις
217             //σε ποιαθέσηείναιηπτήσημετονσυγκεκριμένο
218             //κωδικό πτήσης
219             int index=searchForFlightCode(flightCode);
220             //Παίρνει απότηλίστατονσυγκεκριμένοκόμβο
221             Flight flight=flights.getNodeValue(index);
222             //Παίρνει τιςδιαθέσιμεςθέσειςτηςσυγκεκριμένηςπτήσης
223             int availableSeats=flight.getAvailableSeats();
224             boolean available;
225             if(availableSeats>=1){
226                     //Αν οιδιαθέσιμεςθέσειςείναι
227                     //μεγαλύτερες απόμηδέν
228                     available=true;
229             }else{
230                     //Αν είναιίσεςμεμηδέν
231                     //Προφανώς δενγίνονταιμικρότερεςαπόμηδέν
232                     available=false;
233             }
234
235             return available;
236         }
237         /**
238          * Ελέγχειότιοικωδικοίπουέδωσεοχρήστηςσυμβαδίζουν .
239          * Ανοχρήστηςδώσειπάνωαπόένακωδικόπτήσηςπτήση ( μεενδιάμεση
240          * στάση) θαπρέπειηώραναναχώρισηςτηςδεύτερηςναείναιπιομετάαπό
241          * τηνώραάφιξηςτηςπρώτηςκαιοπροορισμόςτηςπρώτηςναείναιίδιος
242          * μετοσημείοαναχώρισηςτηςδεύτερης .
243          *
244          * @param codeFlights Οικωδικοίπτήσεωνπουέδωσεοχρήστης .
245          * @return true ανοιδιαδοχικέςπτήσειςσυμβαδίζουν , false ανόχι .
246          * @see FlightsBusiness#searchForFlightCode(String)
247          * @see DoublyLinkedList#getNodeValue(int)
248          * @see Flight#getArrivalTime()
249          * @see Flight#getDestination()
250          * @see Flight#getDepartureTime()
251          * @see Flight#getStartingPoint()
252          */
253         public boolean validateFlights(String[] codeFlights){
254             boolean valid=true;
255             if(codeFlights.length==1){
256                     //Αν δώσειμόνοένακωδικότότεπροφανώςείναισωστός
257                     valid=true;
258             }else{
259                     for(int i=0;i<codeFlights.length-1;i++){
260                             //Για κάθεμίαπτήση
261                             //Βρίσκει σεποιαθέσηστηλίσταμετις
262                             //διαθέσιμες πτήσειςανήκει , καθώςκαιη
263                             //επόμενή της
264                             int fIndex=searchForFlightCode(codeFlights[i]);
265                             int nIndex=searchForFlightCode(codeFlights[i+1]);
266                             //Παίρνει τουςκόμβουςαπότηλίστα
267                             Flight fFlight=flights.getNodeValue(fIndex);
```

```
268                             Flight nFlight=flights.getNodeValue(nIndex);
269                             //Ημνια/ άφιξηςτηςπρώτης
270                             Date fArrTime=fFlight.getArrivalTime();
271                             //Προορισμός τηςπρώτης
272                             String fDest=fFlight.getDestination();
273                             //Ημνια/ αναχώρησηςτηςδεύτερης
274                             Date nDepTime=nFlight.getDepartureTime();
275                             //Σημείο αναχώρησηςτηςδεύτερης
276                             String nStart=nFlight.getStartingPoint();
277                             //Οι ημερομηνίεςείναισε  millisecond από EPOCH
278                             long timeDiff=nDepTime.getTime()-fArrTime.getTime();
279                             if((timeDiff>0) && (fDest.equals(nStart))){
280                                     //Αν ημ. αναχώρησηςτηςδεύτερηςείναι
281                                     //μεγαλύτερη τηςημ . άφιξηςτηςδεύτερης
282                                     //και προορισμόςτηςπρώτηςείναιίδιος
283                                     //με σημείοαναχώρησηςτηςδεύτερης
284                                     valid=true;
285                             }else{
286                                     valid=false;
287                             }
288                     }
289             }
290             return valid;
291     }
292     /**
293      * Διαγράφειμίακράτησηαπότηνουράαναμονήςμιαςπτήσης        .
294      *
295      * @param bookingID Τοαναγνωριστικότηςκράτησης   .
296      * @param flightCode Οκωδικόςπτήσης  .
297      * @see FlightsBusiness#searchForFlightCode(String)
298      * @see DoublyLinkedList#getNodeValue(int)
299      * @see Flight#getWaitingPass()
300      * @see FifoQueue#getLength()
301      * @see FifoQueue#removeNode(int)
302      * @see Flight#setWaitingPass(FifoQueue)
303      */
304     public void delPendingCode(String bookingID, String flightCode){
305             //Ψάχνει στηλίσταμετιςδιαθέσιμεςπτήσεις
306             //σε ποιαθέσηείναιηπτήσημετονσυγκεκριμένο
307             //κωδικό πτήσης
308             int index=searchForFlightCode(flightCode);
309             //Παίρνει απότηλίστατονσυγκεκριμένοκόμβο
310             Flight curFlight=flights.getNodeValue(index);
311             //Παίρνει τηνουράαναμονήςτηςσυγκεκριμένηςπτήσης
312             FifoQueue<String> waitingQueue=curFlight.getWaitingPass();
313             int queueLength=waitingQueue.getLength();
314             for(int i=0;i<queueLength;i++){
315                     //Για κάθεέναναγνωριστικόστηνουρά    ,
316                     //αν είναιίσομεαυτόπουέδωσεοχρήστης       ,
317                     //το διαγράφειαπότηνουρά
318                     if(waitingQueue.getNodeValue(i).equals(bookingID)){
319                             waitingQueue.removeNode(i);
320                     }
321             }
322             //Ορίζει τηννέαουράαναμονήςγιατηνσυγκεκριμένηπτήση
323             curFlight.setWaitingPass(waitingQueue);
324     }
325     /**
326      * Διαγράφειτονχρήστηαπότιςλίστεςεπιβίβασηςτωνκωδικών
327      * πτήσεωνπουέχειδώσει   . Στησυνέχειαφτιάχνειμίαλίσταμε
328      * τουςκωδικούςκρατήσεωνπουείναιστηνουράαναμονήςτηςπρώτης
329      * πτήσηςτουεπιβάτηπουδιαγράφηκε     .
330      *
331      * @param bookingID Τοαναγνωριστικότουεπιβάτηκράτησης   -.
332      * @param bookedFlights Οικωδικοίπτήσεωνπουαντιστοιχούν
333      * στιςκατοχυρομένεςπτήσειςτουεπιβάτη     .
334      * @return Μίαλίσταμετααναγνωριστικάτωνκρατήσεωνπουβρίσκονται
335      * στηνουράαναμονήςτηςπρώτηςπτήσηςτουχρήστη        .
```

```
336         * @see FlightsBusiness#searchForFlightCode(String)
337         * @see DoublyLinkedList#getNodeValue(int)
338         * @see Flight#getBoardedPass()
339         * @see SimplyLinkedList#getNodeValue(int)
340         * @see SimplyLinkedList#removeNode(int)
341         * @see Flight#getAvailableSeats()
342         * @see Flight#setAvailableSeats(int)
343         * @see Flight#getWaitingPass()
344         * @see SimplyLinkedList#SimplyLinkedList()
345         * @see FifoQueue#getNodeValue(int)
346         * @see SimplyLinkedList#addTail(Object)
347         */
348        public SimplyLinkedList<String> delBoardedCodePre(String bookingID,
349                        SimplyLinkedList<String> bookedFlights){
350                for(int i=0;i<bookedFlights.getLength();i++){
351                        //Για κάθεένακωδικόπτήσης
352                        //Ψάχνει στηλίσταμετιςδιαθέσιμεςπτήσεις
353                        //σε ποιαθέσηείναιηπτήσημετονσυγκεκριμένο
354                        //κωδικό πτήσης
355                        int index=searchForFlightCode(bookedFlights.getNodeValue(i));
356                        //Παίρνει απότηλίστατονσυγκεκριμένοκόμβο
357                        Flight curFlight=flights.getNodeValue(index);
358                        //Παίρνουμε τηλίσταεπιβίβασηςτηςσυγκεκριμένηςπτήσης
359                        SimplyLinkedList<String> boardingList=curFlight.getBoardedPass();
360                        for(int j=0;j<boardingList.getLength();j++){
361                                //Για κάθεένακωδικόκράτησης
362                                //Αν είναιίδιοςμετονκωδικόκράτησης
363                                //του χρήστη
364                                if(boardingList.getNodeValue(j).equals(bookingID)){
365                                        //Τον διαγράφουμεαπότηλίστα
366                                        boardingList.removeNode(j);
367                                        //Παίρνουμε τιςδιαθέσιμεςθέσειςτηςσυγκεκριμένηςπτήσης
368                                        int availableSeats=curFlight.getAvailableSeats();
369                                        //Αυξάνουμε κατάέναορίζουμετιςνέεςδιαθέσιμεςθέσεις
370                                        availableSeats++;
371                                        curFlight.setAvailableSeats(availableSeats);
372                                        break;
373                                }
374                        }
375                }
376                //Ψάχνει στηλίσταμετιςδιαθέσιμεςπτήσεις
377                //σε ποιαθέσηείναιηπρώτηπτήσηαπότηλίστα
378                //με τιςκατοχυρομένεςπτήσειςτουεπιβάτηπροςδιαγραφή
379                int index=searchForFlightCode(bookedFlights.getNodeValue(0));
380                //Παίρνει απότηλίστατοσυγκεκριμένοκόμβο
381                Flight curFlight=flights.getNodeValue(index);
382                //Παίρνει τηνουράαναμονήςτηςσυγκεκριμένηςπτήσης
383                FifoQueue<String> waitingQueue=curFlight.getWaitingPass();
384                //Δημιουργεί μίανέαμονάσυνδεδεμένηλίστακαι
385                //αποθηκεύει ταδεδομένατης  waitingQueue σεαυτή
386                SimplyLinkedList<String> queueCodes=new SimplyLinkedList<String>();
387                if(waitingQueue.getLength()>0){
388                        for(int i=0;i<waitingQueue.getLength();i++){
389                                queueCodes.addTail(waitingQueue.getNodeValue(i));
390                        }
391                }
392
393                return queueCodes;
394        }
395        /**
396         * Βρίσκειένανεπιβάτηαπόμιαλίστατουοποίουόλεςοιπτήσεις
397         * έχουνδιαθέσιμεςθέσεις , ανυπάρχεικαικάνεινέακράτησησε
398         * κάθεμίααπόαυτέςαφούτονδιαγράψειαπότιςλίστεςαναμονήςπουήταν .
399         *
400         * @param queuePassengers Λίσταμεεπιβάτεςαπότηνμέθοδο
401         * delBoardedCodePre
402         * @return Τονκωδικόκράτησηςτουεπιβάτηπουέχειδιαθέσιμεςθέσεις
403         * σεόλεςτιςπτήσειςτουαπότηνπαραπάνωλίσταανυπάρχει , αλλιώς null
```

