Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

«Αριθμός άσκησης»: 02	
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	Δημήτρης Κωτσόπουλος – Π21220
	Χαράλαμπος Σπυρόπουλος – Π21235
	Γιώργος Σπηλιόπουλος – Π21155
Ημερομηνία παράδοσης	26/06/2024

Εκφώνηση της άσκησης

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ & ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

Ακαδ. Έτος 2023-24

Άσκηση 02

Σε αυτή την άσκηση θα δημιουργήσετε ένα dynamic web project σε Java το οποίο θα αποτελέσει τον κορμό για την τελική εργασία. Σε αυτό το web project, θα χρησιμοποιήσετε τις κλάσεις που δημιουργήσατε στην προηγούμενη εργασία, με τις κατάλληλες τροποποιήσεις με τη χρήση servlet.

Στόχοι άσκησης: *web project, δημιουργία Βάσης Δεδομένων της εφαρμογής, υλοποίηση ορισμένων λειτουργιών.* Σε αυτή την άσκηση θα γίνουν τα εξής:

- ☐ *η εγκατάσταση του application server (π.χ. Tomcat, Glassfish) και του database server (π.χ. mysql ή postgress) και η μεταξύ τους διασύνδεση.*
- ☐ *η δημιουργία* του dynamic web project το οποίο θα αποτελέσει τον κορμό για την τελική εργασία και η υλοποίηση ορισμένων λειτουργιών.
- ☐ η υλοποίηση ορισμένων λειτουργιών των χρηστών της εφαρμογής.

Αναλυτικά Βήματα:

1 Εγκατάσταση και παραμετροποίηση application server και database server.

1.1 Εγκαταστήστε και παραμετροποιήστε τον application server (π.χ. Tomcat, Glassfish) και το Σύστημα Διαχείρισης Βάσης Δεδομένων (π.χ. mysql, postgress, sqlite). Η εγκατάσταση του Application Server να συνδεθεί με το προγραμματιστικό περιβάλλον IDE που χρησιμοποιείτε (π.χ. Eclipse).

1.2 Δημιουργήστε την σύνδεση του application server με τον database server, χρησιμοποιώντας τον αντίστοιχο jdbc database connector για το σύστημα βάσης της επιλογής σας. Χρησιμοποιήστε τη σύνδεση του μοντέλου 3-tier. (μπορείτε να βρείτε αντίστοιχο παράδειγμα στα παραδείγματα κώδικα που περιλαμβάνονται στη σελίδα του μαθήματος).

2 Δημιουργία Βάσης Δεδομένων

2.1 Δημιουργήστε το Μοντέλο Οντοτήτων-Σχέσεων, το οποίο περιγράφει τη Βάση Δεδομένων που θα χρησιμοποιήσετε για την εφαρμογή σας. Ενδεικτικά (και όχι περιοριστικά) θα περιλαμβάνει πίνακες όπως, Πελάτες, Πωλητές, Προγράμματα Τηλεφωνίας, Αριθμούς Τηλεφώνων, Κλήσεις κτλ. Να περιλάβετε στο μοντέλο σας τις σχέσεις μεταξύ των πινάκων.

2.2 Με τη βοήθεια του Μοντέλου Οντοτήτων-Σχέσεων, να δημιουργήσετε και να εκτανήσετε τη βάση στον database server. Μπορείτε να χρησιμοποιήσετε οποιοδήποτε βοηθητικό εργαλείο για την εξαγωγή της βάσης από το μοντέλο (π.χ. mysql Workbench για mysql).

2.3 Εισάγετε εικονικά δεδομένα σε όλους τους πίνακες, λαμβάνοντας υπόψη τα εξωτερικά κλειδιά που πιθανώς έχουν οι πίνακες. Για αυτό μπορείτε να χρησιμοποιήσετε βοηθητικά εργαλεία όπως τα [1],[2],[3].

3 Δημιουργία web project

3.1 Δημιουργήστε ένα Dynamic Web Project.

3.2 Δημιουργήστε ένα ή περισσότερα πακέτα κλάσεων, τα οποία θα περιλαμβάνουν τις βασικές κλάσεις που έχετε υλοποιήσει στην προηγούμενη άσκηση.

3.3 Δημιουργείστε ένα ή περισσότερα νέα πακέτα κλάσεων τα οποίο θα περιλαμβάνουν όλα τα servlet που θα χρησιμοποιήσετε στην εργασία (ενδεικτικά ClientServlet, SellerServlet, AdminServlet). Στην συγκεκριμένη άσκηση θα υλοποιήσετε μόνο ένα μέρος από αυτά, όπως αναφέρεται στο βήμα 5.

4 Δημιουργία διαδικτυακής διεπαφής

4.1 Σε αυτό το βήμα, θα υλοποιήσετε τη διαδικτυακή διεπαφή (html σελίδες) που θα χρησιμοποιούν οι χρήστες όλων των κατηγοριών (Πελάτες, Πωλητές, Διαχειριστές Εφαρμογής) για να αλληλεπιδρούν με την εφαρμογή και να χρησιμοποιούν τις αντίστοιχες μεθόδους που απαιτούνται.

4.2 Θα υπάρχει ένα κεντρικό μενού (π.χ.. μία σελίδα index.html), η οποία θα είναι η αρχική σελίδα για όλους τους χρήστες. Από την αρχική σελίδα οι διάφοροι χρήστες θα μπορούν να συνδεθούν (login) και να έχουν πρόσβαση στις λειτουργίες τους.

(Σε αυτή την άσκηση θα υλοποιήσετε μόνο όσες λειτουργίες αναφέρονται στο βήμα 5)

5 Υλοποίηση επιλεγμένων μεθόδων (λειτουργιών)

5.1 Για τους Πωλητές (Sellers) της εφαρμογής, να υλοποιήστε τις εξής μεθόδους:

5.1.1 Λειτουργία σύνδεσης (login) και αποσύνδεσης (logout).

5.1.2 Προβολή όλων των διαθέσιμων προγραμμάτων/πακέτων τηλεφωνίας.

5.1.3 Εισαγωγή νέου πελάτη.

5.1.4 Αντιστοίχηση πελάτη σε πρόγραμμα τηλεφωνίας.

5.2 Για την προβολή του αποτελέσματος κάθε μίας από τις παραπάνω ενέργειες, θα δημιουργείται μία δυναμική html σελίδα μέσω του servlet (ή συνδυασμός servlet και JSP). Δημιουργήστε επίσης τις απαραίτητες στατικές html σελίδες που απαιτούνται.

