ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Εξαμηνιαία Εργασία στο μάθημα

«Αντικειμενοστραφής Προγραμματισμός Ι»

2ου εξαμήνου, 2022

Καζάζης Γεώργιος, *it214124*

Στόχος

Ο στόχος της εργασίας είναι η δημιουργία ενός προγράμματος που να υλοποιεί ένα **σύστημα ενοικίασης κατοικιών AirBNB** (ή **JavaBnB**), με χρήση της γλώσσας **Java**. Θα πρέπει να εκμεταλλευτούμε τα πλεονεκτήματα που μας προσφέρει ο **αντικειμενοστραφής προγραμματισμός** μέσω της Java, εφαρμόζοντας την **Ενθυλάκωση** (**Encapsulation**), την **Κληρονομικότητα** (**Inheritance**), τον **Πολυμορφισμό** (**Polymorphism**) , και την **Αφαιρετικότητα** (**Abstraction**), όσο το δυνατόν πιο πολύ και όσο το δυνατόν πιο αποτελεσματικά.

Παραδοχές

Παρακάτω παρουσιάζονται κάποιες **παραδοχές** με στόχο να διευκολυνθεί η **κατανόηση** του κώδικα.

- Το **ΑΦΜ** και ο **αριθμός ταυτότητας** είναι **6-ψήφιοι**, και ελέγχονται με **Exception** και **If-statements** κατά το καλύτερο δυνατό τρόπο. Αν ο χρήστης εισάγει αριθμό με λιγότερα από 6 ψηφία, συμπληρώνονται **μηδενικά** στο τέλος. (ο 123 γίνεται 123000)
- Επειδή **Attributes** όπως το **όνομα**, **επίθετο**, **δήμος** κλπ, παίρνουν τιμή απ' τον χρήστη με τη χρήση της μεθόδου next() (της Scanner), αν ο χρήστης εισάγει αριθμό, θα εισαχθεί κανονικά χωρίς να εμφανιστεί κάποιο exception. Έτσι λοιπόν, ο "νέος ενοικιαστής" μπορεί να πάρει όνομα όπως "123". Είναι λοιπόν στην κρίση του χρήστη, αν θα δοθούν σωστά στοιχεία.
- Οι **ημερομηνίες** κατασκευάζονται και τροποποιούνται με τη χρήση της κλάσης Calendar, στην οποία επειδή οι μήνες **ξεκινούν απ' το μηδέν** (0: Γενάρης, 11: Δεκέμβρης), ότι τιμή βάζει ο χρήστης στη θέση του μήνα, αφορά ένα μήνα μετά. Δηλαδή αν εισάγει το '**6**' έχοντας στο μυαλό του τον **Ιούνιο**, το σύστημα καταλαβαίνει **Ιούλιο**.
- Εν συνεχεία, στις **ημερομηνίες** όπου ζητούνται (πχ όταν κάνει ο χρήστης κράτηση), θα πρέπει να τις εισάγει με σειρά **Έτος Μήνας Μέρα**, με **κενά** ανάμεσα. Αν αντί για κενό εισαχθεί **παύλα** ή **slash**, το σύστημα εμφανίζει **error**, το οποίο δυστυχώς **δεν έχω φροντίσει** να αντιμετωπίζει κάποιος **Exception handler**. (**Αποδεκτό**: 2022 6 1, **Μη αποδεκτό**: 2022/6/1)
- Ο τρόπος με τον οποίο χειρίζομαι την **κλάση Reservation**, είναι ο εξής:
  - Όταν δημιουργείται **νέα κατοικία**, δημιουργείται αυτόματα και **νέο Reservation**, το οποίο στα πεδία "**resBeginDate**" και "**resEndDate**", που αφορούν τις **ημερομηνίες κράτησης**, παίρνουν την ημερομηνία του **συστήματος**, της μέρας που δημιουργήθηκαν. Επίσης το πεδίο "**tenantVatNumber**", που αφορά το **ΑΦΜ** του ενοικιαστή, παίρνει την τιμή '**-1**', μιας και δεν υπάρχει ακόμα ενοικιαστής.
  - Έτσι μαζί με το **ArrayList** των κατοικιών, τρέχει και το **ArrayList** των κρατήσεων, και όταν κάποιος κάνει κράτηση, γίνονται οι **απαραίτητοι έλεγχοι** για το αν η κράτηση θα γραφτεί πάνω σε ήδη υπάρχον στοιχείο του ArrayList, ή θα πρέπει να δημιουργηθεί **νέο αντικείμενο κράτησης** και να προστεθεί στο ArraList.
- **Δεν υλοποίησα** το να μπορεί, αυτόματα, ένας ενοικιαστής να καταχωρήσει μια κατοικία, ή ένας ιδιοκτήτης να νοικιάσει κάποια κατοικία, οπότε αν πρόκειται να συμβεί κάτι τέτοιο, το σύστημα **δεν θα καταλάβει** ότι το ΑΦΜ του ιδιοκτήτη, ή του ενοικιαστή αντίστοιχα, υπάρχει ήδη, με αποτέλεσμα να **χρειαστεί μια εγγραφή**.
- Το αν ο ενοικιαστής εισάγει **ρεαλιστικό email**, ελέγχεται με ένα **do…while loop**, όπου για να θεωρηθεί **έγκυρο** θα πρέπει να περιέχει μέσα του ένα από τα παρακάτω **Strings**: "**@hotmail.com**", "**@hotmail.gr**", "**@gmail.com**", "**@gmail.gr**" , "**@hua.gr**"