```
404             * @see SimplyLinkedList#getNodeValue(int)
405             * @see Passenger#getBookedFlights()
406             * @see FlightsBusiness#searchForFlightCode(String)
407             * @see DoublyLinkedList#getNodeValue(int)
408             * @see Flight#getAvailableSeats()
409             * @see SimplyLinkedList#getLength()
410             * @see Passenger#getUid()
411             * @see Flight#getWaitingPass()
412             * @see FifoQueue#getNodeValue(int)
413             * @see FifoQueue#removeNode(int)
414             * @see FlightsBusiness#bookFlight(String, String, boolean)
415             */
416            public String delBoardedCodePost(SimplyLinkedList<Passenger>
417            queuePassengers){
418                    int availableSeats;
419                    //Μετρητής πουμετράεισεπόσεςαπότιςπτήσεις
420                    //του εκάστοτεχρήστηυπάρχουνδιαθέσιμεςθέσεις
421                    int availableFlag=0;
422                    //Κωδικός κράτησηςτουεπιβάτηπουαπότηνουράαναμονής
423                    //των πτήσεώντουθαπάειστιςλίστεςεπιβίβασηςτων
424                    //αντίστοιχων πτήσεων
425                    String luckyBookingCode=null;
426                    for(int i=0;i<queuePassengers.getLength();i++){
427                            //Για κάθεχρήστηστηλίστα     queuePassengers
428                            //Παίρνει τηλίσταμετουςκωδικούςπτήσεων
429                            //που έχεικάνεικράτηση
430                            SimplyLinkedList<String> passBookedFlights=queuePassengers
431                            .getNodeValue(i).getBookedFlights();
432                            for(int j=0;j<passBookedFlights.getLength();j++){
433                                    //Για κάθεμίααπότιςπαραπάνωπτήσεις
434                                    //Ψάχνει στηλίσταμετιςδιαθέσιμεςπτήσεις
435                                    //σε ποιαθέσηείναιηπτήσημετονσυγκεκριμένο
436                                    //κωδικό πτήσης
437                                    int index=searchForFlightCode(passBookedFlights
438                                                    .getNodeValue(j));
439                                    //Παίρνει απότηλίστατοσυγκεκριμένοκόμβο
440                                    Flight curFlight=flights.getNodeValue(index);
441                                    //Παίρνει τιςδιαθέσιμεςθέσειςτηςσυγκεκριμένηςπτήσης
442                                    availableSeats=curFlight.getAvailableSeats();
443                                    if(availableSeats>0)
444                                            //Αν υπάρχουνδιαθέσιμεςθέσεις
445                                            //αυξάνει τομετρητήκατάένα
446                                            availableFlag++;
447                            }
448                            if(availableFlag==passBookedFlights.getLength()){
449                                    //Αν ομετρητής  availableFlag είναίσοςμε
450                                    //το πλήθοςτωνπτήσεωνπουέχεικάνεικράτηση
451                                    //ο συγκεκριμένοςχρήστης .
452                                    //Αυτό σημαίνειότισεόλεςτιςπτήσειςυπάρχουν
453                                    //διαθέσιμες θέσεις, άραμπορείναμεταβείστις
454                                    //λίστες επιβίβασηςτωνπτήσεώντου    .
455                                    //Παίρνουμε τονκωδικόκράτησηςαυτούτουεπιβάτη
456                                    luckyBookingCode=queuePassengers.getNodeValue(i).getUid();
457
458                                    for(int j=0;j<passBookedFlights.getLength();j++){
459                                            //Για κάθεμία  pending κράτησητουεπιβάτη
460                                            //Παίρνει τοκωδικόπτήσης
461                                            String flightCode=passBookedFlights.getNodeValue(j);
462                                            //Για κάθεμίααπότιςπαραπάνωπτήσεις
463                                            //Ψάχνει στηλίσταμετιςδιαθέσιμεςπτήσεις
464                                            //σε ποιαθέσηείναιηπτήσημετονσυγκεκριμένο
465                                            //κωδικό πτήσης
466                                            int index=searchForFlightCode(flightCode);
467                                            //Παίρνει απότηλίστατοσυγκεκριμένοκόμβο
468                                            Flight curFlight=flights.getNodeValue(index);
469                                            //Παίρνει τηνουράαναμονήςτηςπτήσης
470                                            FifoQueue<String> pendingList=curFlight.getWaitingPass();
471                                            for(int k=0;k<pendingList.getLength();k++){
```

```
472                                                     //Για κάθεκαταχώρησηστηνουράαυτή
473                                                     if(pendingList.getNodeValue(k).equals(luckyBookingCode))
474                                                             //Αν τοαναγνωριστικόκράτησηςείναιίδιο
475                                                             //με τοαναγνωριστικότουσυγκεκριμένουεπιβάτη
476                                                             //το αφαιρείαπότηνουρά
477                                                             pendingList.removeNode(k);
478                                             }
479                                             //Κάνει νέακράτησηστηνπτήσημετονκωδικόκράτησης
480                                             //του επιβάτηκαιμετο    flag available ναείναι true
481                                             //δηλαδή ότιπρόκειτεγιακράτησηστηλίσταεπιβίβασης
482                                             bookFlight(luckyBookingCode, flightCode, true);
483                                     }
484                                     break;
485                             }
486                     }
487
488                     return luckyBookingCode;
489             }
490             /**
491              * Ψάχνειστηλίσταμετιςδιαθέσιμεςπτήσειςγιατηνπτήσημε
492              * συγκεκριμένοκωδικόπτήσηςκαιεπιστρέφειτηθέσητηςστηλίστα          .
493              *
494              * @param flightCode Οκωδικόςπτήσηςπουμαςενδιαφέρει      .
495              * @return Τηθέσητηςπτήσηςπουμαςενδιαφέρειαπότηλίστα
496              * μετιςδιαθέσιμεςπτήσεις    . Ανδενβρεθείεπιστρέφει    −1.
497              * @see DoublyLinkedList#getLength()
498              * @see DoublyLinkedList#getNodeValue(int)
499              * @see Flight#getFlightCode()
500              */
501             public int searchForFlightCode(String flightCode){
502                     Flight searchFlight;
503                     int index;
504                     boolean found=false;
505                     int listLength=flights.getLength();
506
507                     for(index=1;index<=listLength;index++){
508                             //Για κάθεένακόμβοστηλίστα
509                             //Παίρνει τηντιμήτουκόμβου
510                             searchFlight=flights.getNodeValue(index);
511                             if(searchFlight.getFlightCode().equals(flightCode)){
512                                     //Αν οκωδικόςπτήσηςείναίδιοςμετον
513                                     //κωδικό πτήσηςπουμαςενδιαφέρει
514                                     found=true;
515                                     break;
516                             }
517                     }
518                     if(found==false)
519                             //Αν δενβρεθείεπιστρέφει   −1
520                             index=−1;
521
522                     return index;
523             }
524     }
```

5.1.2 Main.java

```
1   package business;
2
3   import java.util.Scanner;
4
5   import entities.*;
6   import structures.*;
7
8   /**
9    * Κεντρικήκλάσητηςεφαρμογής    , όπουτυπώνειτομενού
```