Χρήσιμες Πηγές

[1] Web UI data generator. https://www.mockaroo.com

[2] Python based data generator με αναλυτικές οδηγίες για install.

https://github.com/joke2k/faker?fbclid=IwAR1sDpaB2Y74bAPTY5AvQGdkt4ir8yy0nl6H-6P8rxS3MsfuPcnmQaGoutY

[3] Κατηγορίες για επιλογή.

https://faker.readthedocs.io/en/master/providers.html?fbclid=IwAR1Vftrjll1_-SUXGg3HNCt70L60o4Z6GKM4S1AAEIWWrP3q8WFnoCfIev4

Οδηγίες:

- Ισχύουν οι ίδιες ομάδες και οι ίδιες οδηγίες με την προηγούμενη εργασία.
- ☐ Το συνολικό παραδοτέο θα περιλαμβάνει σε ένα συμπιεσμένο αρχείο: (α) το project, (β) τη βάση δεδομένων (.sql ή .mwb αρχείο εάν χρησιμοποιείτε το Workbench) και (γ) την τεκμηρίωση αντίστοιχα

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Add Client Servlet

```java
@WebServlet("/add-client")
public class AddClientServlet extends HttpServlet {
    private static final long serialVersionUID = -2903102612786892472L;
    @Override

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        // Check if the SIGNEDIN cookie exists
        boolean signedIn = false;
        jakarta.servlet.http.Cookie[] cookies = request.getCookies();
        if (cookies != null) {
            for (jakarta.servlet.http.Cookie cookie : cookies) {
                if (cookie.getName().equals("SIGNEDIN")) {
                    signedIn = true;
                    break;
                }
            }
        }

        if (!signedIn) {
            // Redirect to Login page if not signed in
            response.sendRedirect(request.getContextPath() + "/login");
            return;
        }
        String address = "/pages/add-client.jsp";
        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Set encodings for requests and responses
        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");
        request.setCharacterEncoding("UTF-8");
```

```java
        // Get the parameters from the POST request
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String name = request.getParameter("name");
        String surname = request.getParameter("surname");
        String email = request.getParameter("email");
        String am = request.getParameter("am");
        String phoneNumberStr = request.getParameter("phoneNumber");
        int programId = Integer.parseInt(request.getParameter("program"));



        ProgramDao programDao = new ProgramDao();
        Program program = programDao.getProgramById(programId);
        program.displayProgramInfo();
        // Save the phone number using PhoneNumberDao
        PhoneNumberDao phoneNumberDao = new PhoneNumberDao();
        phoneNumberDao.addPhoneNumber(phoneNumberStr,programId);


        UserDao userDao = new UserDao();
        int user_id = userDao.addUser(username, password, name, surname, email, Role.CLIENT);


        ClientDao clientDao = new ClientDao();
        clientDao.addClient(user_id, am, phoneNumberStr);




        // Redirect to the clients page
        response.sendRedirect(request.getContextPath() + "/clients");
    }
}
```

Μέθοδος doPost:

- Θέτει την κωδικοποίηση των αιτημάτων και απαντήσεων σε UTF-8 για να υποστηρίζονται χαρακτήρες διεθνών γλωσσών.
- Λαμβάνει τις παραμέτρους από το POST αίτημα που περιέχει τα στοιχεία του νέου πελάτη.
- Χρησιμοποιεί τα ProgramDao, PhoneNumberDao, UserDao, και ClientDao για να αποθηκεύσει τις πληροφορίες του νέου πελάτη στη βάση δεδομένων.
- Ανακατευθύνει τον χρήστη στη σελίδα /clients αφού ολοκληρωθεί η διαδικασία.

Το servlet AddClientServlet παρέχει τη δυνατότητα προσθήκης νέου πελάτη. Όταν λαμβάνει ένα GET αίτημα, ελέγχει αν ο χρήστης είναι συνδεδεμένος και, αν ναι, δρομολογεί το αίτημα στη σελίδα add-client.jsp για να συμπληρώσει ο χρήστης τη φόρμα προσθήκης νέου πελάτη. Όταν λαμβάνει ένα POST αίτημα, λαμβάνει τα δεδομένα από τη φόρμα, τα αποθηκεύει στη βάση δεδομένων και ανακατευθύνει τον χρήστη στη σελίδα clients.

2 App Exception Handler

AppExceptionHandler.java

src > main > java > main > servlets > AppExceptionHandler.java

```java
package main.servlets;

import java.io.IOException;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/AppExceptionHandler")
public class AppExceptionHandler extends HttpServlet {
    private static final long serialVersionUID = -2938347283864271477L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String address = "/pages/404.jsp";
        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String address = "/pages/404.jsp";
        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}
```

- Το @WebServlet("/AppExceptionHandler") αναφέρει ότι αυτό το servlet θα απαντάει στα αιτήματα που έχουν ως URL το /AppExceptionHandler.

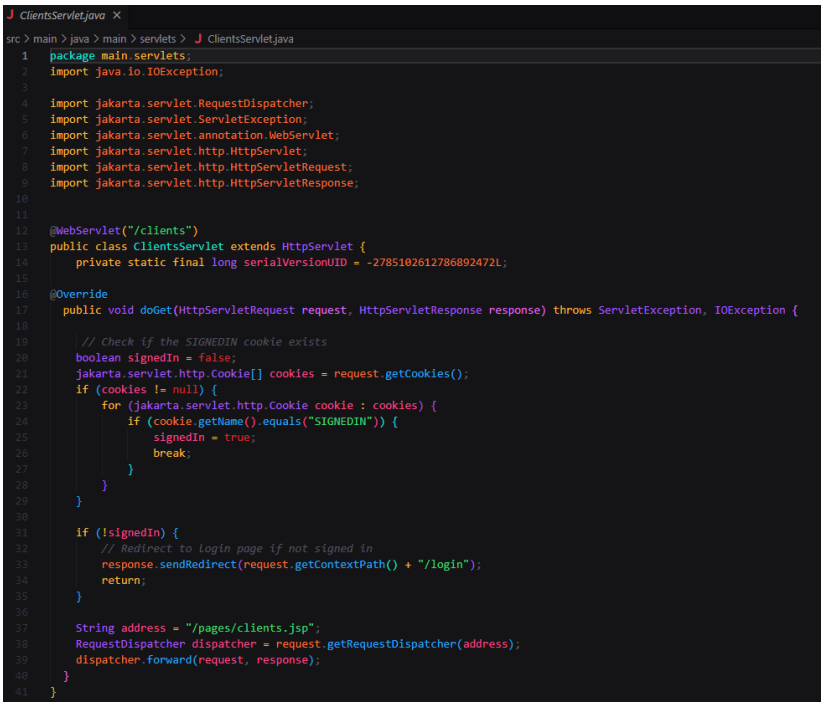
- Μεταβλητή serialVersionUID
  Αυτή η μεταβλητή χρησιμοποιείται για να εξασφαλιστεί η συμβατότητα μεταξύ διαφορετικών εκδόσεων της κλάσης κατά την σειριοποίηση.

- Μέθοδος doPost
  Αυτή η μέθοδος χειρίζεται τα POST αιτήματα. Καθορίζει ότι όλα τα POST αιτήματα θα δρομολογούνται στην σελίδα 404.jsp.

- Μέθοδος doGet
  Αυτή η μέθοδος χειρίζεται τα GET αιτήματα. Καθορίζει ότι όλα τα GET αιτήματα θα δρομολογούνται στην σελίδα 404.jsp.

