ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

«αριθμός άσκησης»	Άσκηση 01
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	Λευτέρης Κοντούρης – Π19077
	Μιχάλης Στυλιανίδης – Π19165
	Κωνσταντίνος Καλογερόπουλος – Π19057
Ημερομηνία παράδοσης	17/04/2021

Εκφώνηση της άσκησης

Στόχοι άσκησης: *Βασικές έννοιες, δημιουργία κλάσεων, κληρονομικότητα, πολυμορφισμός, constructors, προσδιοριστές πρόσβασης, static, final, this keyword. Είσοδος από το πληκτρολόγιο, χειρισμός και δημιουργία εξαιρέσεων, χειρισμός εισόδου-εξόδου.*

Σε αυτή την άσκηση θα δημιουργήσετε ένα project το οποίο θα περιλαμβάνει τις βασικές κλάσεις που θα χρησιμοποιηθούν σε μία εφαρμογή *διαχείρισης ραντεβού τακτικών ιατρικών εξετάσεων*. Θα δημιουργήσετε τις βασικές κλάσεις που θα χρειαστούν και τη βασική λειτουργικότητά τους.

Αναλυτικά Βήματα:

1. (προκαταρτικά βήματα) Έλεγχος εγκατάστασης και καλής λειτουργίας περιβάλλοντος

    1.1. Έλεγχος εγκατάστασης jdk, έλεγχος εγκατάστασης eclipse (ή άλλου IDE της επιλογής σας), έλεγχος ενσωμάτωσης java documentation, έλεγχος δημιουργίας και εκτέλεσης ενός απλού προγράμματος.

2. Εκκίνηση project

    2.1. Δημιουργία ενός Java Project με τίτλο JavaBasics.

    2.2. Δημιουργία ενός πακέτου με τίτλο mainpackage

3. Δημιουργία βασικών κλάσεων

    3.1. Δημιουργία μίας κλάσης Users, η οποία θα περιλαμβάνει τα βασικά χαρακτηριστικά κάθε κατηγορίας χρήστη της εφαρμογής.

    - Χαρακτηριστικά: όνομα χρήστη (username), κωδικό (password), όνομα (name), επίθετο (surname), και άλλα στοιχεία, και τέλος έναν μετρητή χρηστών (usersCounter) ο οποίος θα ξεκινά από την τιμή 0 και θα αυξάνεται κατά ένα, όταν θα καλείται ο constructor της κλάσης (υπόδειξη: ποιος ορισμός counter ώς static μεταβλητής)
    - Μέθοδοι: Σύνδεση / Αποσύνδεση (login(), logout()). Τυπικοί getters και setters για όλες τις μεταβλητές (υπόδειξη: είναι δυνατό να δημιουργηθούν για όλες τις μεταβλητές;)
    - Δημιουργία τυπικού constructor για την κλάση Users.

    3.2. Δημιουργία υποκλάσης ασθενούς (Patient), η οποία θα χρησιμοποιεί τα χαρακτηριστικά και τις μεθόδους της κλάσης Users αλλά επιπλέον:

- Χαρακτηριστικά: Επιπρόσθετα θα χρησιμοποιεί το χαρακτηριστικό αριθμός ΑΜΚΑ. Το χαρακτηριστικό αυτό θα δίδεται μία φορά για κάθε αντικείμενο Patient και στη συνέχεια δεν θα μπορεί να αλλάξει.
- Ο constructor θα πρέπει να τροποποιηθεί κατάλληλα ώστε να περιλαμβάνει τον αριθμό ΑΜΚΑ, τον οποίο θα αρχικοποιεί τη στιγμή της δημιουργίας κάθε αντικειμένου τύπου Patient (Υπόδειξη: κάνετε override τον constructor της υπερκλάσης Users)
- Μέθοδοι της κλάσης: Εγγραφή χρήστη (registration) και άλλες μεθόδους που απαιτούνται, όπως ενδεικτικά, αναζήτηση διαθέσιμου ραντεβού για συγκεκριμένο ιατρό, αναζήτηση ραντεβού για οποιονδήποτε διαθέσιμο ιατρό μίας ειδικότητας, προβολή προγραμματισμένων ραντεβού, προβολή ιστορικού ραντεβού κτλ. Ποιες επιπλέον μεθόδους ή χαρακτηριστικά θα πρέπει να δημιουργήσετε για την κλάση Patient;

3.3. Δημιουργία υποκλάσης ιατρού (Doctor), με επιπλέον χαρακτηριστικό την ειδικότητα (specialty). Ποια επιπλέον χαρακτηριστικά και ποιες νέες μεθόδους χρειάζεται η κλάση Doctor; Να δημιουργηθούν αντίστοιχα. (Ενδεικτικές μέθοδοι: καταχώρηση διαθεσιμότητας ιατρού για ραντεβού ανά μήνα, προβολή προγράμματος ραντεβού κτλ)

3.4. Δημιουργία υποκλάσης διαχειριστή (Admin). Ο διαχειριστής θα εισαγάγει στο σύστημα τους Ιατρούς. Ποια επιπλέον χαρακτηριστικά και ποιες νέες μεθόδους χρειάζεται η κλάση Admin; Να δημιουργηθούν αντίστοιχα.

