Προηγμένα Θέματα Προγραμματισμού

Τελική Εργασία

Ξυράφης Γιώργος
Α.Μ.:2022201800138
Θεοχάρη Ηρώ
Α.Μ.: 2022201800058

19/07/21

Client.java

Το αρχείο Client.java είναι το αρχικό αρχείο που μας δόθηκε από την μεριά του Client με το project. Αρχικά εκεί βρισκόταν το μεγαλύτερο μέρος του κώδικα, όμως μετά από refactoring το μόνο που μένει είναι η μέθοδος start() της javaFX στην οποία εκτελείται η μέθοδος create της κλάσης ClientGUI, η οποία, όπως δηλώνει και το όνομα, δημιουργεί το GUI. Πέρα από αυτά, φυσικά υπάρχει και η main.

ClientGUI.java

Η κλάση ClientGUI.java περιέχει μόνο την στατική μέθοδο create που δέχεται ως ορίσματα το stage, το οποίο είναι απαραίτητο προκειμένου να αλλάζουμε σκηνές κλπ. Στην μέθοδο αυτή πραγματοποιείται η υλοποίηση του GUI της εφαρμογής Client. Αρχικά δηλώνονται οι απαραίτητες μεταβλητές για δύο σκηνές. Η πρώτη σκηνή είναι η αρχική σελίδα της εφαρμογής, όπου ο χρήστης με χρήση εικονικού πληκτρολογίου εισάγει τον αριθμό κυκλοφορίας του οχήματος του. Η δεύτερη είναι η σκηνή με τις παρεχόμενες υπηρεσίες, η οποία ναι μεν έχει ορισμένα σταθερά στοιχεία, αλλά όπως θα δούμε παρακάτω δημιουργείται δυναμικά ως έναν βαθμό ανάλογα με την επιλογή οχήματος του χρήστη. Επίσης ορίζεται ένας eventhandler ο οποίος είναι υπεύθυνος για όλα τα κουμπιά του πληκτρολογίου. Μετά τον ορισμό των μεταβλητών γίνονται οι απαραίτητες ενέργειες προκειμένου τα κουμπιά να πάρουν το κατάλληλο μέγεθος και να μπουν στις σωστές θέσεις με βάση τις οδηγίες της εκφώνησης. Επίσης ανατίθεται σε όλα τα πλήκτρα ο eventhandler που ορίσαμε προηγουμένως. Τέλος το stage δείχνει την πρώτη σκηνή που έχουμε φτιάξει και ορίζεται τίτλος και μέγιστο και ελάχιστο μέγεθος για το παράθυρο της εφαρμογής.