  Ανεξάρτητα από το αν το αίτημα είναι GET ή POST, το servlet AppExceptionHandler ανακατευθύνει τον χρήστη στην σελίδα 404.jsp. Αυτό είναι χρήσιμο για την διαχείριση σφαλμάτων, όταν μια σελίδα δεν βρεθεί ή όταν υπάρχει κάποια εξαίρεση που πρέπει να διαχειριστεί. Η σελίδα 404.jsp πιθανόν να περιέχει ένα μήνυμα σφάλματος ή πληροφορίες για το τι πήγε στραβά.

3 Client Servlet



```java
package main.servlets;
import java.io.IOException;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


@WebServlet("/clients")
public class ClientsServlet extends HttpServlet {
    private static final long serialVersionUID = -2785102612786892472L;

@Override
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

     // Check if the SIGNEDIN cookie exists
    boolean signedIn = false;
    jakarta.servlet.http.Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (jakarta.servlet.http.Cookie cookie : cookies) {
            if (cookie.getName().equals("SIGNEDIN")) {
                signedIn = true;
                break;
            }
        }
    }

    if (!signedIn) {
        // Redirect to login page if not signed in
        response.sendRedirect(request.getContextPath() + "/login");
        return;
    }

    String address = "/pages/clients.jsp";
    RequestDispatcher dispatcher = request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

- Έλεγχος για το Cookie SIGNEDIN
  Ελέγχει αν το cookie με το όνομα SIGNEDIN υπάρχει στα cookies του αιτήματος. Αν υπάρχει, θέτει τη μεταβλητή signedIn σε true.

- Ανακατεύθυνση σε Σελίδα Σύνδεσης
  Αν ο χρήστης δεν είναι συνδεδεμένος (δηλαδή, το cookie SIGNEDIN δεν υπάρχει), το servlet ανακατευθύνει τον χρήστη στη σελίδα σύνδεσης (/login) και σταματάει την εκτέλεση της μεθόδου (return).

- Δρομολόγηση στο clients.jsp

  Αν ο χρήστης είναι συνδεδεμένος, το servlet δρομολογεί το αίτημα στη σελίδα clients.jsp χρησιμοποιώντας το RequestDispatcher.

Όταν το servlet λαμβάνει ένα GET αίτημα στη διαδρομή /clients:

- Ελέγχει αν ο χρήστης είναι συνδεδεμένος μέσω του cookie SIGNEDIN.
- Αν ο χρήστης δεν είναι συνδεδεμένος, τον ανακατευθύνει στη σελίδα σύνδεσης (/login).
- Αν ο χρήστης είναι συνδεδεμένος, δρομολογεί το αίτημα στη σελίδα clients.jsp.

Αυτός ο μηχανισμός χρησιμοποιείται για την προστασία σελίδων που απαιτούν σύνδεση, ώστε να έχουν πρόσβαση σε αυτές μόνο οι εξουσιοδοτημένοι χρήστες.

4 Contact Servlet

ContactServlet.java

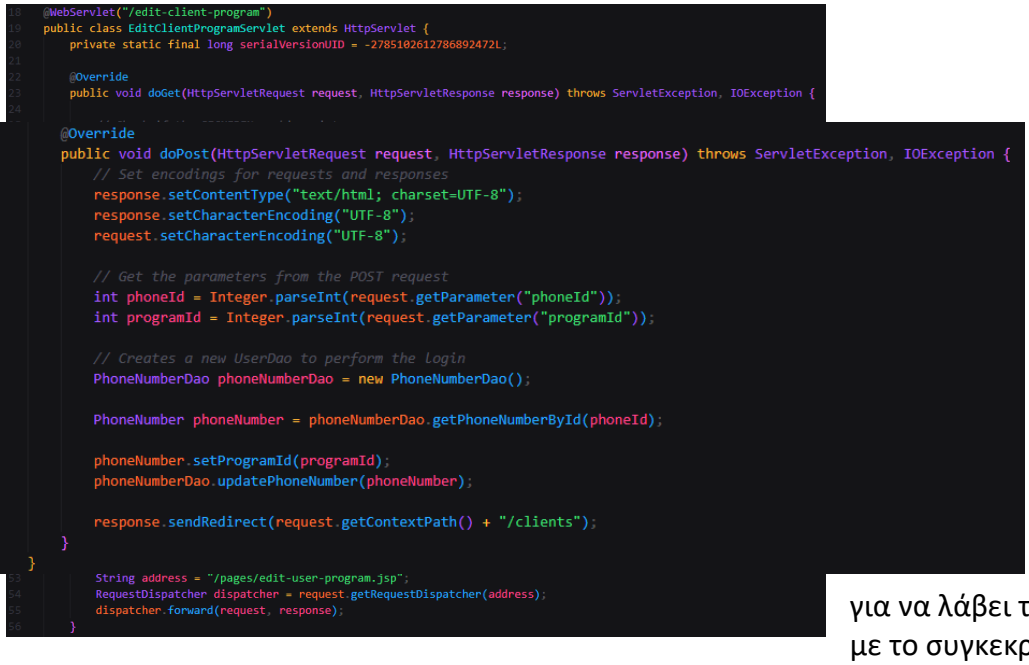
src > main > java > main > servlets > ContactServlet.java

```java
package main.servlets;
import java.io.IOException;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


@WebServlet("/contact")
public class ContactServlet extends HttpServlet {
  private static final long serialVersionUID = 1076048530718946589L;

@Override
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

    String address = "/contact.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

Όταν το servlet λαμβάνει ένα GET αίτημα στη διαδρομή /contact, δρομολογεί το αίτημα στη σελίδα contact.jsp. Δηλαδή όταν κάποιος χρήστης επισκέπτεται τη διεύθυνση /contact, θα βλέπει το περιεχόμενο της σελίδας contact.jsp.

Ουσιαστικά, το servlet αυτό απλά προωθεί το αίτημα στη συγκεκριμένη σελίδα, λειτουργώντας ως ένας τρόπος να οργανωθεί η διαχείριση των σελίδων και των URL στο σύστημα.

5 Edit Client Program Servlet



```java
@WebServlet("/edit-client-program")
public class EditClientProgramServlet extends HttpServlet {
    private static final long serialVersionUID = -2785102612786892472L;

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```



```java
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Set encodings for requests and responses
        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");
        request.setCharacterEncoding("UTF-8");

        // Get the parameters from the POST request
        int phoneId = Integer.parseInt(request.getParameter("phoneId"));
        int programId = Integer.parseInt(request.getParameter("programId"));

        // Creates a new UserDao to perform the login
        PhoneNumberDao phoneNumberDao = new PhoneNumberDao();