```
10  * καιανάλογατηνεπιλογήκαλείτιςκατάλληλεςμεθόδους .
11  */
12 public class Main {
13         /**
14          * Αρχικοποιείτηνεφαρμογή , τυπώνειτομενούκαικαλεί
15          * τιςκατάλληλεςμεθόδους .
16          * @param args
17          */
18         public static void main(String[] args) {
19                 //Δημιουργία μιαςδιπλάσυνδεδεμένηςλίσταςγιατην
20                 //αποθήκευση τωνδιαθέσιμωνπτήσεων
21                 DoublyLinkedList<Flight> flights=new DoublyLinkedList<Flight>();
22                 //Δημιουργία αντικειμένουτηςκλάσης FlightBusiness πουείναι
23                 //υπεύθυνη γιατηδιαχείρισητωνπτήσεων .
24                 FlightsBusiness flightB=new FlightsBusiness();
25                 flightB.setFlightsList(flights);
26                 //Φορτώνει κάποιεςπτήσειςστοσύστημα
27                 flightB.loadFlights();
28                 //Δημιουργία μιαςδιπλάδυνδεδεμένηςλίσταςγιατην
29                 //αποθήκευση τωνεπιβατώνκρατήσεων –
30                 DoublyLinkedList<Passenger> passengers=new DoublyLinkedList<Passenger>();
31                 //Δημιουργία αντικειμένουτηςκλάσης PassengerBusiness πουείναι
32                 //υπεύθυνη γιατηδιαχείρισητωνεπιβατών
33                 PassengerBusiness passB=new PassengerBusiness();
34                 passB.setPassengersList(passengers);
35
36                 Scanner inM=new Scanner(System.in);
37                 //Κλειδί γιατηνεπαναλαμβανόμενηεκτύπωσητουκεντρικούμενού
38                 boolean running=true;
39                 System.out.println("Welcome to the airline booking system");
40                 System.out.println("");
41                 while(running){
42                         System.out.println("Please make your choice");
43                         System.out.println("<--------------------->");
44                         System.out.println("1 -- View available flights");
45                         System.out.println("2 -- Add Flight");
46                         System.out.println("3 -- Delete Flight");
47                         System.out.println("4 -- Add Passenger");
48                         System.out.println("5 -- List Passengers");
49                         System.out.println("6 -- Delete Passenger");
50                         System.out.println("0 -- Exit");
51                         int choice=inM.nextInt();
52
53                         switch (choice) {
54                         case 1:
55                                 //Προβολή διαθέσιμωνπτήσεων
56                                 flightB.listFlights();
57                                 break;
58                         case 2:
59                                 //Προσθήκη νέαςπτήσης
60                                 flightB.addFlight();
61                                 break;
62                         case 3:
63                                 //Διαγραφή μιαςπτήσης
64                                 flightB.removeFlight();
65                                 break;
66                         case 4:
67                                 //Προσθήκη μιαςνέαςκράτησης
68                                 //Συμπληρώνει οχρήστηςτααπαραίτηταστοιχεία
69                                 //και επιστρέφεταισαν String array οικωδικοί
70                                 //πτήσεων πουέχειδώσει
71                                 String[] codeFlights=passB.askAddPassenger();
72                                 //Σύμφωνα μετουςκωδικούςπτήσωνελέγχεταιανη
73                                 //κράτηση είναιέγγυρη . Σχετικάμετηνεγγυρότητα
74                                 //δείτε τηνπαρακάτωμέθοδο . Ανείναιέγγυρηεπιστρέφει
75                                 //true αλλιώς false
76                                 boolean valid=flightB.validateFlights(codeFlights);
77                                 if(valid){
```

```
78                                      //Αν είναιέγγυρη
79                                      //Προσθέτει τονχρήστηστηλίσταμετουςεπιβάτες      ,
80                                      //επιστρέφει τομοναδικόαναγνωριστικόκράτησης
81                                      String bookingID=passB.addPassenger(codeFlights);
82                                      //Βρίσκει ποιόςκόμβοςστηνλίσταείναιοσυγκεκριμένος
83                                      //επιβάτης
84                                      int index=passB.searchForCode(bookingID);
85                                      Passenger newPassenger=passengers.getNodeValue(index);
86                                      int listLength=newPassenger.getBookedFlights().getLength();
87                                      boolean available=true;
88                                      for(int i=0;i<listLength;i++){
89                                              //Ελεγχεί γιακάθεκωδικόπτήσηςανυπάρχουνδιαθέσιμες
90                                              //θέσεις. Ανέστωκαιμίαπτήσηδενέχειδιαθέσιμες
91                                              //θέσεις, τότεοεπιβάτηςμπαίνειστιςλίστεςαναμονής
92                                              //για κάθεμίαπτήση . Γιαλεπτομέρειεςδείτετηνπαρακάτω
93                                              //μέθοδο
94                                              available=flightB.checkAvailability(newPassenger
95                                                              .getBookedFlights().getNodeValue(i));
96                                              if(available==false)
97                                                      break;
98                                      }
99                                      for(int i=0;i<listLength;i++){
100                                             //Από τηλίσταμετουςκωδικούςπτήσεωντουεπιβάτη
    ,
101                                             //πέρνει έναένα τονκωδικό
102                                             String flightID=newPassenger.getBookedFlights()
103                                             .getNodeValue(i);
104                                             //Και κάνειμίακράτηση . Αντο available είναι true,
105                                             //τους βάζειστηλίσταεπιβίβασηςαλλιώςτουςβάζει
106                                             //στη λίστααναμονής
107                                             flightB.bookFlight(bookingID, flightID, available);
108                                     }
109                                     //Τυπώνεται αντίστοιχομήνυμακαιθέτειτηνκατάσταση
110                                     //κράτησης τουχρήστησε  true ή false.
111                                     if(available){
112                                             newPassenger.setStatus(true);
113                                             System.out.println("Your␣booking␣was␣successful");
114                                     }else{
115                                             newPassenger.setStatus(false);
116                                             System.out.println("There␣were␣no␣available␣seats." +
117                                                             "␣You've␣been␣" +
118                                             "placed␣to␣the␣waiting␣queue");
119                                     }
120                                     //Τυπώνει τομοναδικόαναγνωριστικόχρήστηγια
121                                     //πιθανή διαγραφήτηςκράτησης
122                                     System.out.println("Your␣booking␣code␣is:␣"+bookingID);
123                                     System.out.println("");
124                             }else{
125                                     //Αν οικωδικοίπτήσειςδενσυμβαδίζουντότετυπώνεται
126                                     //κατάλληλο μήνυμα
127                                     System.out.println("The␣flights'␣details␣were␣incorrect");
128                                     System.out.println("");
129                             }
130                             break;
131                     case 5:
132                             //Προβολή όλωντωνεπιβατών
133                             //Για λεπτομέρειεςδείτετηνπαρακάτωνμέθοδο
134                             passB.listPassengers();
135                             break;
136                     case 6:
137                             //Διαγραφή επιβάτη
138                             //Ο επιβάτηςσυμπληρώνειτομοναδικό
139                             //αναγνωριστικό του
140                             String bookingID=passB.askRemovePassenger();
141                             //Βρίσκει ποιοςκόμβοςαπότηλίσταμετουςεπιβάτες
142                             //είναι οσυγκεκριμένος .
143                             int index=passB.searchForCode(bookingID);
144                             Passenger delPassenger=passengers.getNodeValue(index);
```

```
145                             //Παίρνει τηλίσταμετουςκωδικούςπτήσεωνπουέχεικάνει
146                             //κράτηση οχρήστης .
147                             SimplyLinkedList<String> bookedFlights=delPassenger
148                             .getBookedFlights();
149                             int listLength=delPassenger.getBookedFlights().getLength();
150                             //Η κατάστασητηςκράτησηςτουχρήστη
151                             boolean status=delPassenger.getStatus();
152                                     if(status==false){
153                                             //Αν οχρήστηςήτανστιςουρέςαναμονήςτωνπτήσεων
154                                             //και όχιστιςλίστεςεπιβίβασης
155                                             for(int i=0;i<listLength;i++){
156                                                     //Για κάθεένακωδικόπτήσηςπουέχεικάνεικράτηση
157                                                     String flightCode=delPassenger.getBookedFlights()
158                                                     .getNodeValue(i);
159                                                     //Διαγράφει τονεπιβάτη . Γιαλεπτομέρειεςδείτετην
160                                                     //παρακάτω μέθοδο
161                                                     flightB.delPendingCode(bookingID, flightCode);
162                                             }
163                                             //Διαγράφει τονεπιβάτηαπότηλίσταμετουςεπιβάτες
164                                             passB.removePassenger(bookingID);
165                                     }else{
166                                             //Αν οχρήστηςήτανστιςλίστεςεπιβίβασης .
167                                             //Διαγράφουμε τονχρήστηαπότη boarding list κάθεπτήσης
168                                             //και πέρνουμετηνουράναναμονήςτηςπρώτηςπτήσηςσεσείρα
169 .
                                                //Λεπτομερώς στηνυλοποίησητηςπαρακάτωμεθόδου
170                                             SimplyLinkedList<String> queueCodes=flightB
171                                             .delBoardedCodePre(bookingID, bookedFlights);
172                                             //Μία λίσταπουθαέχειαντικείμενατύπου Passenger.
173                                             //Θα κρατάειτουςεπιβάτεςπουείναιστηνουράναμονής
174                                             //της παραπάνωπτήσης
175                                             SimplyLinkedList<Passenger> queuePassengers=new
176                                             SimplyLinkedList<Passenger>();
177                                             //Αντιστοιχίζει τουςκωδικούςεπιβατώναπότηλίστα
178 queueCodes
                                                //σε αντικείμενατύπου Passenger στηλίστα queuePassengers
179                                             for(int j=0;j<queueCodes.getLength();j++){
180                                                     index=passB.searchForCode(queueCodes.getNodeValue(j));
181                                                     queuePassengers.addTail(passengers.getNodeValue(index
182                                             }
183                                             if(queuePassengers!=null){
184                                                     //Αν υπάρχουνεπιβάτεςστηνουράναμονής
185                                                     //Ψάχνει γιαεπιβάτεςστηνουράναμονήςπουθα
186                                                     //πληρούσαν τακριτήριαγιαναμπουνστις
187 boarding
                                                        //lists τωνδικώντουςπτήσεων . Ανβρεθεί
188 ,
                                                        //επιστρέφεται οκωδικόςτουεπιβάτηκράτησης
189 _
                                                        String luckyBookingCode=flightB
190                                                     .delBoardedCodePost(queuePassengers);
191                                                     if(luckyBookingCode!=null){
192                                                             //Στον παραπάνωεπιβάτηορίζουμετο
193 status σε true
                                                             //Δηλαδή είναισεστιςλίστεςεπιβίβασηςκαιόχι
194                                                          //στις λίστεςαναμονής
195                                                          int indexb=passB.searchForCode(luckyBookingCo
196                                                          passengers.getNodeValue(indexb).setStatus(true
197                                                     }
198                                                     //Τέλος αφαιρούμετονεπιβάτηαπό
199                                                     //τη λίσταμετουςεπιβάτες
200                                                     passB.removePassenger(bookingID);
201                                             }
202                                     }
203                             break;
204                     case 0:
205                             //Έξοδος απότηνεφαρμογή
206                             //Το running είναι false οπότεσταματάει
```