3.5. Δημιουργία της κλάσης ραντεβού (Appointment), η οποία θα χρησιμοποιείται για την καταγραφή ενός ραντεβού. Η κλάση αυτή ενδέχεται να χρησιμοποιεί ως χαρακτηριστικά αντικείμενα των κλάσεων Patient και Doctor.

3.6. Δημιουργία άλλων πιθανώς κλάσεων που είναι αναγκαίες για την εφαρμογή σύμφωνα με το σχεδιασμό σας.

4. Δοκιμή και τεκμηρίωση των βασικών κλάσεων

4.1. Δημιουργήστε μία κλάση με όνομα CreateUsers η οποία θα περιλαμβάνει τη συνάρτηση main(). Μέσω της κλάσης αυτής θα δημιουργήσετε αντικείμενα τύπου Patient, Doctor, Admin κτλ. Δημιουργήστε ένα αντικείμενο από κάθε κλάση και χρησιμοποιείστε τις μεθόδους που έχετε δημιουργήσει. Με τη χρήση ενδεικτικών μηνυμάτων, δείξτε τη δημιουργία και τα χαρακτηριστικά των αντικειμένων.

4.2. Προσθέστε κατάλληλη τεκμηρίωση (documentation) για τις κλάσεις Patient, Doctor, Admin και Appointment και για όσες μεθόδους απαιτείται.

5. Έλεγχος εισόδου και χειρισμός εξαιρέσεων

5.1. Μέσω της κύριας κλάσης CreateUsers, χρησιμοποιείστε ένα αντικείμενο τύπου Scanner ώστε να δημιουργήσετε αντικείμενα τύπου Patient και Doctor, λαμβάνοντας τα ορίσματα του constructor από το πληκτρολόγιο. Χρησιμοποιείστε exceptions με τη βοήθεια των εντολών try…catch για να ελέγξετε την ορθότητα των παραμέτρων εισόδου.

5.2. Προσθέστε κατάλληλη τεκμηρίωση (documentation) όπου απαιτείται.

6. Δημιουργία αντικειμένων με τη χρήση αρχείου εισόδου

6.1. Δημιουργείστε ένα txt αρχείο το οποίο περιλαμβάνει τα χαρακτηριστικά ενός αντικειμένου τύπου Patient (με κενά μεταξύ των χαρακτηριστικών). Δημιουργήστε ένα αντικείμενο, διαβάζοντας τα χαρακτηριστικά του από αυτό το αρχείο.

6.2. Χρησιμοποιείστε το μηχανισμό εξαιρέσεων για να ελέγξετε για πιθανά σφάλματα ανάγνωσης, εγγραφής, και σωστού τύπου των χαρακτηριστικών.

7. Διάγραμμα Κλάσεων

7.1. Περιλάβετε στο παραδοτέο σας το διάγραμμα κλάσεων (Class Diagram).

- Χαρακτηριστικά: Επιπρόσθετα θα χρησιμοποιεί το χαρακτηριστικό αριθμός ΑΜΚΑ. Το χαρακτηριστικό αυτό θα δίδεται μία φορά για κάθε αντικείμενο Patient και στη συνέχεια δεν θα μπορεί να αλλάξει.

- Ο constructor θα πρέπει να τροποποιηθεί κατάλληλα ώστε να περιλαμβάνει τον αριθμό ΑΜΚΑ, τον οποίο θα αρχικοποιεί τη στιγμή της δημιουργίας κάθε αντικειμένου τύπου Patient (Υπόδειξη: κάνετε override τον constructor της υπερκλάσης Users)

- Μέθοδοι της κλάσης: Εγγραφή χρήστη (registration) και άλλες μεθόδους που απαιτούνται, όπως ενδεικτικά, αναζήτηση διαθέσιμου ραντεβού για συγκεκριμένο ιατρό, αναζήτηση ραντεβού για οποιονδήποτε διαθέσιμο ιατρό μίας ειδικότητας, προβολή προγραμματισμένων ραντεβού, προβολή ιστορικού ραντεβού κτλ. Ποιες επιπλέον μεθόδους ή χαρακτηριστικά θα πρέπει να δημιουργήσετε για την κλάση Patient;