        PhoneNumber phoneNumber = phoneNumberDao.getPhoneNumberById(phoneId);

        phoneNumber.setProgramId(programId);
        phoneNumberDao.updatePhoneNumber(phoneNumber);

        response.sendRedirect(request.getContextPath() + "/clients");
    }
}
```



```java
        String address = "/pages/edit-user-program.jsp";
        RequestDispatcher dispatcher = request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
```

Μέθοδος doGet

Ελέγχει αν υπάρχει το cookie SIGNEDIN. Αν δεν υπάρχει, ανακατευθύνει τον χρήστη στη σελίδα σύνδεσης (/login). Αν υπάρχει, λαμβάνει το clientId από το αίτημα, χρησιμοποιεί το ClientDao για να λάβει τις λεπτομέρειες του πελάτη με το συγκεκριμένο clientId, και τις θέτει στο αίτημα. Τέλος, δρομολογεί το αίτημα στη σελίδα edit-user-program.jsp.

Μέθοδος doPost

- Θέτει την κωδικοποίηση των αιτημάτων και απαντήσεων σε UTF-8 για να υποστηρίζονται χαρακτήρες διεθνών γλωσσών.
- Λαμβάνει τις παραμέτρους phoneId και programId από το POST αίτημα.
- Χρησιμοποιεί το PhoneNumberDao για να λάβει το αντικείμενο PhoneNumber με το συγκεκριμένο phoneId, αλλάζει το πρόγραμμα αυτού του αριθμού (θέτει το programId), και ενημερώνει τη βάση δεδομένων με την αλλαγή αυτή.
- Ανακατευθύνει τον χρήστη στη σελίδα clients.

Το servlet EditClientProgramServlet παρέχει τη δυνατότητα επεξεργασίας του προγράμματος ενός πελάτη. Όταν λαμβάνει ένα GET αίτημα, ελέγχει αν ο χρήστης είναι συνδεδεμένος και, αν ναι, δρομολογεί το αίτημα στη σελίδα edit-user-program.jsp για να επεξεργαστεί ο χρήστης τα στοιχεία του πελάτη. Όταν λαμβάνει ένα POST αίτημα, λαμβάνει τα δεδομένα από τη φόρμα, ενημερώνει τη βάση δεδομένων με τις αλλαγές και ανακατευθύνει τον χρήστη στη σελίδα clients.

6 Home Servlet

```java
@WebServlet("/home")
public class HomeServlet extends HttpServlet {
    private static final long serialVersionUID = -2785102612786892472L;

@Override
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

     // Check if the SIGNEDIN cookie exists
    boolean signedIn = false;
    jakarta.servlet.http.Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (jakarta.servlet.http.Cookie cookie : cookies) {
            if (cookie.getName().equals("SIGNEDIN")) {
                signedIn = true;
                break;
            }
        }
    }

    if (!signedIn) {
        // Redirect to login page if not signed in
        response.sendRedirect(request.getContextPath() + "/login");
        return;
    }

    String address = "/pages/home.jsp";
    RequestDispatcher dispatcher =
    request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

Το servlet HomeServlet παρέχει τη δυνατότητα εμφάνισης της αρχικής σελίδας (home page) για τον χρήστη. Όταν λαμβάνει ένα GET αίτημα, ελέγχει αν ο χρήστης είναι συνδεδεμένος μέσω του cookie SIGNEDIN. Αν ο χρήστης δεν είναι συνδεδεμένος, τον ανακατευθύνει στη σελίδα σύνδεσης (/login). Αν ο χρήστης είναι συνδεδεμένος, δρομολογεί το αίτημα στη σελίδα home.jsp.

7 Login Servlet

```java
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1076048530718946589L;

// Handle the GET Requests
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Returns the login page from the pages
        RequestDispatcher dispatcher = request.getRequestDispatcher("/pages/login.jsp");
        dispatcher.forward(request, response);
    }
```

```java
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Set encodings for requests and responses
        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");
        request.setCharacterEncoding("UTF-8");

        // Get the parameters from the POST request
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Creates a new UserDao to perform the login
        UserDao login = new UserDao();

        // Checks if there is a user and stores its id on the variable else it defaults
        // to 0
        int userId = login.findUser(username, password);

        // If authentication succeeded
        if (userId != 0) {
            // Set cookie with called SIGNEDIN with userId
            Cookie cookie = new Cookie("SIGNEDIN", String.valueOf(userId));

            // Set its duration to 1 day
            cookie.setMaxAge(24 * 60 * 60);

            // Adds the cookie to the response
            response.addCookie(cookie);

            // Redirect to /home
            response.sendRedirect(request.getContextPath() + "/home");
            // If authentication failed
        } else {
            // Handle login failure (e.g., show error message, redirect back to login page)
            request.setAttribute("loginError", "Incorrect Username or Password!");

            // Forward the request back to the login page
            RequestDispatcher dispatcher = request.getRequestDispatcher("/pages/login.jsp");
            dispatcher.forward(request, response);
        }
    }

}
```

Μέθοδος `doPost`

- Θέτει την κωδικοποίηση των αιτημάτων και απαντήσεων σε UTF-8 για να υποστηρίζονται χαρακτήρες διεθνών γλωσσών.
- Λαμβάνει το username και το password από το POST αίτημα.
- Δημιουργεί ένα νέο αντικείμενο UserDao για να πραγματοποιήσει την αυθεντικοποίηση.
- Χρησιμοποιεί τη μέθοδο findUser για να ελέγξει αν υπάρχει χρήστης με αυτά τα στοιχεία και αν ναι, επιστρέφει το userId.

- Αν η αυθεντικοποίηση επιτύχει (δηλαδή το userId δεν είναι 0), δημιουργεί ένα cookie με όνομα SIGNEDIN και τιμή το userId, θέτει τη διάρκεια ζωής του cookie σε 1 ημέρα, προσθέτει το cookie στην απάντηση και ανακατευθύνει τον χρήστη στη σελίδα /home.
- Αν η αυθεντικοποίηση αποτύχει, θέτει ένα μήνυμα λάθους στο αίτημα ("Incorrect Username or Password!") και δρομολογεί το αίτημα ξανά στη σελίδα σύνδεσης (/pages/login.jsp).

Το servlet LoginServlet παρέχει τη δυνατότητα στον χρήστη να συνδεθεί στην εφαρμογή. Όταν λαμβάνει ένα GET αίτημα, εμφανίζει τη σελίδα σύνδεσης. Όταν λαμβάνει ένα POST αίτημα, επεξεργάζεται τα στοιχεία σύνδεσης του χρήστη, ελέγχει την αυθεντικότητα τους μέσω της κλάσης UserDao, και αν τα στοιχεία είναι έγκυρα, δημιουργεί ένα cookie που υποδηλώνει ότι ο χρήστης είναι συνδεδεμένος και ανακατευθύνει τον χρήστη στην αρχική σελίδα. Αν τα στοιχεία δεν είναι έγκυρα, εμφανίζει μήνυμα λάθους και δρομολογεί το αίτημα ξανά στη σελίδα σύνδεσης.