```
207                                     //να εκτελείταιηεπανάληψη
208                                     running=false;
209                                     System.out.println("");
210                                     System.out.println("Arrivederci");
211                                     break;
212                             default:
213                                     //Σε περίπτωσηπουδοθείεπιλογήπουδεναντιστοιχεί
214                                     //σε κάποιοαπόταπαραπάνωτυπώνεικατάλληλομήνυμα
215                                     //λάθους
216                                     System.out.println("No available choice!");
217                                     break;
218                             }
219                     }
220             }
221     }
```

5.1.3 PassengerBusiness.java

```
1   package business;
2
3   import java.math.BigInteger;
4   import java.util.Scanner;
5
6   import entities.Passenger;
7   import structures.DoublyLinkedList;
8
9   /**
10   * Κλάσηγιατηδιαχείρισητωνεπιβατώνκαιότι
11   * έχεισχέσημεαυτούς .
12   */
13  public class PassengerBusiness {
14          private DoublyLinkedList<Passenger> passengers;
15          private String surname;
16          private String name;
17          private String idNumber;
18          private String nationality;
19          private String address;
20          private BigInteger phone;
21
22          /**
23           * Ορίζειτηλίσταμετουςεπιβάτεςγιατηνκλάσηαυτή .
24           *
25           * @param passengers Ηκλάσημετουςεπιβάτες .
26           */
27          public void setPassengersList(DoublyLinkedList<Passenger> passengers){
28                  this.passengers=passengers;
29          }
30          /**
31           * Τυπώνειόλουςτουςκαταχωρημένουςεπιβάτεςστηνεφαρμογή .
32           *
33           * @see Passenger#toString()
34           * @see DoublyLinkedList#toString()
35           */
36          public void listPassengers(){
37                  //Χρησιμοποιεί τιςμεθόδους toString γιανα
38                  //τυπώσει όλουςτουςεπιβάτες
39                  System.out.println(passengers);
40          }
41          /**
42           * Παίρνειόλαταναπαραίτητασταοιχείααπότον
43           * χρήστηγιατηδημιουργίαμιαςνέαςκράτησης .
44           *
45           * @return Τουςκωδικούςπτήσεωνπουέδωσεοχρήστης .
46           */
47          public String[] askAddPassenger(){
```

```
48                  Scanner in=new Scanner(System.in);
49                  System.out.println("Surname:");
50                  String surname=in.nextLine();
51                  this.surname=surname;
52                  System.out.println("Name:");
53                  String name=in.nextLine();
54                  this.name=name;
55                  System.out.println("ID Number:");
56                  String idNumber=in.nextLine();
57                  this.idNumber=idNumber;
58                  System.out.println("Nationality:");
59                  String nationality=in.nextLine();
60                  this.nationality=nationality;
61                  System.out.println("Address:");
62                  String address=in.nextLine();
63                  this.address=address;
64                  System.out.println("Flight code (comma separated):");
65                  String codeFlight=in.nextLine();
66                  System.out.println("Phone number:");
67                  BigInteger phone=in.nextBigInteger();
68                  this.phone=phone;
69                  //Μετατρέπει τουςκωδικούςπτήσεωνσεκεφαλαίαγια
70                  //διευκόλυνση τουχρήστη
71                  codeFlight=codeFlight.toUpperCase();
72                  //Χωρίζει τουςκωδικούςπτήσεωνμεδιαχωριστικότο       ','
73                  String[] codeFlights=codeFlight.split("[,]");
74
75                  return codeFlights;
76          }
77          /**
78           * Προσθέτειένανέοεπιβάτηκράτηση  - στηλίσταμε
79           * τουςεπιβάτες .
80           *
81           * @param codeFlights Οικωδικοίπτήσεωνπουέδωσεοεπιβάτης       .
82           * @return Τομοναδικόαναγνωριστικότηςκράτησης     .
83           * @see Utilities#Utilities()
84           * @see Utilities#MD5(String)
85           * @see Passenger#Passenger(String, String, String, String, String,
86           *  BigInteger, String)
87           * @see Passenger#setBookedFlights(String)
88           * @see DoublyLinkedList#addTail(Object)
89           */
90          public String addPassenger(String[] codeFlights){
91                  String uid;
92                  String stringUid="";
93                  Utilities util=new Utilities();
94                  for(int i=0;i<codeFlights.length;i++)
95                          //Για κάθεκωδικόπτήσης
96                          //Φτιάχνει ένα String μεόλουςτουςκωδικούςπτήσεων
97                          stringUid=stringUid.concat(codeFlights[i]);
98                  //Στο παραπάνω String προσθέτειτοναριθμό
99                  //ταυτότητας τουχρήστη
100                 stringUid=stringUid.concat(idNumber);
101                 //Χρησιμοποιώντας τησυνάρτησηκατακερματισμού
102                 //md5 στοπαραπάνω  String, παράγεταιτοαναγνωριστικό
103                 //της κράτησης
104                 uid=util.MD5(stringUid);
105                 //Από τοπαραπάνωαναγνωριστικότελικάχρησιμοποιούμε
106                 //μόνο ταπέντεπρώταψηφία
107                 uid=uid.substring(0, 5);
108                 //Δημιουργείται έναν έο  instance τηςκλάσης  Passenger γιατην
109                 //αποθήκευση τουνέοεπιβάτη
110                 Passenger newPassenger=new Passenger(surname,name,idNumber,nationality,
111                                 address,phone,uid);
112                 //Προσθέτει κάθεκωδικόπτήσηςπουέδωσεοχρήστηςστηλίστα
113                 //με τουςκωδικούςπτήσεωνπουέχεικάνειεκράτηση
114                 for(int i=0;i<codeFlights.length;i++){
115                         newPassenger.setBookedFlights(codeFlights[i]);
```

```
116                 }
117                 //Προσθέτει στηλίσταμετουςεπιβάτεςτονέοεπιβάτη
118                 passengers.addTail(newPassenger);
119 
120                 return uid;
121         }
122         /**
123          * Παίρνειαπότονχρήστητομοναδικόαναγνωριστικό
124          * κράτησηςπροςδιαγραφή .
125          *
126          * @return Τονκωδικόκράτησηςπουέδωσεοχρήστης .
127          */
128         public String askRemovePassenger(){
129                 System.out.println("Give your booking code:");
130                 Scanner in=new Scanner(System.in);
131                 String bookingCode=in.nextLine();
132 
133                 return bookingCode;
134         }
135 
136         /**
137          * Διαγράφειένανεπιβάτηαπότηλίσταμετουςεπιβάτες .
138          *
139          * @param bookingCode Τοαναγνωριστικόκράτησηςπροςδιαγραφή .
140          * @see PassengerBusiness#searchForCode(String)
141          * @see DoublyLinkedList#removeNode(int)
142          */
143         public void removePassenger(String bookingCode){
144                 //Ψάχνει στηλίσταμετουςεπιβάτες
145                 //τον επιβάτημετοσυγκεκριμένο
146                 //κωδικό κράτησης
147                 int index=searchForCode(bookingCode);
148                 passengers.removeNode(index);
149         }
150 
151         /**
152          * Ψάχνειστηλίσταμετουςεπιβάτεςγιατονεπιβάτημετο
153          * συγκεκριμένοκωδικόκράτησηςκαιεπιστρέφειτηθέσητου
154          * στηλίστα .
155          *
156          * @param bookingCode Οκωδικόςκράτησηςπουμαςενδιαφέρει .
157          * @return Τηθέσητουστηλίσταμετουςεπιβάτες . Ανδενβρεθεί
158          * επιστρέφει -1
159          * @see DoublyLinkedList#getLength()
160          * @see DoublyLinkedList#getNodeValue(int)
161          * @see Passenger#getUid()
162          */
163         public int searchForCode(String bookingCode){
164                 Passenger searchPassenger;
165                 int index;
166                 boolean found=false;
167                 int listLength=passengers.getLength();
168 
169                 for(index=1;index<=listLength;index++){
170                         //Για κάθεέναανεπιβάτη
171                         //Παίρνει τηντιμήτουκόμβου
172                         searchPassenger=passengers.getNodeValue(index);
173                         if(searchPassenger.getUid().equals(bookingCode)){
174                                 //Αν οκωδικόςκράτησηςείναιίδιοςμετον
175                                 //κωδικό κράτησηςπουμαςενδιαφέρει
176                                 found=true;
177                                 break;
178                         }
179                 }
180                 if(found==false)
181                         //Αν δενβρεθείεπιστρέφει -1
182                         index=-1;
183 
```