7.2. Δημιουργία υποκλάσης ιατρού (Doctor), με επιπλέον χαρακτηριστικό την ειδικότητα (specialty). Ποια επιπλέον χαρακτηριστικά και ποιες νέες μεθόδους χρειάζεται η κλάση Doctor; Να δημιουργηθούν αντίστοιχα. (Ενδεικτικές μέθοδοι: καταχώρηση διαθεσιμότητας ιατρού για ραντεβού ανά μήνα, προβολή προγράμματος ραντεβού κτλ)

7.3. Δημιουργία υποκλάσης διαχειριστή (Admin). Ο διαχειριστής θα εισαγάγει στο σύστημα τους Ιατρούς. Ποια επιπλέον χαρακτηριστικά και ποιες νέες μεθόδους χρειάζεται η κλάση Admin; Να δημιουργηθούν αντίστοιχα.

7.4. Δημιουργία της κλάσης ραντεβού (Appointment), η οποία θα χρησιμοποιείται για την καταγραφή ενός ραντεβού. Η κλάση αυτή ενδέχεται να χρησιμοποιεί ως χαρακτηριστικά αντικείμενα των κλάσεων Patient και Doctor.

7.5. Δημιουργία άλλων πιθανώς κλάσεων που είναι αναγκαίες για την εφαρμογή σύμφωνα με το σχεδιασμό σας.

8. Δοκιμή και τεκμηρίωση των βασικών κλάσεων

8.1. Δημιουργήστε μία κλάση με όνομα CreateUsers η οποία θα περιλαμβάνει τη συνάρτηση main(). Μέσω της κλάσης αυτής θα δημιουργήσετε αντικείμενα τύπου Patient, Doctor, Admin κτλ. Δημιουργήστε ένα αντικείμενο από κάθε κλάση και χρησιμοποιείστε τις μεθόδους που έχετε δημιουργήσει. Με τη χρήση ενδεικτικών μηνυμάτων, δείξτε τη δημιουργία και τα χαρακτηριστικά των αντικειμένων.

8.2. Προσθέστε κατάλληλη τεκμηρίωση (documentation) για τις κλάσεις Patient, Doctor, Admin και Appointment και για όσες μεθόδους απαιτείται.

9. Έλεγχος εισόδου και χειρισμός εξαιρέσεων

9.1. Μέσω της κύριας κλάσης CreateUsers, χρησιμοποιείστε ένα αντικείμενο τύπου Scanner

ώστε να δημιουργήσετε αντικείμενα τύπου Patient και Doctor, λαμβάνοντας τα ορίσματα του constructor από το πληκτρολόγιο. Χρησιμοποιείστε exceptions με τη βοήθεια των εντολών try…catch για να ελέγξετε την ορθότητα των παραμέτρων εισόδου.

9.2. Προσθέστε κατάλληλη τεκμηρίωση (documentation) όπου απαιτείται.

10. Δημιουργία αντικειμένων με τη χρήση αρχείου εισόδου

10.1. Δημιουργείστε ένα txt αρχείο το οποίο περιλαμβάνει τα χαρακτηριστικά ενός αντικειμένου τύπου Patient (με κενά μεταξύ των χαρακτηριστικών). Δημιουργήστε ένα αντικείμενο, διαβάζοντας τα χαρακτηριστικά του από αυτό το αρχείο.

10.2. Χρησιμοποιείστε το μηχανισμό εξαιρέσεων για να ελέγξετε για πιθανά σφάλματα ανάγνωσης, εγγραφής, και σωστού τύπου των χαρακτηριστικών.

11. Διάγραμμα Κλάσεων

11.1. Περιλάβετε στο παραδοτέο σας το διάγραμμα κλάσεων (Class Diagram).