Keyboardhandler.java

Η κλάση Keyboardhandler.java είναι μια κλάση που συνδυάζει λειτουργικότητα και γραφικά. Αυτό, ενώ ίσως να μην είναι η καλύτερη επιλογή, είναι κάτι που έγινε συνειδητά, μια αναγκαιότητα για την συγκεκριμένη υλοποίηση του πρότζεκτ, αλλά αυτό θα το αναλύσουμε παρακάτω. Αρχικά αυτή η κλάση παίρνει ως ορίσματα το label που προβάλει την πινακίδα, το label που προβάλει την τιμή στην δεύτερη σκηνή, το stage, ένα gridpane που περιέχει τις διάφορες υπηρεσίες που προσφέρονται, ένα κουμπί πίσω και ένα κουμπί οκ. Πέρα από τον constructor που αρχικοποιεί όλες τις παραπάνω τιμές υπάρχει και η μέθοδος handle αφού η κλάση αυτή κάνει implement το interface EventHandler. Στην μέθοδο handle αρχικά παίρνουμε την υπάρχουσα τιμή του label display που περιέχει τον αριθμό κυκλοφορίας και έπειτα αποθηκεύουμε σε ένα κουμπί το κουμπί με το οποίο μπήκαμε στον eventhandler, κάτι που είναι απαραίτητο καθώς όλα τα κουμπιά του πληκτρολογίου χρησιμοποιούν τον ίδιο eventhandler. Αν το κείμενο του κουμπιού έχει μήκος 1 χαρακτήρα, αυτό σημαίνει πως το κουμπί είναι είτε γράμμα είτε αριθμός, συνεπώς απλά προσθέτουμε τον χαρακτήρα του κουμπιού στο κείμενο του label preview και ανανεώνουμε το label με το νέο κείμενο. Σε περίπτωση που δεν ισχύει αυτό, έχουμε ορίσει σε κάθε ένα από τα 3 “special” κουμπιά ένα ID το οποίο χρησιμοποιούμε για να αναγνωρίσουμε ποιο είναι αυτό που πατήθηκε. Αν είναι το backspace, τότε απλά αφαιρούμε από το κείμενο τον τελευταίο χαρακτήρα με χρήση της substring. Αν είναι το space απλά προσθέτουμε έναν κενό χαρακτήρα στο string και ανανεώνουμε το κείμενο του label. Εναλλακτικά, αν είναι το enter συμβαίνει μια ακολουθία πραγμάτων. Αρχικά ελέγχεται η εγκυρότητα όπως ζητήθηκε στην εκφώνηση. Αν ο αριθμός κυκλοφορίας είναι λιγότερο από 2 ψηφία τότε πετάγεται στον χρήστη pop up alert με μήνυμα λάθους που του εξηγεί πως ο αριθμός πρέπει να περιέχει παραπάνω από δύο ψηφία. Ο χρήστης δεν μπορεί να γυρίσει στην εφαρμογή μέχρι να πατήσει οκ και να αποδεχτεί το μήνυμα αυτό. Αν όλα έχουν πάει σωστά με τον αριθμό κυκλοφορίας, τότε πετάγεται ένα διαφορετικό pop up window που προτρέπει τον χρήστη να

διαλέξει τον τύπο του οχήματος του. Στην συνέχεια διαβάζονται όλες οι διαθέσιμες υπηρεσίες από ένα αρχείο και αποθηκεύονται σε μια δομή servicemenu την οποία θα αναλύσουμε παρακάτω. Ανάλογα με τον τύπο οχήματος που έχει επιλεχθεί, από την service menu αποθηκεύονται οι διαθέσιμες υπηρεσίες για αυτό το όχημα σε έναν πίνακα και σε έναν άλλον πίνακα αποθηκεύονται τα ID τους. Οι υπηρεσίες που έχουν αποθηκευτεί τους πίνακες αυτούς μαζί με τις τιμές τους ανατίθενται σε checkboxes και αυτά τοποθετούνται σε δύο στήλες πάνω στην οθόνη, ανεξάρτητα από το πόσες είναι οι υπηρεσίες. Αφότου μπουν όλα τα κουμπιά, checkboxes και labels στην θέση τους γίνεται ανάθεση eventhandlers. Για όλα τα checkboxes χρησιμοποιείται ο ίδιος handler που θα αναλύσουμε παρακάτω. Για το κουμπί back υλοποιείται ξεχωριστός eventhandler ο οποίος μηδενίζει το συνολικό ποσό προς πληρωμή στο label price και αδειάζει το label display που περιέχει τον αριθμό κυκλοφορίας. Μετά εξαφανίζει όλα τα services και ξαναβάζει το πρώτο scene στο stage. Το κουμπί οκ επίσης έχει ξεχωριστό eventhandler. Όταν πατηθεί πετάγεται ένα παράθυρο επιβεβαίωσης που ενημερώνει τον χρήστη για την τελική τιμή. Αν πατήσει άκυρο γυρνάει στην οθόνη με τις υπηρεσίες, ενώ αν πατήσει οκ ανοίγει σύνδεση με τον server με χρήση συναρτήσεων της κλάσης functionality, αποστέλλονται τα απαραίτητα δεδομένα σε μορφή String και έπειτα ακολουθεί η ίδια διαδικασία με αυτά που έγιναν στο κουμπί back. Αφότου ανατεθούν όλα τα κουμπιά και checkboxes στις σωστές θέσεις δίνεται η το stage στο scene2.