```
184                 return index;
185         }
186 }
```

5.1.4 Utilities.java

```
 1 package business;
 2
 3 import java.io.UnsupportedEncodingException;
 4 import java.security.MessageDigest;
 5 import java.security.NoSuchAlgorithmException;
 6
 7 /**
 8  * Κλάσημεβοηθητικέςμεθόδους    .
 9  */
10 public class Utilities {
11         /**
12          * Μετατρέπειμιασυμβολοσειράσεδεκαεξαδικήμορφή      .
13          *
14          * @param data Συμβολοσειρά.
15          * @return Δεκαεξαδικήαναπαράσταση .
16          */
17         private String convertToHex(byte[] data){
18                 StringBuffer buf=new StringBuffer();
19                 for(int i=0;i<data.length;i++){
20                         int halfByte=(data[i] >>> 4) & 0x0F;
21                         int twoHalfs=0;
22                         do{
23                                 if((0<=halfByte) && (halfByte<=9)){
24                                         buf.append((char) ('0'+halfByte));
25                                 }else{
26                                         buf.append((char) ('a'+(halfByte-10)));
27                                 }
28                                 halfByte=data[i] & 0x0F;
29                         }while(twoHalfs++<1);
30                 }
31
32                 return buf.toString();
33         }
34         /**
35          * Υλοποίησητηνσυνάρτησηςκατακερματισμού     md5.
36          *
37          * @param text Τοκείμενοαπότοοποίοθέλουμεναπάρουμε
38          * το md5sum.
39          * @return Το md5sum σεδεκαεξαδικήαναπαράσταση   .
40          */
41         public String MD5(String text){
42                 byte[] md5Hash=new byte[32];
43                 try{
44                         MessageDigest md=MessageDigest.getInstance("MD5");
45                         md.update(text.getBytes("UTF-8"),0,text.length());
46                         md5Hash=md.digest();
47                 }catch(NoSuchAlgorithmException e0){
48                         e0.printStackTrace();
49                 }catch(UnsupportedEncodingException e1){
50                         e1.printStackTrace();
51                 }
52
53                 return convertToHex(md5Hash);
54         }
55 }
```

5.2 entities

5.2.1 Flight.java

```
1  package entities;
2
3  import java.util.Date;
4  import structures.FifoQueue;
5  import structures.SimplyLinkedList;
6
7  /**
8   * Κλάσηπουκρατάειπληροφορίεςγιατιςδιαθέσιμεςπτήσεις        .
9   * Κάθε instance αυτήςτηςκλάσηςαποθηκεύεταισεμία
10  * διπλάσυνδεδεμένηλίστα   .
11  */
12 public class Flight {
13         private String flightCode;
14         private String startingPoint;
15         private String destination;
16         private Date departureTime;
17         private Date arrivalTime;
18         private double ticketPrice;
19         private String planeType;
20         private int totalSeats;
21         private int availableSeats;
22         //Μονά συνδεδεμένηλίσταμεόλουςτουςκωδικούς
23         //κράτησης τηςεκάστοτεπτήσης
24         private SimplyLinkedList<String> boardedPass;
25         //Ουρά αναμονήςμετουςκωδικούςκράτησης     , των
26         //επιβατών πουδενβρίκανελεύθερηθέσηστηνπτήση
27         //και προστέθηκανστηλίστααναμονής
28         private FifoQueue<String> waitingPass;
29
30         /**
31          * Constructor γιατηνδημιουργίαενόςνέου      instance μιαςπτήσης .
32          *
33          * @param flightCode Οκωδικόςπτήσης   .
34          * @param startingPoint Ηαφαιτηρία .
35          * @param destination Οπροορισμός .
36          * @param departureTime Ημερομηνίααναχώρησης .
37          * @param arrivalTime Ημερομηνίαάφιξης .
38          * @param ticketPrice Τιμήεισιτηρίου .
39          * @param planeType Τύποςαεροπλάνου .
40          * @param totalSeats Συνολικέςθέσειςαεροπλάνου   .
41          * @param availableSeats Διαθέσιμεςθέσειςαεροπλάνου   .
42          */
43         public Flight(String flightCode, String startingPoint, String destination,
44                         Date departureTime, Date arrivalTime, double ticketPrice,
45                         String planeType, int totalSeats, int availableSeats){
46                 this.flightCode=flightCode;
47                 this.startingPoint=startingPoint;
48                 this.destination=destination;
49                 this.departureTime=departureTime;
50                 this.arrivalTime=arrivalTime;
51                 this.ticketPrice=ticketPrice;
52                 this.planeType=planeType;
53                 this.totalSeats=totalSeats;
54                 this.availableSeats=availableSeats;
55                 //Δημιουργία τηςλίσταςμετουςεπιβάτες
56                 boardedPass=new SimplyLinkedList<String>();
57                 //Δημιουργία τηςουράςαναμονήςμετουςεπιβάτεςπουαναμένουν
58                 waitingPass=new FifoQueue<String>();
59         }
60
61         /**
62          * Παίρνουμετονκωδικόπτήσης    .
63          *
```

```
64          * @return Τονκωδικόπτήσης   .
65          */
66         public String getFlightCode(){
67                 return flightCode;
68         }
69         /**
70          * Παίρνουμετηναφετηρίατηςπτήσης     .
71          *
72          * @return Τηναφετηρίατηςπτήσης    .
73          */
74         public String getStartingPoint(){
75                 return startingPoint;
76         }
77         /**
78          * Παίρνουμετονπροορισμότηςπτήσης     .
79          *
80          * @return Τονπροορισμότηςπτήσης    .
81          */
82         public String getDestination(){
83                 return destination;
84         }
85         /**
86          * Παίρνουμετηνημερομηνίααναχώρησηςτηςπτήσης      .
87          *
88          * @return Τηνημερομηνίααναχώρησηςτηςπτήσης     .
89          */
90         public Date getDepartureTime(){
91                 return departureTime;
92         }
93         /**
94          * Παίρνουμετηνημερομηνίαάφιξηςτηςπτήσης      .
95          *
96          * @return Τηνημερομηνίαάφιξηςτηςπτήσης     .
97          */
98         public Date getArrivalTime(){
99                 return arrivalTime;
100        }
101        /**
102         * Παίρνουμετηντιμήτουεισιτηρίουτηςπτήσης       .
103         *
104         * @return Τηντιμήτουεισιτηρίουτηςπτήσης      .
105         */
106        public double getTicketprice(){
107                return ticketPrice;
108        }
109        /**
110         * Παίρνουμετοντύποτουαεροπλάνου     .
111         *
112         * @return Τοντύποτουαεροπλάνου    .
113         */
114        public String getPlaneType(){
115                return planeType;
116        }
117        /**
118         * Παίρνουμετοναριθμότωνσυνολικώνθέσεωνστοαεροπλάνο       .
119         *
120         * @return Τοναριθμότωνσυνολικώνθέσεωνστοαεροπλάνο       .
121         */
122        public int getTotalSeats(){
123                return totalSeats;
124        }
125        /**
126         * Παίρνουμετιςδιαθέσιμεςθέσειςστοαεροπλάνο      .
127         *
128         * @return Τιςδιαθέσιμεςθέσειςστοαεροπλάνο     .
129         */
130        public int getAvailableSeats(){
131                return availableSeats;
```