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

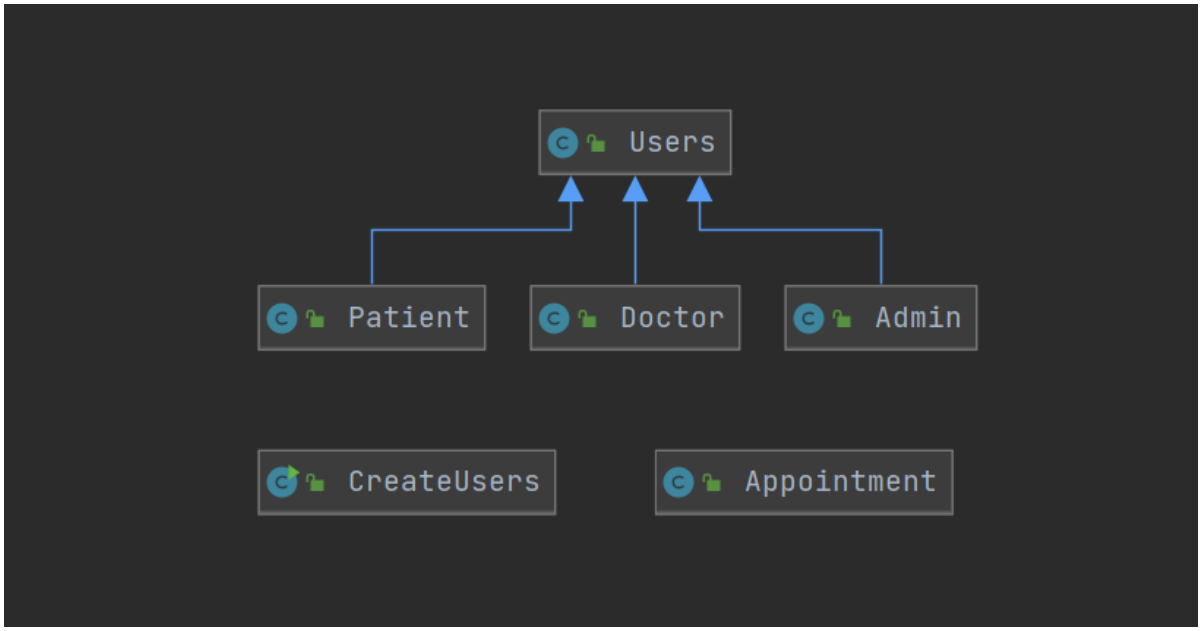
1. Γενική Περιγραφή της λύσης

Το πρόγραμμα αποτελείται από έξι κλάσεις εκ των οποίων οι τρεις από αυτές έχουν κάποια κοινά χαρακτηρίστηκα μέσω της κλάσης Users. Η Users είναι υπεύθυνη για την διαχείριση των δεδομένων ενός νέου χρήστη της εφαρμογής είτε αυτός είναι ασθενής, γιατρός ή διαχειριστής.

Η κλάσεις Patient, Doctor και Admin κληρονομούν από την κλάση Users τα στοιχεία ενός χρήστη και τα επεξεργάζονται περαιτέρω ανάλογα με τα επιπλέον χαρακτηριστικά που χρειάζεται. Για παράδειγμα ο γιατρός έχει επιπλέον χαρακτηριστικό την ειδικότητα του, ο ασθενής το αριθμό ασφάλισης του και ο διαχειριστής τα αναβαθμισμένα δικαιώματα του(permissions) σε σχέση με τους απλούς χρήστες εντός της εφαρμογής.

Η κλάση CreateUsers είναι υπεύθυνη για την αρχικοποίηση κάποιων δοκιμαστικών χρηστών και περιέχει την μέθοδο main η οποία καλείται στην εκτέλεση του προγράμματος.

Η κλάση Appointment διαχειρίζεται τα ραντεβού των ασθενών και αργότερα θα συνδεθεί με την βάση δεδομένων.



Εικόνα 1 Σχεδιάγραμμα κλάσεων

2. Κώδικας προγράμματος

Ακολουθεί η αναλυτική περιγραφή του προγράμματος.

1.1 Κώδικας της κλάσης Users

```
3   public class Users {
4
5       private String username;
6       private String password;
7       private String name;
8       private String surname;
9       private static int usersCounter = 0;
10
11      // Constructor
12      public Users(String username, String password, String name, String surname) {
13          this.username = username;
14          this.password = password;
15          this.name = name;
16          this.surname = surname;
17
18          usersCounter += 1;
19      }
```

Εικόνα 2 Constructor of User's class

```
20
21      // Getters and Setters
22  >   public String getUsername() {···
25
26  >   public void setUsername(String username) {···
29
30  >   public String getPassword() {···
33
34  >   public void setPassword(String password) {···
37
38  >   public String getName() {···
41
42  >   public void setName(String name) {···
45
46  >   public String getSurname() {···
49
50  >   public void setSurname(String surname) {···
53
54  >   public static int getUsersCounter() {···
57
```

Εικόνα 3 Getters and Setters

```
57
58      // Functionality methods
59      /**
60       * User login
61       */
62      public void login() { }
63
64      /**
65       * User logout
66       */
67      public void logout() { }
68  }
```

Εικόνα 4 Login() and Logout() Methods

1.2 Κώδικας της κλάσης Patient

```
6   public class Patient extends Users {
7
8       private final String amka;
9       private ArrayList<Appointment> scheduledAppointments = new ArrayList<>();
10
11      public Patient(String username, String password, String name, String surname, String amka) {
12          super(username, password, name, surname);
13          this.amka = amka;
14      }
15
16      public String getAmka() {
17          return amka;
18      }
19
20      public ArrayList<Appointment> getScheduledAppointments() {
21          return scheduledAppointments;
22      }
```