CheckboxHandler.java

Η κλάση αυτή είναι η κλάση όπου υλοποιείται ο handler για τα checkboxes με τις υπηρεσίες. Παίρνει ως ορίσματα το αποτέλεσμα της επιλογής οχήματος, το μενού με τις υπηρεσίες, τον πίνακα με τα id, μια μεταβλητή int με το όνομα index που δείχνει ποιο checkbox από τον πίνακα με τα checkboxes είναι αυτό που πατήθηκε. Αυτή η μεταβλητή ήταν αναγκαία καθώς δεν επιτρεπόταν μέσα στον eventhandler να περάσουμε μεταβλητές που έχουν ορισθεί προηγουμένως και δεν είναι final. Επίσης παίρνει τον πίνακα από τις υπηρεσίες ως checkboxes, το label με την συνολική τιμή των επιλεγμένων υπηρεσιών και τέλος τον αριθμό των διαθέσιμων υπηρεσιών ανάλογα το όχημα. Στην μέθοδο handle, ανάλογα τον τύπο του οχήματος που επιλέχθηκε παίρνουμε την κατάλληλη τιμή για την υπηρεσία μας από την δομή με τις υπηρεσίες. Αν μόλις έχουμε επιλέξει το checkbox, τότε από την μεταβλητή με την τιμή παίρνουμε το τωρινό κόστος και σε αυτό προσθέτουμε αυτό της επιλεγμένης υπηρεσίας. Επιπλέον γίνεται έλεγχος για το ποιες υπηρεσίες δεν είναι συμβατές με την υπηρεσία που επιλέξαμε και τις κάνουμε disable ώστε να μην μπορεί ο χρήστης να τις επιλέξει. Αν μόλις έχουμε κάνει uncheck το checkbox, τότε γίνεται η ίδια διαδικασία αλλά αφαιρούμε από το σύνολο την τιμή της υπηρεσίας. Έπειτα γίνεται διπλός έλεγχος. Αρχικά ξανα ενεργοποιούμε όσες υπηρεσίες είχαν απενεργοποιηθεί λόγω ασυμβατότητας με αυτήν. Έπειτα ελέγχουμε ποιες υπηρεσίες είναι επιλεγμένες και ξανά απενεργοποιούμε όλες αυτές που δεν είναι συμβατές μαζί τους, καθώς μπορεί κάποια από αυτές που ενεργοποιήσαμε ξανά να άνηκε σε αυτήν την κατηγορία.

Service.java

Η κλάση Service είναι μια κλάση αναπαράστασης πληροφορίας. Σε αυτήν αποθηκεύονται οι υπηρεσίες, κάθε μια ξεχωριστά. Συγκεκριμένα αποθηκεύονται το id της υπηρεσίας, το όνομα, οι τιμές για τους διάφορους τύπου οχημάτων και τα id των μη συμβατών υπηρεσιών με αυτή. Μέσα στον constructor γίνεται έλεγχος ώστε να μην υπάρξει αρνητική τιμή, με εξαίρεση το -1 που χρησιμοποιείται για να δηλώσει πως η υπηρεσία δεν υποστηρίζεται για κάποιον τύπο οχήματος. Αν δεν είναι αποδεκτή η τιμή τότε πετάει InvalidPriceException. Πέραν αυτού υπάρχουν οι μέθοδοι set και get για κάθε πεδίο καθώς και μια μέθοδος toString.

InvalidPriceException.java

Μια κλάση εξαίρεσης που χρησιμοποιείται στο Service.java σε περίπτωση εισαγωγής αρνητικής τιμής. Έχει το κατάλληλο μήνυμα λάθους.