```
132         }
133         /**
134          * Πέρνουμετηλίσταμετουςεπιβάτεςπροςεπιβίβαση .
135          *
136          * @return Τηλίσταμετουςεπιβάτεςπροςεπιβίβαση .
137          */
138         public SimplyLinkedList<String> getBoardedPass(){
139                 return boardedPass;
140         }
141         /**
142          * Παίρνουμετηνουράαναμονήςμετουςεπιβάτεςσεαναμονή .
143          *
144          * @return Τηνουράαναμονήςμετουςεπιβάτεςσεαναμονή .
145          */
146         public FifoQueue<String> getWaitingPass(){
147                 return waitingPass;
148         }
149         /**
150          * Προσθέτουμεστηλίσταεπιβίβασηςένανέοεπιβάτη .
151          *
152          * @param bookingCode Ομοναδικόςκωδικόςεπιβάτηκράτησης −.
153          */
154         public void setBoardedPass(String bookingCode){
155                 boardedPass.addTail(bookingCode);
156         }
157         /**
158          * Προσθέτουμεστηνουράαναμονήςένανέοεπιβάτη .
159          *
160          * @param bookingCode Ομοναδικόςκωδικόςεπιβάτηκράτησης −.
161          */
162         public void setWaitingPass(String bookingCode){
163                 waitingPass.enqueue(bookingCode);
164         }
165         /**
166          * Ορίζουμετηνουράαναμονής .
167          *
168          * @param waitingPass Ηουράαναμονής .
169          */
170         public void setWaitingPass(FifoQueue<String> waitingPass){
171                 this.waitingPass=waitingPass;
172         }
173         /**
174          * Ορίζουμετιςδιαθέσιμεςθέσειςτουαεροπλάνου .
175          *
176          * @param availableSeats Οιδιαθέσιμεςθέσειςτουαεροπλάνου .
177          */
178         public void setAvailableSeats(int availableSeats){
179                 this.availableSeats=availableSeats;
180         }
181         /**
182          * Επιστρέφειόλεςτιςλεπτομέρειεςμιαςπτήσης .
183          *
184          * @return Τιςλεπτομέρειεςμιαςπτήσης .
185          */
186         @Override
187         public String toString(){
188                 StringBuilder sb=new StringBuilder();
189                 sb.append("\n");
190                 sb.append("————————————————————").append("\n");
191                 sb.append("Flight␣Code:␣").append(flightCode).append("\n");
192                 sb.append("Starting␣Point:␣").append(startingPoint).append("\n");
193                 sb.append("Destination:␣").append(destination).append("\n");
194                 sb.append("Departure␣Time:␣").append(departureTime).append("\n");
195                 sb.append("Arrival␣Time:␣").append(arrivalTime).append("\n");
196                 sb.append("Ticket␣Price:␣").append(ticketPrice).append("\n");
197                 sb.append("Plane␣Type:␣").append(planeType).append("\n");
198                 sb.append("Total␣Seats:␣").append(totalSeats).append("\n");
199                 sb.append("Available␣Seats:␣").append(availableSeats);
```

```
200                     //Αν ηουράαναμονήςέχεικόμβους     ,
201                     //δηλαδή ανυπάρχουνεπιβάτεςσεαναμονή     .
202                     if(waitingPass.getLength()>0){
203                             sb.append("\n");
204                             sb.append("Pending␣Seats:␣").append(waitingPass.getLength())
205                             .append("\n");
206                     }
207
208                     return sb.toString();
209             }
210     }
```

5.2.2 Passenger.java

```
1   package entities;
2
3   import java.math.BigInteger;
4   import structures.SimplyLinkedList;
5
6   /**
7    * Κλάσηπουκρατάειπληροφορίεςγιατουεπιβάτεςκρατήσεις       −.
8    * Κάθε instance αυτήςτηςκλάσηςαποθηκεύεταισεμίαδιπλά
9    * συνδεδεμένηλίστα .
10   */
11  public class Passenger {
12          private String surname;
13          private String name;
14          private String idNumber;
15          private String nationality;
16          private String address;
17          private BigInteger phone;
18          private String uid;
19          //true ανοεπιβάτηςείναισελίσταεπιβίβασης       ,
20          //false ανοεπιβάτηςείναισεουράαναμονής
21          private boolean status;
22          //Λίστα μετουςκωδικούςπτήσεωνπουέχεικάνεικράτησηοεπιβάτης
23          private SimplyLinkedList<String> bookedFlightsList;
24
25          /**
26           * Constructor γιατηδημιουργίαενόςνέου     instance επιβάτηκράτησης−.
27           *
28           * @param surname Τοεπίθετοτουεπιβάτη    .
29           * @param name Τοόνοματουεπιβάτη    .
30           * @param idNumber Οαριθμόςταυτότητηταςτουεπιβάτη     .
31           * @param nationality Ηεθνικότητατουεπιβάτη    .
32           * @param address Ηδιεύθυνσητουεπιβάτη    .
33           * @param phone Τοτηλέφωνοτουεπιβάτη    .
34           * @param uid Τομοναδικόαναγνωριστικότηςκράτησης     .
35           */
36          public Passenger(String surname, String name, String idNumber,
37                          String nationality, String address, BigInteger phone, String uid){
38                  this.surname=surname;
39                  this.name=name;
40                  this.idNumber=idNumber;
41                  this.nationality=nationality;
42                  this.address=address;
43                  this.phone=phone;
44                  this.uid=uid;
45                  //Δημιουργία τηςλίσταςμετουςκωδικούςπτήσηςπουέχεικάνει
46                  //κράτηση οεπιβάτης
47                  bookedFlightsList=new SimplyLinkedList<String>();
48          }
49          /**
50           * Παίρνουμετοεπίθετοτουεπιβάτη     .
51           *
```

```
52              * @return Τοεπίθετοτουεπιβάτη    .
53              */
54             public String getSurname(){
55                     return surname;
56             }
57             /**
58              * Παίρνουμετοόνοματουεπιβάτη     .
59              *
60              * @return Τοόνοματουεπιβάτη    .
61              */
62             public String getName(){
63                     return name;
64             }
65             /**
66              * Παίρνουμετοναριθμόταυτότηταςτουεπιβάτη      .
67              *
68              * @return Τοναριθμόταυτότηταςτουεπιβάτη     .
69              */
70             public String getIdNumber(){
71                     return idNumber;
72             }
73             /**
74              * Παίρνουμετηνεθνικότητατουχρήστη     .
75              *
76              * @return Τηνεθνικότητατουχρήστης    .
77              */
78             public String getNationality(){
79                     return nationality;
80             }
81             /**
82              * Παίρνουμετηδιεύθυνσητουχρήστη     .
83              *
84              * @return Τηδιεύθυνσητουχρήστη    .
85              */
86             public String getAddress(){
87                     return address;
88             }
89             /**
90              * Παίρνουμετοτηλέφωνοτουχρήστη     .
91              *
92              * @return Τοτηλέφωνοτουχρήστη    .
93              */
94             public BigInteger getPhone(){
95                     return phone;
96             }
97             /**
98              * Παίρνουμετομοναδικόαναγνωριστικότηςκράτησης      .
99              *
100             * @return Τομοναδικόαναγνωριστικότηςκράτησης     .
101             */
102            public String getUid(){
103                    return uid;
104            }
105            /**
106             * Παίρνουμετηνκατάστασηκράτησηςτουεπιβάτη      .
107             *
108             * @return Τηνκατάστασηκράτησηςτουεπιβάτη     .
109             */
110            public boolean getStatus(){
111                    return status;
112            }
113            /**
114             * Παίρνουμετηλίσταμετουςκωδικούςπτήσεωνπουέχεικάνει
115             * κράτησηοεπιβάτης   .
116             *
117             * @return Τηλίσταμετουςκωδικούςπτήσεωνπουέχεικάνει
118             * κράτησηοεπιβάτης   .
119             */
```

```
120         public SimplyLinkedList<String> getBookedFlights(){
121                 return bookedFlightsList;
122         }
123         /**
124          * Προσθέτουμεένανέοκωδικόπτήσηςστηνυπάρχουσαλίστακράτησης .
125          *
126          * @param flightCode Οκωδικόςπτήσης .
127          */
128         public void setBookedFlights(String flightCode){
129                 bookedFlightsList.addTail(flightCode);
130         }
131         /**
132          * Ορίζουμετηνκατάστασηκράτησηςτουεπιβάτη .
133          *
134          * @param status Ηκατάστασηκράτησηςτουεπιβάτη .
135          */
136         public void setStatus(boolean status){
137                 this.status=status;
138         }
139         /**
140          * Επιστρέφειόλεςτιςλεπτομέρειεςενόςεπιβάτηκράτησης -.
141          *
142          * @return Τιςλεπτομέρειεςενόςεπιβάτηκράτησης -.
143          */
144         @Override
145         public String toString(){
146                 StringBuilder sb=new StringBuilder();
147                 sb.append("\n");
148                 sb.append("Surname: ").append(surname).append("\n");
149                 sb.append("Name: ").append(name).append("\n");
150                 sb.append("ID Number: ").append(idNumber).append("\n");
151                 sb.append("Nationality: ").append(nationality).append("\n");
152                 sb.append("Address: ").append(address).append("\n");
153                 sb.append("Phone: ").append(phone).append("\n");
154                 sb.append("uid: ").append(uid).append("\n");
155                 sb.append("Booked Flights: ").append(bookedFlightsList).append("\n");
156                 sb.append("Status: ");
157                 if(status){
158                         sb.append("Boarded");
159                 }else{
160                         sb.append("Pending");
161                 }
162                 sb.append("\n");
163
164                 return sb.toString();
165         }
166 }
```

5.3 structures

5.3.1 DoublyLinkedList.java

```
1  package structures;
2
3  /**
4   * Κλάσηπουαντιπροσωπεύειένακόμβοστηδιπλάσυνδεδεμένηλίστα .
5   * Κρατάειτηντιμήτουκόμβου , τονεπόμενοκαιτονπροηγούμενο .
6   *
7   * @param <E> Οτύποςτωνδεδομένωνπουθααποθηκεύονταιστοκόμβο .
8   */
9  class DNode<E>{
10         private E value;
11         private DNode<E> previousNode;
12         private DNode<E> nextNode;
```