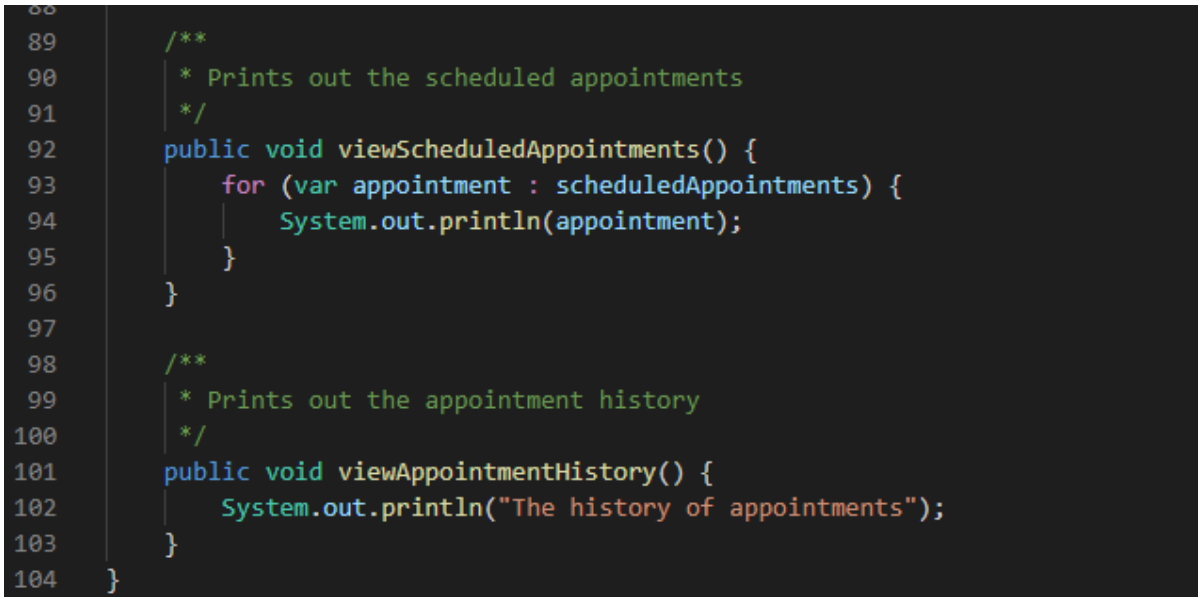
Εικόνα 5 Patient's constructor and variables initialization

```
23
24      /**
25       * Sets the appointment schedule
26       * @param scheduledAppointments
27       */
28      public void setScheduledAppointments(ArrayList<Appointment> scheduledAppointments) {
29          this.scheduledAppointments = scheduledAppointments;
30      }
31
32      /**
33       * Adds an appointment to the schedule
34       * @param appointment
35       */
36      public void addAppointment(Appointment appointment) {
37          scheduledAppointments.add(appointment);
38      }
39
40      /**
41       * Removes appointment from the list of scheduled appointments
42       * @param appointment
43       */
44      public void cancelAppointment(Appointment appointment) {
45          scheduledAppointments.remove(appointment);
46      }
```

Εικόνα 6 Set, add and cancel patient's appointment methods

```
48      /**
49       * Replaces the old appointment with the new one in the scheduled appointments list
50       * @param oldAppointment
51       * @param newAppointment
52       */
53      public void replaceAppointment(Appointment oldAppointment, Appointment newAppointment) {
54          if (scheduledAppointments.contains(oldAppointment)) {
55              scheduledAppointments.set(scheduledAppointments.indexOf(oldAppointment), newAppointment);
56          }
57      }
58
59      /**
60       * Registers a new user to the database
61       */
62      public void registerUser() {
63          System.out.println("User registered");
64      }
65
66      /**
67       * Finds the available appointments of the given doctor
68       * @return ArrayList of type Appointment
69       */
70      public ArrayList<Appointment> getAvailableAppointments(Doctor doctor) {
71          ArrayList<Appointment> availableAppointments = new ArrayList<Appointment>();
72
73          System.out.println("Found appointments");
74          return availableAppointments;
75      }
```