ServiceMenu.java

Η κλάση ServiceMenu είναι και αυτή μια κλάση αναπαράστασης πληροφορίας. Αποτελείται από έναν HashMap στον οποίο αποθηκεύονται τα διάφορα services που παίρνουμε από το αρχείο. Μέσα στην κλάση αυτή υπάρχει η μέθοδος setFile που παίρνει ως όρισμα το path ενός αρχείου. Από εκεί με χρήση Scanner (ρυθμισμένο σε UTF-8 για να καταλαβαίνει ελληνικά) διαβάζονται γραμμή γραμμή οι υπηρεσίες από το αρχείο. Γίνονται οι κατάλληλες μετατροπές από το String που διαβάστηκε για τα πεδία ενός Service και τέλος εισάγεται αυτό το service στο HashMap με χρήστη της μεθόδου putService και κλείνει το αρχείο. Η putService λαμβάνει ως όρισμα μια μεταβλητή τύπου service και την προσθέτει απλά στο HashMap, καθώς και αυξάνει την μεταβλητή που κρατάει το id του κάθε στοιχείου στο hashmap κατά 1. Η μέθοδος print map εκτυπώνει για κάθε κλειδί του χάρτη τα στοιχεία της αντίστοιχης υπηρεσίας. Η μέθοδος getPriceService ανάλογα με το id της υπηρεσίας και τον τύπο του οχήματος επιστρέφει το κόστος της υπηρεσίας. Η μέθοδος getServiceById δέχεται ως όρισμα το id μιας υπηρεσίας στο hashmap και την επιστρέφει και τέλος η getNumberOfServices επιστρέφει τον αριθμό των υπηρεσιών αποθηκευμένο στην δομή.

Functionality.java

Η κλάση αυτή είναι που ορίζει τις συναρτήσεις που αφορούν την σύνδεση και μεταφορά δεδομένων από Client σε Server και είναι μια κλάση που αφορά καθαρά την λειτουργικότητα. Περιέχει 2 στατικές συναρτήσεις, την client connect που δέχεται ως όρισμα ένα ip και ένα port επιστρέφει ένα Socket το οποίο έχει συνδέσει με τον σέρβερ με την χρήση του constructor του Socket. Αν δεν γίνει σύνδεση επιστρέφει null. Επιπλέον περιέχει την συνάρτηση sendStringToServer η οποία δέχεται ως ορίσματα ένα Socket και ένα string. Με βάση το που είναι συνδεδεμένο το socket αυτό φτιάχνει εσωτερικά έναν PrintWriter για το Server και στέλνει σε αυτόν το String.

SeverThread.java

Η κλάση αυτή είναι η αντίστοιχη της Functionality αλλά για τον Σέρβερ. Η κλάση αυτή κληρονομεί την κλάση Thread και περιέχει δύο μεταβλητές, ένα ServerSocket και μια μεταβλητή τύπου Simplementation που είναι υπεύθυνη για την διαχείριση της λίστας που βλέπει ο χρήστης του προγράμματος Server καθώς και για το προσωρινό αρχείο στο οποίο αποθηκεύονται τα δεδομένα προς ακύρωση ή πληρωμή. Το thread αυτό έχει φτιαχτεί ώστε να τρέχει παράλληλα με το κύριο πρόγραμμα. Ακούσει συνεχώς στο port που έχει οριστεί από το εισαγόμενο ServerSocket και όποτε σταλθεί οτιδήποτε από τον Client, το παίρνει και το περνάει στην μέθοδο newclient της μεταβλητής τύπου Simplementation προκειμένου να προστεθεί νέα εγγραφή στην λίστα.

Server

Το πρόγραμμα του ταμείου υλοποιήθηκε κατά κύριο λόγο σε τρείς βασικές κλάσεις: την ServerUI.java , την SImplementation.java και την IncomeBook.java.

ServerUI

Η ServerUI.java διαχειρίζεται κυρίως το εμφανισιακό κομμάτι του προγράμματος και τις ενέργειες των κουμπιών. Οι μέθοδοι που υλοποιούνται σε αυτή είναι η CreateTable(), η Buttons(), η ButtonActions(), η ConfirmAlert(), η DeleteAlert(),η CloseAlert(), η DeleteSelected() , η ListCheck() και η Structure().