```
13
14          /**
15           * Ο constructor μετονοποίοορίζουμετην
16           * τιμήτουκόμβου .
17           *
18           * @param value Ητιμήτουκόμβου .
19           */
20          DNode(E value){
21                  this.value=value;
22          }
23
24          /**
25           * Ορίζουμετονπροηγούμενοκόμβοστηλίστα .
26           *
27           * @param previousNode Οπροηγούμενοςκόμβοςστηλίστα .
28           */
29          void setPreviousNode(DNode<E> previousNode){
30                  this.previousNode=previousNode;
31          }
32
33          /**
34           * Ορίζουμετονεπόμενοκόμβοστηλίστα .
35           *
36           * @param nextNode Οεπόμενοςκόμβοςστηλίστα .
37           */
38          void setNextNode(DNode<E> nextNode){
39                  this.nextNode=nextNode;
40          }
41
42          /**
43           * Παίρνουμετηντιμήτουκόμβου .
44           *
45           * @return Τηντιμήτουκόμβου .
46           */
47          E getValue(){
48                  return value;
49          }
50
51          /**
52           * Παίρνουμετονπροηγούμενοκόμβοαπότηλίστα .
53           *
54           * @return Τονπροηγούμενοκόμβοαπότηλίστα .
55           */
56          DNode<E> getPreviousNode(){
57                  return previousNode;
58          }
59
60          /**
61           * Παίρνουμετονεπόμενοκόμβοαπότηλίστα .
62           *
63           * @return Τονεπόμενοκόμβοαπότηλίστα .
64           */
65          DNode<E> getNextNode(){
66                  return nextNode;
67          }
68  }
69
70  /**
71   * Κλάσηπουυλοποιείτηνδιπλάσυνδεδεμένηλίστα .
72   *
73   * @param <E> Τοντύποδεδομένωνπουθαχειρίζεταιηλίστα .
74   */
75  public class DoublyLinkedList<E> {
76          //Δύο βοηθητικοίκόμβοιπουαντιπροσωπεύουν
77          //την αρχήκαιτοτέλοςτηςλίστας
78          private DNode<E> head=new DNode<E>(null);
79          private DNode<E> tail=new DNode<E>(null);
80          //Το μήκοςτηςλίστας
```

```
81          private int length=0;
82
83          /**
84           * Constructor μετονοποίοορίζουμεότιοπροηγούμενος
85           * κόμβοςτουπρώτουείναιτο      null, οεπόμενόςτουείναι
86           * οτελευταίος , οπροηγούμενοςτουτελευταίουείναιο
87           * πρώτοςκαιοεπόμενοςτουτελευταίουείναιτο          null.
88           */
89          public DoublyLinkedList(){
90                  head.setPreviousNode(null);
91                  head.setNextNode(tail);
92                  tail.setPreviousNode(head);
93                  tail.setNextNode(null);
94          }
95
96          /**
97           * Τουδίνουμεέναδείκτηκαιπαίρνουμετονσυγκεκριμένοκόμβο          .
98           *
99           * @param Οδείκτηςτουκόμβουπουμαςενδιαφέρει        .
100          * @return Τονκόμβοσύμφωναμετονδείκτη       .
101          * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
102          * μικρότεροςτουμηδένήμεγαλύτεροςτουμήκουςτηςλίστας          .
103          */
104         public DNode<E> getNode(int index) throws IndexOutOfBoundsException{
105                 if(index<0 || index>length){
106                         System.err.println("Index␣out␣of␣bounds");
107                         throw new IndexOutOfBoundsException();
108                 }else{
109                         //Παίρνουμε τονεπόμενοκόμβομέχριτονδείκτηπουορίσαμε
110                         DNode<E> cursor=head;
111                         for(int i=0;i<index;i++)
112                                 cursor=cursor.getNextNode();
113
114                         return cursor;
115                 }
116         }
117
118         /**
119          * Παίρνουμετηντιμήενόςκόμβουχωρίςόμωςνατοναφαιρέσουμεαπότηλίστα          .
120          *
121          * @param index Οδείκτηςγιατονκόμβοπουθέλουμε       .
122          * @return Τηντιμήτουκόμβουπουθέλουμε       .
123          * @see DoublyLinkedList#getNode(int)
124          */
125         public E getNodeValue(int index){
126                 DNode<E> cursor=getNode(index);
127
128                 return cursor.getValue();
129         }
130
131         /**
132          * Αφαιρείένασυγκεκριμένοκόμβοαπότηλίστακαιπαίρνουμετηντιμήτου
133          * σύμφωναμετονδείκτηπουδώσαμε       .
134          *
135          * @param index Οδείκτηςγιατονκόμβοπουθέλουμε       .
136          * @return Τηντίμητουκόμβουπουθέλουμε       .
137          * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
138          * μικρότεροςτουμηδένήμεγαλύτεροςτουμήκουςτηςλίστας          .
139          * @see DoublyLinkedList#getNode(int)
140          * @see DNode#getNextNode()
141          * @see DNode#setPreviousNode(DNode)
142          * @see DNode#getPreviousNode()
143          * @see DNode#setNextNode(DNode)
144          */
145         public E removeNode(int index) throws IndexOutOfBoundsException{
146                 if(index<0 || index>length){
147                         System.err.println("Index␣out␣of␣bounds");
148                         throw new IndexOutOfBoundsException();
```

```
149                 }else{
150                         DNode<E> resultNode=getNode(index);
151                         resultNode.getNextNode().setPreviousNode(resultNode.getPreviousNode());
152                         resultNode.getPreviousNode().setNextNode(resultNode.getNextNode());
153                         length--;
154 
155                         return resultNode.getValue();
156                 }
157         }
158 
159         /**
160          * Προσθέτειένακόμβοστηλίστασεσυγκεκριμένηθέση        .
161          *
162          * @param index Ηθέσηπουθέλουμεναβάλουμετονκόμβο        .
163          * @param value Ητιμήπουθέλουμεναέχειοκόμβος        .
164          * @see DoublyLinkedList#getNode(int)
165          * @see DNode#DNode(Object)
166          * @see DNode#setPreviousNode(DNode)
167          * @see DNode#setNextNode(DNode)
168          * @see DNode#getNextNode()
169          */
170         public void add(int index, E value){
171                 DNode<E> cursor=getNode(index);
172                 DNode<E> newNode=new DNode<E>(value);
173                 newNode.setPreviousNode(cursor);
174                 newNode.setNextNode(cursor.getNextNode());
175                 cursor.getNextNode().setPreviousNode(newNode);
176                 cursor.setNextNode(newNode);
177                 length++;
178         }
179 
180         /**
181          * Προσθέτειένανέοκόμβοστηναρχήτηςλίστας        .
182          *
183          * @param value Ητιμήπουθέλουμεναέχειονέοςκόμβος        .
184          * @see DNode#DNode(Object)
185          * @see DNode#setPreviousNode(DNode)
186          * @see DNode#setNextNode(DNode)
187          * @see DNode#getNextNode()
188          */
189         public void addHead(E value){
190                 DNode<E> cursor=head;
191                 DNode<E> newNode=new DNode<E>(value);
192                 newNode.setPreviousNode(cursor);
193                 newNode.setNextNode(cursor.getNextNode());
194                 cursor.getNextNode().setPreviousNode(newNode);
195                 cursor.setNextNode(newNode);
196                 length++;
197         }
198 
199         /**
200          * Προσθέτειένανέοκόμβοστοτέλοςτηςλίστας        .
201          *
202          * @param value Ητιμήπουθέλουμεναέχειονέοςκόμβος        .
203          * @see DNode#DNode(Object)
204          * @see DNode#getPreviousNode()
205          * @see DNode#setPreviousNode(DNode)
206          * @see DNode#getNextNode()
207          * @see DNode#setNextNode(DNode)
208          */
209         public void addTail(E value){
210                 DNode<E> cursor=tail.getPreviousNode();
211                 DNode<E> newNode=new DNode<E>(value);
212                 newNode.setPreviousNode(cursor);
213                 newNode.setNextNode(cursor.getNextNode());
214                 cursor.getNextNode().setPreviousNode(newNode);
215                 cursor.setNextNode(newNode);
216                 length++;
```

```
217         }
218
219         /**
220          * Μαςεπιστρέφειτομήκοςτηςλίστας      .
221          *
222          * @return οT μήκοςτηςλίστας   .
223          */
224         public int getLength(){
225                 return length;
226         }
227
228         /**
229          * Ελέγχειανηλίσταείναιάδεια      .
230          *
231          * @return true ανηλίσταείναιάδεια     , false αν
232          * ηλίσταέχεικόμβους    .
233          */
234         public boolean isEmpty(){
235                 return length==0?true:false;
236         }
237
238         /**
239          * Επιστρέφειτιςτιμέςκάθεκόμβουτηςλίστας       .
240          *
241          * @return ιςT τιμέςκάθεκόμβουτηςλίστας     .
242          */
243         @Override
244         public String toString(){
245                 StringBuffer sb=new StringBuffer();
246                 sb.append("\n");
247                 DNode<E> tmpNode=head;
248                 while(tmpNode.getNextNode()!=tail){
249                         tmpNode=tmpNode.getNextNode();
250                         sb.append(tmpNode.getValue());
251                 }
252                 sb.append("\n");
253
254                 return sb.toString();
255         }
256 }
```

5.3.2 FifoQueue.java