Βέβαια θα περάσει ακόμα κι αν ο χρήστης εισάγει "b**lah@gmail.grblahblah**" ή "**blah@gmail.gr@hotmail.com**" , αλλά αυτό είναι **καλύτερο** που μπορούσα να κάνω **χωρίς να συμβουλευτώ το διαδίκτυο**.

- Όπου ζητείται **διεύθυνση**, ο χρήστης πρέπει είτε να εισάγει **Οδό** και **αριθμό** (με **κενό ενδιάμεσα**) και να πατήσει **Enter**, είτε να εισάγει **Οδό**, να πατήσει **Enter** και ύστερα να εισάγει **αριθμό** και πάλι **Enter**.
- Όταν ο χρήστης επιλέξει να **τροποποιήσει** ή να **ακυρώσει** μία **κράτηση** του, θα πρέπει να **θυμάται το Reservation Code του**, γιατί **δεν κατάφερα** να το κάνω να εμφανίζονται οι κρατήσεις του την ώρα που του ζητείται να εισάγει το **Reservation Code**.

Class Analysis

Στο πρόγραμμα αυτό κατασκευάστηκαν και χρησιμοποιήθηκαν **8 κλάσεις**. Η μία κλάση είναι η **βασική κλάση** (**it214124**) που περιέχει τη **main**. Στην αρχή αυτής δημιουργούνται τα **ArrayList**, καθώς και **αντικείμενα κάθε κλάσης** (που θα αναφερθούν παρακάτω), ώστε να εισαχθούν στα αντίστοιχα **ArrayList**. Αυτό συμβαίνει ώστε να μπορεί ο χρήστης να εκτελέσει όποια διαδικασία θέλει από το **Menu**, μιας και δεν δουλεύουμε με **files** οπότε μετά τον τερματισμό του προγράμματος, **δεν αποθηκεύεται** οποιαδήποτε αλλαγή έχει γίνει κατά την εκτέλεση.
Πιο κάτω στην κλάση αυτή, εντός της **main**, θα βρούμε το **menu**, στο οποίο παρατηρείται το χαρακτηριστικό της **Αφαιρετικότητας** (**Abstraction**), μιας και φαίνονται **μόνο οι μέθοδοι** που αντιστοιχούν σε κάθε επιλογή, και **όχι η υλοποίησή** τους. Ο χρήστης επιλέγει έναν αριθμό από **1** έως και **5**, όπου **1. Καταχώρηση κατοικίας** **2. Καταχώρηση κράτησης**
3. Τροποποίηση/Ακύρωση κράτησης **4. Εμφάνιση αναφορών** **5. Έξοδος.**