8 Program Servlet

```java
@WebServlet("/program")
public class ProgramServlet extends HttpServlet {
    private static final long serialVersionUID = -2785102612786892472L;

@Override
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    // Check if the SIGNEDIN cookie exists
    boolean signedIn = false;
    jakarta.servlet.http.Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (jakarta.servlet.http.Cookie cookie : cookies) {
            if (cookie.getName().equals("SIGNEDIN")) {
                signedIn = true;
                break;
            }
        }
    }

    if (!signedIn) {
        // Redirect to login page if not signed in
        response.sendRedirect(request.getContextPath() + "/login");
        return;
    }

    String address = "/pages/program.jsp";
    RequestDispatcher dispatcher = request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

Το servlet ProgramServlet ελέγχει αν ο χρήστης είναι συνδεδεμένος μέσω ενός cookie με όνομα SIGNEDIN. Εάν το cookie δεν υπάρχει, ο χρήστης ανακατευθύνεται στη σελίδα σύνδεσης. Εάν το cookie υπάρχει, το αίτημα δρομολογείται στη σελίδα προγραμμάτων (/pages/program.jsp).

9 Client Dao

```java
public class ClientDao {
    Connection connection;
    static String err = "n/a";

    public ClientDao() {
        connection = Database.getConnection();
    }

    // Adds the Program
    public int addClient(int user_id,String am, String phonenumber) {
        try {
            PreparedStatement statement = connection.prepareStatement("INSERT INTO program (user_id, am, phonenumber) values (?, ?, ?);");
            statement.setInt(1, user_id);
            statement.setString(2, am);
            statement.setString(3, phonenumber);
            statement.executeUpdate();
            statement.close();
        } catch (Exception sqle) {
            err = sqle.toString();
            sqle.printStackTrace();
            return 1;
        }
        return 0;
    }
```

```java
    public List<Client> getAllClients() {
        List<Client> clients = new ArrayList<Client>();
        try {
            Statement statement = connection.createStatement();
            ResultSet result_set = statement.executeQuery(
                    "SELECT u.id, u.username, u.password, u.name, u.surname, u.email, u.role, c.AM, c.phoneNumber, pn.id as phone_id, pn.status as phone_status, p.id AS program_id\r\n"
                            + "FROM \"user\" u\r\n" + "JOIN client c ON u.id = c.user_id\r\n"
                            + "JOIN phoneNumber pn ON c.phoneNumber = pn.number\r\n"
                            + "JOIN program p ON pn.program_id = p.id;");
            while (result_set.next()) {
                String roleString = result_set.getString("role");
                Role role = Role.valueOf(roleString.toUpperCase());

                String phoneNumberString = result_set.getString("phoneNumber");
                int phoneIdInt = result_set.getInt("phone_id");
                int programIdInt = result_set.getInt("program_id");
                String phoneStatus = result_set.getString("phone_status");
                PhoneNumber phoneNumber = new PhoneNumber(phoneIdInt, phoneNumberString, programIdInt, phoneStatus);
                Client client = new Client(result_set.getInt("id"), result_set.getString("username"),
                        result_set.getString("name"), result_set.getString("surname"), result_set.getString("email"),
                        role, result_set.getString("AM"), phoneNumber);
                clients.add(client);
            }
        } catch (Exception sqle) {
            sqle.printStackTrace();
        }
        return clients;
    }
```

- **Connection connection**: Χρησιμοποιείται για να συνδεθεί στη βάση δεδομένων.

```
    public Client getClientById(int clientId) {
        Client client = null;
        try {
            PreparedStatement preparedStatement = connection.prepareStatement(
                    "SELECT u.id, u.username, u.password, u.name, u.surname, u.email, u.role, c.AM, c.phoneNumber, pn.status as phone_status, pn.id as phone_id, p.id AS program_id\r\n"
                            + "FROM \"user\" u\r\n" + "JOIN client c ON u.id = c.user_id\r\n"
                            + "JOIN phoneNumber pn ON c.phoneNumber = pn.number\r\n"
                            + "JOIN program p ON pn.program_id = p.id\r\n" + "WHERE u.id = ?;");
            preparedStatement.setInt(1, clientId);
            ResultSet result_set = preparedStatement.executeQuery();

            if (result_set.next()) {
                String roleString = result_set.getString("role");
                Role role = Role.valueOf(roleString.toUpperCase());

                String phoneNumberString = result_set.getString("phoneNumber");
                int phoneIdInt = result_set.getInt("phone_id");
                int programIdInt = result_set.getInt("program_id");
                String phoneStatus = result_set.getString("phone_status");
                PhoneNumber phoneNumber = new PhoneNumber(phoneIdInt, phoneNumberString, programIdInt, phoneStatus);
                client = new Client(result_set.getInt("id"), result_set.getString("username"),
                        result_set.getString("name"), result_set.getString("surname"), result_set.getString("email"),
                        role, result_set.getString("AM"), phoneNumber);
            }
        } catch (Exception sqle) {
            sqle.printStackTrace();
        }
        return client;
    }
}
```

- **static String err**: Χρησιμοποιείται για την αποθήκευση μηνυμάτων σφάλματος.

Ο κατασκευαστής αρχικοποιεί τη σύνδεση με τη βάση δεδομένων μέσω της κλάσης Database.

1. addClient(int user_id, String am, String phonenumber)

- Προσθέτει έναν νέο πελάτη στη βάση δεδομένων.
- Επιστρέφει 0 αν η προσθήκη ήταν επιτυχής και 1 αν υπήρξε σφάλμα.

2. getAllClients()

- Επιστρέφει μια λίστα με όλους τους πελάτες από τη βάση δεδομένων.

3.getClientById(int clientId)

- Επιστρέφει έναν πελάτη με βάση το ID του από τη βάση δεδομένων.

Η κλάση ClientDao χρησιμοποιεί SQL ερωτήματα για την αλληλεπίδραση με τη βάση δεδομένων και μετατρέπει τα αποτελέσματα των ερωτημάτων σε αντικείμενα Java για εύκολη διαχείριση και επεξεργασία.

10 Phone Number Dao

```java
public class PhoneNumberDao {
    Connection connection;
    static String err = "n/a";

    public PhoneNumberDao() {
        connection = Database.getConnection();
    }

    // Adds the PhoneNumber
    public int addPhoneNumber(String number, int programId) {
        try {


            PreparedStatement statement = connection.prepareStatement("INSERT INTO phoneNumber (number, program_id, status) values (?, ?, ?);");
            statement.setString(1, number);
            statement.setInt(2, programId);
            statement.setString(3, "ACTIVE");
            statement.executeUpdate();
            statement.close();
        } catch (Exception sqle) {
            err = sqle.toString();
            sqle.printStackTrace();
            return 1;
        }
        return 0;
    }

    public int updatePhoneNumber(PhoneNumber phoneNumber) {
        try {
            PreparedStatement statement = connection.prepareStatement("UPDATE phoneNumber SET id = ?, number= ?, program_id = ?, status = ? WHERE id = ?;");
            statement.setInt(1, phoneNumber.getId());
            statement.setString(2, phoneNumber.getPhone());
            statement.setInt(3, phoneNumber.getProgramId());
            statement.setString(4, phoneNumber.getStatus());
            statement.setInt(5, phoneNumber.getId());
            statement.executeUpdate();
        } catch (Exception sqle) {
            sqle.printStackTrace();
            return 1;
        }
        return 0;
    }
```