```
1  package structures;
2
3  /**
4   * Κλάσηπουαντιπροσωπεύειένακόμβοστηνουράαναμονής        .
5   * Κρατάειτηντιμήτουκόμβουκαιτονεπόμενοκόμβο          .
6   *
7   * @param <E> Οτύποςτωνδεδομένωνπουθααποθηκεύονταιστοκόμβο          .
8   */
9  class QNode<E>{
10         private E value;
11         private QNode<E> nextNode;
12
13         /**
14          * Constructor μετονοποίοορίζουμετηντιμήτουκόμβου        .
15          *
16          * @param value Ητίμητουκόμβου    .
17          */
18         QNode(E value){
19                 this.value=value;
20         }
21
22         /**
```

```
23          * Ορίζουμετονεπόμενοκόμβοστηνουρά      .
24          *
25          * @param nextNode Οεπόμενοςκόμβος   .
26          */
27         void setNextNode(QNode<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          * Παίρνουμετονεπόμενοκόμβοαπότηνουρά       .
42          *
43          * @return Τονεπόμενοκόμβοαπότηνουρά      .
44          */
45         QNode<E> getNextNode(){
46                 return nextNode;
47         }
48 }
49
50 /**
51  * Κλάσηπουυλοποιείτηνουράαναμονής      .
52  *
53  * @param <E> Οτύποςδεδομένωνπουθαχειρίζεταιηουρά        .
54  */
55 public class FifoQueue<E> {
56         //Ο πρώτοςκαιτελευταίοςκόμβοςστηνουρά
57         private QNode<E> head;
58         private QNode<E> tail;
59         //Το μήκοςτηςουράς
60         private int length;
61
62         /**
63          * Constructor πουαρχικοποιείτονπρώτοκαιτοντελευταίο
64          * κόμβοσε  null καιτομήκοςσεμηδέν     .
65          */
66         public FifoQueue(){
67                 head=null;
68                 tail=null;
69                 length=0;
70         }
71
72         /**
73          * Βάζειμίατιμήστηνουράαναμονής      .
74          *
75          * @param value Ητιμήτουνέουκόμβουστηνουρά       .
76          * @see QNode#QNode(Object)
77          * @see QNode#setNextNode(QNode)
78          */
79         public void enqueue(E value){
80                 if(head==null){
81                         //Ο νέοςκόμβοςείναιοπρώτοςκόμβος
82                         //που μπαίνειστηνουρά
83                         head=new QNode<E>(value);
84                         head.setNextNode(null);
85                         tail=head;
86                 }else{
87                         //Ο νέοςκόμβοςμπαίνειμετάτοντελευταίοκόμβο
88                         QNode<E> newNode=new QNode<E>(value);
89                         newNode.setNextNode(null);
90                         tail.setNextNode(newNode);
```

```
91                      tail=newNode;
92              }
93              length++;
94      }
95
96      /**
97       * Παίρνουμετονπρώτοκόμβοαπότη       FIFO ουράκαιτονδιαγράφουμε    .
98       *
99       * @return Τηντιμήτουπρώτουκόμβου    .
100      * @throws IndexOutOfBoundsException Σεπερίπτωσηπουηλίσταείναιάδεια       .
101      * @see QNode#getValue()
102      * @see QNode#getNextNode()
103      */
104     public E dequeue() throws IndexOutOfBoundsException{
105             if(length==0){
106                     System.err.println("Queue is empty!");
107                     throw new IndexOutOfBoundsException();
108             }else{
109                     E value=head.getValue();
110                     //Ορίζουμε ωςπρώτοκόμβο  , τονεπόμενο
111                     head=head.getNextNode();
112                     length--;
113
114                     return value;
115             }
116     }
117
118     /**
119      * Παίρνουμεένασυγκεκριμένοκόμβοαπότηνουράσύμφωνα
120      * μετονδείκτηχωρίςνατονδιαγράψουμε       .
121      *
122      * @param index Οδείκτηςτουκόμβουπουμαςενδιαφέρει       .
123      * @return Τονκόμβοσύμφωναμετοδείκτη     .
124      * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
125      * μικρότεροςτουμηδένήμεγαλύτεροςτουμήκουςτηςουράς         .
126      * @see QNode#getNextNode()
127      */
128     public QNode<E> getNode(int index) throws IndexOutOfBoundsException{
129             if(index<0 || index>length){
130                     System.err.println("Index out of bounds");
131                     throw new IndexOutOfBoundsException();
132             }else{
133                     QNode<E> cursor=head;
134                     for(int i=0;i<index;i++)
135                             cursor=cursor.getNextNode();
136
137                     return cursor;
138             }
139     }
140
141     /**
142      * Παίρνειτηντιμήαπόένανενδιάμεσοκόμβοαπότηνουρά
143      * καιτοναφαιρεί  .
144      *
145      * @param index Οδείκτηςτουκόμβουπουμαςενδιαφέρει       .
146      * @return Τηντιμήτουκόμβουπουμαςενδιαφέρει       .
147      * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
148      * μικρότεροςτουμηδένήμεγαλύτεροςτουμήκουςτηςουράς         .
149      * @see FifoQueue#dequeue()
150      * @see FifoQueue#getNode(int)
151      * @see QNode#getNextNode()
152      * @see QNode#getValue()
153      * @see QNode#setNextNode(QNode)
154      */
155     public E removeNode(int index) throws IndexOutOfBoundsException{
156             if(index<0 || index>length){
157                     System.err.println("Index out of bounds");
158                     throw new IndexOutOfBoundsException();
```

```
159                 }else if(index==0){
160                         return dequeue();
161                 }else{
162                         QNode<E> tmpNode=getNode(index-1);
163                         E value=tmpNode.getNextNode().getValue();
164                         tmpNode.setNextNode(tmpNode.getNextNode().getNextNode());
165                         length--;
166 
167                         return value;
168                 }
169         }
170 
171         /**
172          * Παίρνουμετηντιμήενόςσυγκεκριμένουκόμβουσύμφωναμετονδείκτη
173          * χωρίςόμωςνατοναφαιρέσουμεαπότηνουρά .
174          *
175          * @param index Ηθέσητουκόμβουστηνουρά .
176          * @return Τηντιμήτουκόμβουπουμαςενδιαφέρει .
177          * @throws IndexOutOfBoundsException Σεπερίπτωσηπουοδείκτηςείναι
178          * μικρότεροςτουμηδένήμεγαλύτεροςτουμεγέθουςτηςουράς .
179          * @see FifoQueue#getNode(int)
180          * @see QNode#getValue()
181          */
182         public E getNodeValue(int index) throws IndexOutOfBoundsException{
183                 if(index<0 || index>length){
184                         System.err.println("Index out of bounds");
185                         throw new IndexOutOfBoundsException();
186                 }else if(index==0){
187                         return head.getValue();
188                 }else{
189                         QNode<E> tmpNode=getNode(index-1);
190                         E value=tmpNode.getValue();
191 
192                         return value;
193                 }
194         }
195         /**
196          * Παίρνουμετομήκοςτηςουράς .
197          *
198          * @return Τομήκοςτηςουράς .
199          */
200         public int getLength(){
201                 return length;
202         }
203 
204         /**
205          * Ελέγχουμεανηουράείναιάδεια .
206          *
207          * @return true ανηουράείναιάδεια , false ανηουράέχεικόμβους .
208          */
209         public boolean isEmpty(){
210                 return length==0?true:false;
211         }
212 
213         /**
214          * Επιστρέφειτιςτιμέςκάθεκόμβουστηνουράαναμονής .
215          *
216          * @return Τιςτιμέςκάθεκόμβουστηνουράαναμονής .
217          */
218         public String toString(){
219                 if(length==0){
220                         return "Queue is empty!";
221                 }else{
222                         StringBuilder sb=new StringBuilder();
223                         sb.append("\n");
224                         QNode<E> tmpNode=head;
225                         while(tmpNode.getNextNode()!=null){
226                                 sb.append(tmpNode.getValue()).append(" - ");
```

```
227                                     tmpNode=tmpNode.getNextNode();
228                             }
229                             sb.append(tmpNode.getValue());
230                             sb.append("\n");
231
232                             return sb.toString();
233                     }
234             }
235     }
```

5.3.3 SimplyLinkedList.java

```
 1  package structures;
 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         /**
297          * Επεστρέφειτιςτιμέςκάθεκόμβουστηλίστα .
298          *
299          * @return Τιςτιμέςκάθεκόμβουστηλίστα .
300          * @see SNode#getValue()
301          * @see SNode#getNextNode()
302          */
303         @Override
304         public String toString(){
305                 if(length==0){
306                         return "List is empty";
307                 }else{
308                         StringBuilder sb=new StringBuilder();
309                         sb.append("\n");
310                         SNode<E> tmpNode=head;
311                         while(tmpNode.getNextNode()!=null){
312                                 sb.append(tmpNode.getValue()).append(" - ");
313                                 tmpNode=tmpNode.getNextNode();
314                         }
315                         sb.append(tmpNode.getValue());
316                         sb.append("\n");
317
318                         return sb.toString();
319                 }
320         }
321 }
```