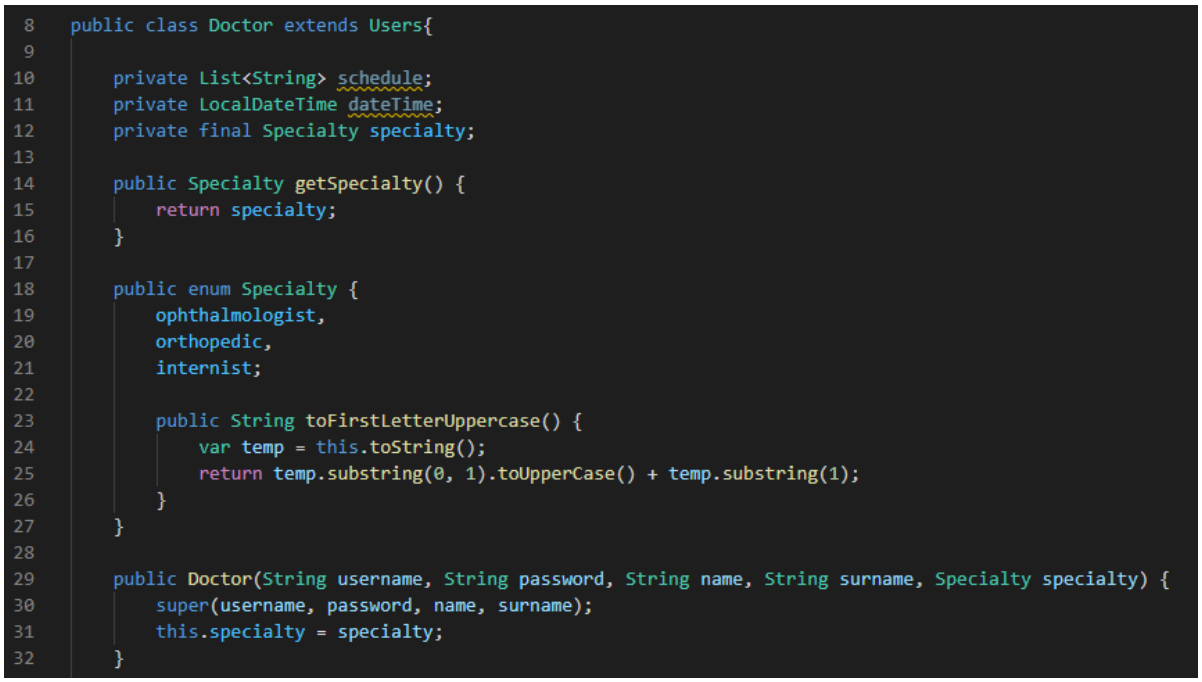
Εικόνα 7 Replace patient's appointment, register a new patient into the database, get available appointments



```
89      /**
90       * Prints out the scheduled appointments
91       */
92      public void viewScheduledAppointments() {
93          for (var appointment : scheduledAppointments) {
94              System.out.println(appointment);
95          }
96      }
97
98      /**
99       * Prints out the appointment history
100      */
101     public void viewAppointmentHistory() {
102         System.out.println("The history of appointments");
103     }
104 }
```

Εικόνα 8 View scheduled appointments and appointments' history

1.3 Κώδικας της κλάσης Doctor



```
8   public class Doctor extends Users{
9
10      private List<String> schedule;
11      private LocalDateTime dateTime;
12      private final Specialty specialty;
13
14      public Specialty getSpecialty() {
15          return specialty;
16      }
17
18      public enum Specialty {
19          ophthalmologist,
20          orthopedic,
21          internist;
22
23          public String toFirstLetterUppercase() {
24              var temp = this.toString();
25              return temp.substring(0, 1).toUpperCase() + temp.substring(1);
26          }
27      }
28
29      public Doctor(String username, String password, String name, String surname, Specialty specialty) {
30          super(username, password, name, surname);
31          this.specialty = specialty;
32      }
```

Εικόνα 9 Doctor's constructor, an enum called Specialty and other variables

```
34      /**
35       * Doctor inserts the date he is available to check in patients
36       */
37      public void insertDateAvailability(String doctorAvailability)
38      {
39          dateTime = LocalDateTime.now();
40          System.out.println("Available date added");
41      }
42
43      /**
44       * View doctor's appointment availability
45       */
46      public void viewAppointmentAvailability()
47      {
48          System.out.println("The doctor is available 24/7");
49      }
50
51      /**
52       * Cancel patient's appointment only if it's scheduled at least 3 days later compared to the current date.
53       */
54      public void cancelAppointment(Appointment appointmentToBeCancelled)
55      {
56          System.out.println("Appointment Cancelled");
57      }
58
59  }
```

Εικόνα 10 Insert availability, view availability and cancel scheduled appointments methods

1.4 Κώδικας της κλάσης Admin

```
3   public class Admin extends Users {
4
5       protected String superuserPassword;
6
7       public Admin(String username, String password, String superuserPassword, String name, String surname) {
8           super(username, password, name, surname);
9           this.superuserPassword = superuserPassword;
10      }
11
12      public String getSuperuserPassword() {
13          return superuserPassword;
14      }
15
16      public void setSuperuserPassword(String superuserPassword) {
17          this.superuserPassword = superuserPassword;
18      }
19
20      /**
21       * Inserts Doctor to database
22       */
23      public void InsertDoctor(Doctor doctor) {
24          System.out.println(doctor.getName() + " " + doctor.getSurname() + " inserted to database");
25      }
26
27      /**
28       * Removes Doctor from database
29       */
30      public void DeleteDoctor(Doctor doctor) {
31          System.out.println(doctor.getName() + " " + doctor.getSurname() + " removed from database");
32      }
33  }
```

Εικόνα 11 Admin's constructor and methods to insert and remove doctors into and from the database.