Επόμενη κλάση είναι η **Residence** (**Κατοικία**), η οποία περιέχει τα απαιτούμενα κατά την εκφώνηση **Attributes**, τον **Constructor**, τους **setters & getters**, και από βασικές μεθόδους έχει την **getVerificID()** που δημιουργεί τον **αριθμό ταυτοποίησης της κατοικίας** σε μορφή **ΧΧ-ΑΑΑΑ**, και την **getReservationCode()** που δημιουργεί τον **κωδικό κράτησης** σε μορφή **ΧΧ-ΕΤΟΣ-ΚΚΚΚΚ**.

Ύστερα έχουμε τις κλάσεις **Apartment** (**Διαμέρισμα**) και **DetatchedHouse** (**Μονοκατοικία**), οι οποίες είναι **υποκλάσεις** της **Residence**, και έτσι κληρονομούν τα **Attributes** της και τις **μεθόδους** της. Εδώ λοιπόν χρησιμοποιούμε το χαρακτηριστικό της **Κληρονομικότητας**. Η **DetatchedHouse**, εκτός από **Constructor** και **Setters/Getters**, έχει την **toString()** και **toView()** κάνοντας χρήση του **@Override**, διότι υπάρχουν και στην **Residence**. Το ίδιο συμβαίνει και στην **Apartment**.

Στην συνέχεια έχουμε τις κλάσεις **Owner** (**Ιδιοκτήτης**) και **Tenant** (**Ενοικιαστής**), οι οποίες έχουν τα **Attributes** που αναφέρει η εκφώνηση, τον **Constructor** και τους **Setters/Getters**. Μέσα στην Owner, έχω βάλει την μέθοδο **registerResidence()** (επιλογή **1** του **Menu**), η οποία τσεκάρει αν υπάρχει ο ιδιοκτήτης. Αν υπάρχει τότε καλεί την μέθοδο **registerApartment()** ή **registerDetatchedHouse(),** ανάλογα την επιλογή του. Αν δεν υπάρχει, τότε καλείται η **registerOwner(),** ώστε να εισαχθεί **νέος ιδιοκτήτης** στο σύστημα.

Τέλος, έχουμε τις κλάσεις **Reservation** (**Κράτηση**) και **Backup** (**Εφεδρική**). Η **Backup** περιέχει κάποιες μεθόδους **γενικής φύσεως**, όπως η **printMenu(),** η **exitMenu()** , η **isIncorrectVat()** που ελέγχει αν το ΑΦΜ έχει περισσότερα από 6 ψηφία, και η **getReport()** που αφορά την επιλογή **4** του **Menu**.
Η **Reservation** έχει τα **Attributes** που ζητάει η εκφώνηση και που θα έπρεπε να έχει μία κράτηση. Από μεθόδους έχει την **totalCost()** που υπολογίζει το **συνολικό κόστος μίας κράτησης**, την **getReservationCode()** που δημιουργεί τον **κωδικό κράτησης**, ακριβώς όπως και στην **Residence class**, την **toString()** (με **@Override**), την **makeReservation()** η οποία αφορά την επιλογή **2** του **Menu**, μέσα στην οποία υπάρχει και η **δημιουργία νέου ενοικιαστή** αν αυτός δεν υπάρχει στο σύστημα, και την **changeReservation()** η οποία αφορά την επιλογή **3** του **Menu**, και μέσω της οποίας ο χρήστης μπορεί να **τροποποιήσει ημερομηνίες** των **κρατήσεων** του, ή να **ακυρώσει κάποια κράτηση** του.