Η CreateTable() δημιουργεί το TableView το οποίο εμφανίζεται στο παράθυρο του διαχειριστή. Τα στοιχεία του ενημερώνονται αυτόματα από μία ObservableList η οποία περιέχει αντικείμενα ClientData στα οποία αποθηκεύονται ο αριθμός κυκλοφορίας, οι υπηρεσίες, το κόστος υπηρεσιών και η ημερομηνία άφιξης του κάθε πελάτη. Στον πίνακα ωστόσο, εμφανίζονται μόνο ο αριθμός κυκλοφορίας, το κόστος υπηρεσιών και η ημερομηνία άφιξης του.

Η Buttons() δημιουργεί τα κουμπιά της “Πληρωμής” και της “Ακύρωσης” και καλεί την μέθοδο ButtonActions() για την υλοποίηση αυτών.

Η ButtonActions() όπως αναφέρθηκε, διαχειρίζεται όλες τις ενέργειες των κουμπιών: “Πληρωμής”, “Ακύρωσης” , κλείσιμο παραθύρου και το πάτημα του πλήκτρου Enter. Ανάλογα με την ενέργεια που θέλει να πραγματοποιήσει καλεί και τις αντίστοιχες μεθόδους ConfirmAlert(),DeleteAlert() ή CloseAlert(). Επιπλέον ελέγχει αν κάποιο στοιχείο του πίνακα είναι πατημένο ή όχι και αντίστοιχα ενεργοποιεί ή απενεργοποιεί τα κουμπιά.

Η ConfirmAlert() καλείται στις περιπτώσεις που έχει πατηθεί το κουμπί “Πληρωμή” ή το πλήκτρο Enter. Εκεί, δημιουργείται ένα alert στο οποίο επιβεβαιώνει ο χρήστης αν θέλει ή όχι να επιβεβαιώσει την πληρωμή. Σε περίπτωση επιβεβαίωσης, καλείται η μέθοδος AddClientToIncomeBook η οποία προσθέτει το επιλεγμένο όχημα στο βιβλίο εσόδων (Η μέθοδος αυτή αναλύεται παρακάτω). Στη συνέχεια, καλείται η μέθοδος DeleteSelected() και η μέθοδος UpdateTempFile() οι οποίες διαγράφουν το όχημα από τον πίνακα, και αυτόματα και από την λίστα, και από ένα προσωρινό αρχείο αντίστοιχα.Τέλος καλείται η ListCheck() η οποία ελέγχει αν ο πίνακας είναι κενός.

Η DeleteAlert() καλείται στην περίπτωση που έχει πατηθεί το κουμπί “Ακύρωση”. Εκεί, δημιουργείται ένα alert στο οποίο επιβεβαιώνει ο χρήστης αν θέλει ή όχι ακυρώσει το συγκεκριμένο όχημα. Σε περίπτωση επιβεβαίωσης, καλείται η μέθοδος DeleteSelected() και η μέθοδος UpdateTempFile() οι οποίες διαγράφουν το όχημα από τον πίνακα, και αυτόματα και από την λίστα, και από ένα προσωρινό αρχείο αντίστοιχα. Τέλος καλείται η ListCheck() η οποία ελέγχει αν ο πίνακας είναι κενός.

Η CloseAlert() καλείται στην περίπτωση που ο χρήστης θέλει να κλείσει το παράθυρο της εφαρμογής και υπάρχουν ακόμα οχήματα στον πίνακα.
Η DeleteSelected(),όπως αναφέρθηκε, διαγράφει ένα επιλεγμένο στοιχείο από τον πίνακα.

Η ListCheck(), ,όπως αναφέρθηκε, ελέγχει αν ο πίνακας είναι κενός.

Η Structure() είναι η βασική μέθοδος της δημιουργίας του παραθύρου. Καλεί τις μεθόδους CreateTable() και Buttons() και δημιουργεί την σκηνή.