1.5 Κώδικας της κλάσης CreateUsers

```java
public class CreateUsers {

    /**
     * Handles user input
     */
    public static class UserInput {

        /**
         * Prints out the displayMessage and infinitely loops until the user gives a valid int value
         * that is between the provided bounds. If the input format is invalid the errorMessage is displayed
         * @param lowBound The lowest value that can be returned
         * @param upperBound The highest value that can be returned
         * @param displayMessage The message that will be printed at the start
         * @param errorMessage The message that will be printed in case of invalid input
         * @return The user input
         */
        public static int getIntInput(int lowBound, int upperBound, String displayMessage, String errorMessage) {
            int selection = 0;
            do
            {
                Scanner scanner = new Scanner(System.in);
                System.out.print(displayMessage);
                try {
                    selection = scanner.nextInt();
                    if (selection < lowBound || selection > upperBound) {
                        System.out.println("The given value is out of bounds. Type an integer between "
                                + lowBound + " and " + upperBound);
                    }
                } catch (InputMismatchException e) {
                    System.out.println(errorMessage);
                } catch (Exception e) {
                    System.out.println("An unusual error has occurred. Try again");
                }
            } while (selection < lowBound || selection > upperBound);
            return selection;
        }
```

Εικόνα 12 Handles user's numeric input with try..catch exceptions

```java
        /**
         * Prints out the displayMessage and infinitely loops until the user gives a valid input.
         * If the input format is invalid the errorMessage is displayed
         * @param displayMessage The message that will be printed at the start
         * @param errorMessage The message that will be printed in case of invalid input
         * @return The user input
         */
        public static String getStringInput(String displayMessage, String errorMessage) {
            while(true) {
                Scanner scanner = new Scanner(System.in);
                try {
                    System.out.print(displayMessage);
                    return scanner.nextLine();
                } catch (InputMismatchException e) {
                    System.out.println(e.getMessage());
                    System.out.println(errorMessage);
                } catch (Exception e) {
                    System.out.println("An unusual error has occurred. Try again");
                }
            }
        }
    }

```

Εικόνα 13 Handles user's string inputs

```
71      public static void main(String[] args) {
72
73          // Create sample objects of each class
74          Admin admin_user = new Admin("admin", "1234567890LT", "25670GTA!", "LEFTERIS", "KONTOURIS");
75          System.out.println("Admin: " + admin_user.getUsername() + ", "+ admin_user.getPassword() + ", " + admin_user.getSuperuserPassword()
76                             + ", " + admin_user.getName() + " " + admin_user.getSurname());
77          System.out.println("Admin created\n");
78
79          Doctor doctor_user = new Doctor("kostas2001", "26483dgdun", "KOSTAS", "KALOGEROPOULOS", Doctor.Specialty.internist);
80          System.out.println("Doctor: " + doctor_user.getUsername() + ", "+ doctor_user.getPassword() + ", " + doctor_user.getName() + " "
81                             + doctor_user.getSurname() + " " + doctor_user.getSpecialty().toFirstLetterUppercase());
82          System.out.println("Doctor created\n");
83
84          admin_user.InsertDoctor(doctor_user);
85          admin_user.DeleteDoctor(doctor_user);
86          System.out.print("\n");
87
88          Patient patient_user = new Patient("MichaelX26", "89054809HDHJ", "MIXALIS", "STYLIANIDIS", "280501014523");
89          System.out.println("Patient: " + patient_user.getUsername() + ", "+ patient_user.getPassword() + ", " + patient_user.getName()
90                             + " " + patient_user.getSurname() + ", " + patient_user.getAmka());
91          System.out.println("Patient created\n");
92
93          // Print a menu for choosing what type of user to add
94          System.out.println("-------Main Menu-------");
95          System.out.println("1. Doctor");
96          System.out.println("2. Patient");
97
98          // Select type
99          int entitySelection = UserInput.getIntInput(1, 3, "Choose entity to add (enter number): ", "Invalid input. Only integers are acceptable");
100
101         // Get input for the common attributes
102         String username = UserInput.getStringInput("Username: ", "Invalid username. Try again");
103         String password = UserInput.getStringInput("Password: ", "Invalid password. Try again");
104         String name = UserInput.getStringInput("Name: ","Invalid name. Try again");
105         String surname = UserInput.getStringInput("Surname: " ,"Invalid surname. Try again");
```