```java
    public List<PhoneNumber> getAllPhoneNumbers(){
        List<PhoneNumber> phoneNumbers = new ArrayList<PhoneNumber>();
        try {
            Statement statement = connection.createStatement();
            ResultSet result_set = statement.executeQuery("SELECT * FROM phoneNumber;");
            while (result_set.next()) {
                PhoneNumber phoneNumber = new PhoneNumber(
                    result_set.getInt("id"),
                    result_set.getString("number"),
                    result_set.getInt("program_id"),
                    result_set.getString("status")
                );
                phoneNumbers.add(phoneNumber);
            }
        } catch (Exception sqle) {
            sqle.printStackTrace();
        }
        return phoneNumbers;
    }

    // Returns a Specific PhoneNumber from the Database which has the id given
    public PhoneNumber getPhoneNumberById(int phoneId) {
        try {
            PreparedStatement statement = connection.prepareStatement("SELECT * FROM PhoneNumber WHERE id = ?;");
            statement.setInt(1, phoneId);
            ResultSet result_set = statement.executeQuery();
            while (result_set.next()) {
                PhoneNumber pack = new PhoneNumber(
                        result_set.getInt("id"),
                        result_set.getString("number"),
                        result_set.getInt("program_id"),
                        result_set.getString("status")
                    );
                return pack;
            }
        } catch (Exception sqle) {
            sqle.printStackTrace();
        }
        return new PhoneNumber(0, "6935558090", 1, "INACTIVE");
    }

    public String giveErr() {
        return err;
    }
}
```

1. addPhoneNumber(String number, int programId)

- Προσθέτει έναν νέο τηλεφωνικό αριθμό στη βάση δεδομένων με συγκεκριμένο πρόγραμμα.
- Επιστρέφει 0 αν η προσθήκη ήταν επιτυχής και 1 αν υπήρξε σφάλμα.

2.updatePhoneNumber(PhoneNumber phoneNumber)

- Ενημερώνει τις πληροφορίες ενός τηλεφωνικού αριθμού στη βάση δεδομένων.
- Επιστρέφει 0 αν η ενημέρωση ήταν επιτυχής και 1 αν υπήρξε σφάλμα.

3.getAllPhoneNumbers()

- Επιστρέφει μια λίστα με όλους τους τηλεφωνικούς αριθμούς από τη βάση δεδομένων.

4,getPhoneNumberById(int phoneId)

- Επιστρέφει έναν συγκεκριμένο τηλεφωνικό αριθμό με βάση το ID του από τη βάση δεδομένων.
- Επιστρέφει έναν προκαθορισμένο τηλεφωνικό αριθμό αν δεν βρεθεί ο ζητούμενος.

5.giveErr()

- Επιστρέφει το μήνυμα σφάλματος.

-Προσθήκη Τηλεφωνικού Αριθμού: Προσθέτει έναν νέο τηλεφωνικό αριθμό στη βάση δεδομένων με χρήση της μεθόδου addPhoneNumber.

-Ενημέρωση Τηλεφωνικού Αριθμού: Ενημερώνει τις πληροφορίες ενός τηλεφωνικού αριθμού χρησιμοποιώντας τη μέθοδο updatePhoneNumber.

-Λήψη Όλων των Τηλεφωνικών Αριθμών: Επιστρέφει μια λίστα με όλους τους τηλεφωνικούς αριθμούς χρησιμοποιώντας τη μέθοδο getAllPhoneNumbers.

-Λήψη Τηλεφωνικού Αριθμού με Βάση το ID: Επιστρέφει έναν συγκεκριμένο τηλεφωνικό αριθμό με βάση το ID του χρησιμοποιώντας τη μέθοδο getPhoneNumberById.

-Αντιμετώπιση Σφαλμάτων: Αποθηκεύει και επιστρέφει μηνύματα σφάλματος με τη μέθοδο giveErr.

11 Program Dao

```java
public class ProgramDao {
    static String err = "n/a";

    public ProgramDao() {
        connection = Database.getConnection();
    }

    // Adds the Program
    public int addProgramme(Program newProgramme) {
        try {
            PreparedStatement statement = connection.prepareStatement("INSERT INTO program (name, description, costs, benefits, duration) values (?, ?, ?, ?, ?);");
            statement.setString(1, newProgramme.getName());
            statement.setString(2, newProgramme.getDescription());
            statement.setString(3, newProgramme.getCosts());
            statement.setString(4, newProgramme.getBenefits());
            statement.setInt(5, newProgramme.getDuration());
            statement.executeUpdate();
            statement.close();
        } catch (Exception sqle) {
            err = sqle.toString();
            sqle.printStackTrace();
            return 1;
        }
        return 0;
    }

    public int updateProgramme(Program pack) {
        try {
            PreparedStatement statement = connection.prepareStatement("UPDATE program SET name = ?, description= ?, costs= ?, benefits = ?, duration = ? WHERE id = ?;");
            statement.setInt(1, pack.getProgramID());
            statement.setString(2, pack.getName());
            statement.setString(3, pack.getDescription());
            statement.setString(4, pack.getCosts());
            statement.setInt(5, pack.getDuration());
            statement.executeUpdate();
        } catch (Exception sqle) {
            sqle.printStackTrace();
            return 1;
        }
        return 0;
    }
```

Η Κλάση Program Dao κάνει:

- **Προσθήκη Προγράμματος:** Προσθέτει ένα νέο πρόγραμμα στη βάση δεδομένων με χρήση της μεθόδου addProgramme.
- **Ενημέρωση Προγράμματος:** Ενημερώνει τις πληροφορίες ενός προγράμματος χρησιμοποιώντας τη μέθοδο updateProgramme.
- **Λήψη Όλων των Προγραμμάτων:** Επιστρέφει μια λίστα με όλα τα προγράμματα χρησιμοποιώντας τη μέθοδο getAllPrograms.
- **Λήψη Προγράμματος με Βάση το ID:** Επιστρέφει ένα συγκεκριμένο πρόγραμμα με βάση το ID του χρησιμοποιώντας τη μέθοδο getProgramById.

- **Αντιμετώπιση Σφαλμάτων:** Αποθηκεύει και επιστρέφει μηνύματα σφάλματος με τη μέθοδο giveErr.

```java
public class ProgramDao {
    public List<Program> getAllPrograms(){
        List<Program> packs = new ArrayList<Program>();
        try {
            Statement statement = connection.createStatement();
            ResultSet result_set = statement.executeQuery("SELECT * FROM program;");
            while (result_set.next()) {
                Program pack = new Program(
                    result_set.getInt("id"),
                    result_set.getString("name"),
                    result_set.getString("description"),
                    result_set.getString("benefits"),
                    result_set.getString("costs"),
                    result_set.getInt("duration")
                );
                packs.add(pack);
            }
        } catch (Exception sqle) {
            sqle.printStackTrace();
        }
        return packs;
    }

    // Returns a Specific Program from the Database which has the id given
    public Program getProgramById(int packid) {
        try {
            PreparedStatement statement = connection.prepareStatement("SELECT * FROM program WHERE id = ?;");
            statement.setInt(1, packid);
            ResultSet result_set = statement.executeQuery();
            while (result_set.next()) {
                Program pack = new Program(
                        result_set.getInt("id"),
                        result_set.getString("name"),
                        result_set.getString("description"),
                        result_set.getString("benefits"),
                        result_set.getString("costs"),
                        result_set.getInt("duration")
                    );
                return pack;
            }
        } catch (Exception sqle) {
            sqle.printStackTrace();
        }
        return new Program(0, "Fake Data Plan", "This is a Fake Data Plan", "No Benefits Included", "Maximum Cost Applied", 60 );
    }
```

12 User Dao