SImplementation

Η SImpementation.java διαχειρίζεται κυρίως τα δεδομένα που δέχεται το πρόγραμμα ταμείου από το πρόγραμμα υποδοχής.Οι μέθοδοι που υλοποιούνται σε αυτή είναι η CreateClient(), η newClient(), η UpdateTempFile() και η TempToList().

Η CreateClient() δέχεται ένα String το οποίο με κατάλληλη μέθοδο το χωρίζει και δημιουργεί ένα αντικείμενο ClientData.

Η newClient() είναι η βασική μέθοδος που υλοποιεί την εισαγωγή του νέου πελάτη σε μία ObservableList<ClientData> και την ενημέρωση ενός προσωρινού αρχείου με τον νέο χρήστη.

Η UpdateTempFile() είναι η μέθοδος η οποία διαχειρίζεται ένα αρχείο Temporary.txt . “Διαβάζει” όλα τα στοιχεία της ObservableList<ClientData> στην οποία βρίσκονται οι πελάτες που δεν έχουν ακόμα πραγματοποιήσει πληρωμή και τα καταγράφει στο Temporary.txt. Η μέθοδος αυτή καλείται κάθε φορά που υπάρχει μια καινούρια εισαγωγή στην λίστα (newClient()) ή κάποια διαγραφή είτε λόγω ακύρωσης του οχήματος είτε λόγω επιβεβαίωσης πληρωμής.

Η TempToList() διαβάζει τις εγγραφές από το Temporary.txt αρχείο και τις περνάει στην ObservableList<ClientData>. Η μέθοδος αυτή καλείται στην έναρξη του προγράμματος και σκοπός της είναι να διαχειρίζεται την περίπτωση που το πρόγραμμα του ταμείου κλείσει αναπόφευκτα και δεν έχουν πραγματοποιηθεί όλες οι πληρωμές των οχημάτων.

IncomeBook

Η IncomeBook.java διαχειρίζεται το Βιβλίο Εσόδων.Οι μέθοδοι που υλοποιούνται σε αυτή είναι η AddClientToIncomeBook() και η Date().

Η AddClientToIncomeBook() δέχεται ένα αντικείμενο ClientData και το καταγράφει σε ένα αρχείο IncomeBook.txt προσθέτοντας του επίσης την ημερομηνία και ώρα της αναχώρησης του.Η μέθοδος αυτή καλείται όταν γίνεται επιβεβαίωση πληρωμής για ένα όχημα.

Η Date() γυρνάει την τοπική ημερομηνία και ώρα.

ClientData

Η ClientData.java είναι η κλάση η οποία κρατάει όλα τα στοιχεία του χρήστη: αριθμός πινακίδας,υπηρεσίες,συνολικό κόστος και επιπλέον την ημερομηνία εισαγωγής του. Περιέχει μόνο μεθόδους get() και set() καθώς και Date() η γυρνάει την τοπική ημερομηνία και ώρα.

ClientDataList

Η ClientDataList.java περιέχει την μέθοδο addClientToList η οποία εισάγει ένα ClientData στην ObservableList<ClientData> και καλείται κάθε φορά που “έρχεται” ένας νέος πελάτης (newClient())

Server

Η Server.java είναι η βασική κλάση που καλείται όταν τρέχει το πρόγραμμα. Αρχικά δημιουργείται η ObservableList η οποία περνάει ως παράμετρος σε όλους τους constructors των κλάσεων ServerUI.java , SImplementation.java και IncomeBook.java.Στην συνέχεια καλούνται οι μέθοδοι Structure για την δημιουργία του παραθύρου και TempToList για την εγγραφή των οχημάτων, που δεν έχει πραγματοποιηθεί πληρωμή, στον πίνακα. Στην συνέχεια δημιουργείται ένα socket για την επικοινωνία του με το πρόγραμμα ταμείου και ένα αντικείμενο ServerThread. Τέλος με την μέθοδο start() ξεκινάει να τρέχει ένα νέο thread.