Εικόνα 14 Creates sample data for basic functionality of the program

```
106
107         if (entitySelection == 1)
108         {
109             // Create doctor
110             System.out.println("Select doctor specialty according to the following menu: ");
111
112             // Print all the possible specialties of the Specialty enum
113             var allDoctorSpecialties = Doctor.Specialty.values();
114             for (int i = 0; i < allDoctorSpecialties.length; i++) {
115                 System.out.println(i+1 + ". " + allDoctorSpecialties[i]);
116             }
117
118             // Ask the user to select a specialty
119             int selectedSpecialtyIndex = UserInput.getIntInput(1, allDoctorSpecialties.length, "Selection: ", "Invalid input. Only integers are acceptable");
120             selectedSpecialtyIndex--;
121
122             // Create a doctor object and use it's methods
123             Doctor doctor = new Doctor(username, password, name, surname, allDoctorSpecialties[selectedSpecialtyIndex]);
124             System.out.println(doctor.getSpecialty().toFirstLetterUppercase() + " created!");
125
126             doctor.insertDateAvailability("24/7");
127             doctor.viewAppointmentAvailability();
128             // doctor.cancelAppointment(anAppointment);
129         }
130         else
131         {
132             // Create patient
133             String amka = UserInput.getStringInput("AMKA: ", "Invalid AMKA. Try again");
134             Patient patient = new Patient(username, password, name, surname, amka);
135             System.out.println("Patient created!");
136
137             // Patient methods. Some of them have been commented out,
138             // because they require parameters that can't be provided at the moment
139             patient.registerUser();
140             //patient.getAvailableAppointments(aDoctor);
141             //patient.addAppointment(aNewAppointment);
142             //patient.cancelAppointment(theAppointmentToBeCancelled);
143             //patient.replaceAppointment(theOldAppointment, theNewAppointment);
144             //patient.setScheduledAppointments(aListOfAppointments);
145             patient.viewAppointmentHistory();
146         }
147
```

Εικόνα 15 Displays a menu to the user and handles the user's input accordingly

```
        // Read a patient from txt
        File file = new File(System.getProperty("user.dir") + "\\patient.txt");
        BufferedReader bufferedReader;

        try {
            bufferedReader = new BufferedReader(new FileReader(file));

            String str;
            while ((str = bufferedReader.readLine()) != null)
            {
                String[] fields = str.split("\\s+");
                Patient patient = new Patient(fields[0], fields[1], fields[2], fields[3], fields[4]);
                System.out.println("New patient created from text file:");
                System.out.println("Patient: " + patient.getUsername() + ", "+ patient.getPassword() + ", " + patient.getName()
                        + " " + patient.getSurname() + ", " + patient.getAmka());
            }
        } catch (FileNotFoundException e) {
            System.out.println("The specified file was not found");
        } catch (Exception e) {
            System.out.println("An unusual error has occurred while reading the file");
        }

    }
}
```

Εικόνα 16 Read patient's info directly from a text file by using bufferReader.

1.6 Κώδικας της κλάσης Appointment

```java
public class Appointment {

    private Patient patient;
    private Doctor doctor;
    private LocalDateTime DateTime;

    // Constructor
    public Appointment(Patient patient, Doctor doctor, LocalDateTime dateTime) {
        this.patient = patient;
        this.doctor = doctor;
        DateTime = dateTime;
    }

    // Getters and setters
    public Patient getPatient() {
        return patient;
    }

    public void setPatient(Patient patient) {
        this.patient = patient;
    }

    public Doctor getDoctor() {
        return doctor;
    }

    public void setDoctor(Doctor doctor) {
        this.doctor = doctor;
    }

    public LocalDateTime getDateTime() {
        return DateTime;
    }

    public void setDateTime(LocalDateTime dateTime) {
        DateTime = dateTime;
    }
}
```

Εικόνα 17 Appointment data handling

3. Εκτέλεση του προγράμματος

```
-------Main Menu-------
1. Doctor
2. Patient
Choose entity to add (enter number): 1
Username: Kostas01
Password: 123456
Name: Konstantinos
Surname: Kalogeropoulos
Select doctor specialty according to the following menu:
1. ophthalmologist
2. orthopedic
3. internist
Selection: 2
Orthopedic created!
Available date added
The doctor is available 24/7
```

Εικόνα 18 Adding a new doctor into the platform

```
-------Main Menu-------
1. Doctor
2. Patient
Choose entity to add (enter number): 2
Username: Jack
Password: 123456
Name: Jack
Surname: Sparrow
AMKA: 120021312
Patient created!
User registered
The history of appointments
```

Εικόνα 19 Adding a new patient into the platform

```
New patient created from text file:
Patient: Jack_o_piratis, ef4wufjh, Jack O_Piratis, 180461694081
```

Εικόνα 20 Adding a new patient by getting his info inserted into the txt file.