```java
public class UserDao {
    Connection connection;
    static String err = "n/a";

    public UserDao() {
        connection = Database.getConnection();
    }


    public int addUser(String username, String password, String name, String surname, String email, Role role) {
        int userId = -1;  // Default value indicating failure
        try {
            // Prepare the SQL statement with RETURN_GENERATED_KEYS to retrieve the new ID
            String query = "INSERT INTO user (username, password, name, surname, email, role) values (?, ?, ?, ?, ?, ?);";
            PreparedStatement statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
            statement.setString(1, username);
            statement.setString(2, password);
            statement.setString(3, name);
            statement.setString(4, surname);
            statement.setString(5, email);
            statement.setString(6, role.toLowercase());

            // Execute the statement
            statement.executeUpdate();

            // Retrieve the generated keys
            ResultSet generatedKeys = statement.getGeneratedKeys();
            if (generatedKeys.next()) {
                userId = generatedKeys.getInt(1);  // Assuming the ID is the first column
            }

            // Close the resources
            generatedKeys.close();
            statement.close();
        } catch (Exception sqle) {
            err = sqle.toString();
            sqle.printStackTrace();
        }
        return userId;
    }
```

```java
public class UserDao {
    public int findUser(String username, String password) {
        try {
            // The prepared query
            String query = "SELECT * FROM \"user\" WHERE username = ? AND password = ?";

            // Prepare the Statement and set username and password
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setString(1, username);
            statement.setString(2, password);

            // Executes the result set
            ResultSet result_set = statement.executeQuery();
             if (result_set.next()) {
                    // Gets the id of the user
                    int userId = result_set.getInt("id");
                    System.out.println("User found with ID: " + userId);

                    // Return the user ID
```

```java
public User getUser(int id) {
    try {
        // The prepared query
        String query = "SELECT * FROM \"user\" WHERE id = ?";

        // Prepare the Statement and set username and password
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        // Executes the result set
        ResultSet result_set = statement.executeQuery();
         if (result_set.next()) {
            // Gets the user and returns it
            String roleString = result_set.getString("role");
            Role role = Role.valueOf(roleString.toUpperCase());
            User user_to_return = new User(result_set.getInt("id"), result_set.getString("username"),  result_set.getString("name"), result_set.getString("surname"), result_set.getStr
              return user_to_return;
         } else {
            // User not found
            return new User(0, "jdoe", "John", "Doe",  "johndoe@gmail.com", Role.CLIENT);
         }
    } catch (Exception sqle) {
        sqle.printStackTrace();
    }
    // Return Fake User if not found anything
    return new User(0, "jdoe", "John", "Doe",  "johndoe@gmail.com", Role.CLIENT);
}
}
```

Η Κλάση UserDao κάνει:

- **Προσθήκη Χρήστη:** Προσθέτει έναν νέο χρήστη στη βάση δεδομένων με χρήση της μεθόδου addUser.
- **Αναζήτηση Χρήστη:** Αναζητά έναν χρήστη με βάση το όνομα χρήστη και τον κωδικό πρόσβασης χρησιμοποιώντας τη μέθοδο findUser.
- **Λήψη Χρήστη με Βάση το ID:** Επιστρέφει τις πληροφορίες ενός χρήστη με βάση το ID του χρησιμοποιώντας τη μέθοδο getUser.
- **Αντιμετώπιση Σφαλμάτων:** Αποθηκεύει και επιστρέφει μηνύματα σφάλματος μέσω της στατικής μεταβλητής err.

13 User

```java
package mainpackage;

public class User {  10 usages  3 inheritors
    // username of User ( unique )
    private String username;  6 usages
    // name of User
    private String name;  3 usages
    // surname of User
    private String surname;  3 usages
    // email of User
    private String email;  3 usages
    // Role of User ( CLIENT, SELLER, ADMIN )
    private Role role;  3 usages
    // Total Users
    private static int userCounter = 0;  2 usages

    public User(String username, String name, String surname, String email, Role role) {  9 usages
        this.username = username;
        this.name = name;
        this.surname = surname;
        this.email = email;
        this.role = role;
        userCounter++;
    }

    public void register() { System.out.println("[+] User " + username + " has been registered."); }

    public void login() { System.out.println("[+] User " + username + " has logged in."); }

    public void logout() { System.out.println("[+] User " + username + " has logged out."); }

    // Gets the User Counter
    public static int getUserCounter() { return userCounter; }

    // Gets the Username of the User
    public String getUsername() { return username; }

    // Sets the Username of the User using a String
    public void setUsername(String newUsername) { this.username = newUsername; }
```

```java

    // Gets the name of the User
    public String getName() { return name; }

    // Sets the name of the User
    public void setName(String newName) { this.name = newName; }

    // Gets the surname of the User
    public String getSurname() { return surname; }

    // Sets the surname of the User
    public void setSurname(String newSurname) { this.surname = newSurname; }

    // Gets the email of the User
    public String getEmail() { return email; }

    // Sets the email of the User
    public void setEmail(String newEmail) { this.email = newEmail; }

    // Gets the role of the User
    public Role getRole() { return role; }

    // Sets the role of the User
    public void setRole(Role newRole) { this.role = newRole; }

    // The enum of Role to ensure only specific values
    public enum Role {  11 usages
        CLIENT,  2 usages
        SELLER,  1 usage
        ADMIN  1 usage
    }
}
```

Βασικά χαρακτηριστικά και μέθοδοι της κλάσης User:
Μεταβλητές:

username: Το όνομα χρήστη του χρήστη.
name: Το όνομα του χρήστη.
surname: Το επώνυμο του χρήστη.
email: Το email του χρήστη.
role: Η ιδιότητα του χρήστη (CLIENT, SELLER, ADMIN).
userCounter: Ένας αριθμός που μετράει το σύνολο των χρηστών που έχουν δημιουργηθεί.

Μέθοδοι:

register(): Εμφανίζει ένα μήνυμα ότι ο χρήστης έχει εγγραφεί.
login(): Εμφανίζει ένα μήνυμα ότι ο χρήστης έχει συνδεθεί.
logout(): Εμφανίζει ένα μήνυμα ότι ο χρήστης έχει αποσυνδεθεί.

Μεθόδοι πρόσβασης (getter/setter) για κάθε μεταβλητή ιδιότητας ώστε να μπορούν να διαβαστούν και να οριστούν οι τιμές τους.
Μια εσωτερική enum Role που ορίζει τους διαφορετικούς ρόλους που μπορεί να έχει ένας χρήστης.

13.1 Υποκλάση Client

```java
package mainpackage;

public class Client extends User {  9 usages
    private final int AM;  2 usages
    private PhoneNumber phoneNumber;  3 usages

    public Client(String username, String name, String surname, String email, Role role, int AM, PhoneNumber phoneNumber) {  3 usages
        super(username, name, surname, email, role);
        this.AM = AM;
        this.phoneNumber = phoneNumber;
    }

    public void showBill() {  1 usage
        System.out.println("[+] Showing bill for Client " + getUsername());
        // Implement logic to show bill
    }

    public void showHistory() {  1 usage
        System.out.println("[+] Showing history for Client " + getUsername());
        // Implement logic to show history
    }

    public void payBill() {  1 usage
        System.out.println("[+] Payment successful for Client " + getUsername());
        // Implement logic to pay bill
    }

    // Gets the Unique AM of the Client
    public int getAM() { return AM; }

    // Gets the phoneNumber of the Client
    public PhoneNumber getPhoneNumber() { return phoneNumber; }

    // Sets the phoneNumber of the Client
    public void setPhoneNumber(PhoneNumber newPhoneNumber) { this.phoneNumber = newPhoneNumber; }
}
```

Η υποκλάση Client επεκτείνει την κλάση User, προσθέτοντας ειδικές λειτουργίες και χαρακτηριστικά που είναι συγκεκριμένα για τους πελάτες στο σύστημά.

Βασικά στοιχεία της κλάσης Client:

- Μεταβλητές:

AM: Τον αριθμό μητρώου (AM) του πελάτη. Είναι μια σταθερή τιμή που δίνεται στον κάθε πελάτη και δεν αλλάζει.
phoneNumber: Ο αριθμός τηλεφώνου του πελάτη, αποθηκευμένος ως αντικείμενο της κλάσης “PhoneNumber”.

- Constructor:

Ο constructor δέχεται τα ίδια ορίσματα με τον constructor της κλάσης User, καθώς και δύο επιπλέον ορίσματα: τον AM και τον αριθμό τηλεφώνου. Καλεί τον constructor της γονικής κλάσης User για να αρχικοποιήσει τα κοινά στοιχεία.

- Μέθοδοι:

showBill(), showHistory(), payBill():
Αυτές οι μέθοδοι υλοποιούν τη λογική για να εμφανίσουν τον λογαριασμό, το ιστορικό και να πληρώσουν τον λογαριασμό του πελάτη αντίστοιχα.
Μέθοδοι πρόσβασης (getter/setter) για τις μεταβλητές ιδιότητες του πελάτη, όπως ο ΑΜ και ο αριθμός τηλεφώνου.

Με αυτήν την υποκλάση, μπορούμε να διαχειριστούμε ειδικές λειτουργίες που αφορούν τους πελάτες, όπως η εμφάνιση λογαριασμού, ιστορικού και πληρωμής λογαριασμού. Επίσης, ορίζει τον μοναδικό αριθμό ΑΜ για κάθε πελάτη και αποθηκεύει τον αριθμό τηλεφώνου του πελάτη.

13.2 Υποκλάση Seller

```java
package mainpackage;

public class Seller extends User {  6 usages
    private final int AM;  2 usages

    public Seller(String username, String name, String surname, String email, Role role, int AM) {  3 usages
        super(username, name, surname, email, role);
        this.AM = AM;
    }

    public void addNewClient(Client client) {  1 usage
        System.out.println("[+] New client " + client.getUsername() + " added by seller " + getUsername());
        // Implement logic to add new client
    }

    public void changeClientProgram(Client client) {  1 usage
        System.out.println("[+] Program changed for client " + client.getUsername() + " by seller " + getUsername());
        // Implement logic to change client program
    }

    public void issueClientBill(Client client) {  1 usage
        System.out.println("[+] Bill issued for client " + client.getUsername() + " by seller " + getUsername());
        // Implement logic to issue client bill
    }

    public int getAM() { return AM; }
}
```

Η υποκλάση Seller επίσης επεκτείνει την κλάση User και προσθέτει ειδικές λειτουργίες που σχετίζονται με τους πωλητές στο σύστημά.

- Μεταβλητές:

AM: Ο Αριθμός Μητρώου (AM) του πωλητή.

- Constructor:

Ο constructor δέχεται τα ίδια ορίσματα με τον constructor της κλάσης User, καθώς και μία επιπλέον παράμετρο για τον AM του πωλητή. Καλεί τον constructor της γονικής κλάσης User για να αρχικοποιήσει τα κοινά στοιχεία.

- Μέθοδοι (Methods):

addNewClient(Client client): Προσθέτει ένα νέο πελάτη στο σύστημα και εμφανίζει ένα μήνυμα με το όνομα του πελάτη και το όνομα χρήστη του πωλητή που πραγματοποίησε την προσθήκη.

changeClientProgram(Client client): Αλλάζει το πρόγραμμα ενός πελάτη και εμφανίζει ένα μήνυμα με το όνομα του πελάτη και το όνομα χρήστη του πωλητή που πραγματοποίησε την αλλαγή.

issueClientBill(Client client): Εκδίδει τον λογαριασμό ενός πελάτη και εμφανίζει ένα μήνυμα με το όνομα του πελάτη και το όνομα χρήστη του πωλητή που εξέδωσε τον λογαριασμό.

getAM(): Επιστρέφει τον AM του πωλητή. Αυτή η υποκλάση επιτρέπει στο σύστημά να διαχειρίζεται τις ειδικές λειτουργίες που σχετίζονται με τους πωλητές, όπως η προσθήκη νέων πελατών, η αλλαγή προγραμμάτων πελατών και η έκδοση λογαριασμών πελατών.

13.3 Υποκλάση Admin

```
package mainpackage;

public final class Admin extends User {  3 usages

    public Admin(String username, String name, String surname, String email, Role role) {  3 usages
        super(username, name, surname, email, role);
    }

    public void createClient(Client client) {  1 usage
        System.out.println("[+] Client " + client.getUsername() + " created by admin " + getUsername());
    }

    public void editClient(Client client) {  1 usage
        System.out.println("[+] Client " + client.getUsername() + " edited by admin " + getUsername());
    }

    public void deleteClient(Client client) {  1 usage
        System.out.println("[+] Client " +  client.getUsername() + " deleted by admin " + getUsername());
    }

    public void createSeller(Seller seller) {  1 usage
        System.out.println("[+] Seller " + seller.getUsername() + " created by admin " + getUsername());
    }

    public void editSeller(Seller seller) {  1 usage
        System.out.println("[+] Seller " + seller.getUsername() + " edited by admin " + getUsername());
    }

    public void deleteSeller(Seller seller) {  1 usage
        System.out.println("[+] Seller " + seller.getUsername() + " deleted by admin " + getUsername());
    }

    public void createProgram(Program program) { System.out.println("[+] Program created by admin " + getUsername()); }

    public void editProgram(Program program) { System.out.println("[+] Program edited by admin " + getUsername()); }

    public void deleteProgram(Program program) { System.out.println("[+] Program deleted by admin " + getUsername()); }
}
```

Η υποκλάση Admin επίσης επεκτείνει την κλάση User, αλλά παρέχει λειτουργίες που είναι διαθέσιμες μόνο για τους διαχειριστές του συστήματος.

- Constructor:

Ο constructor δέχεται τα ίδια ορίσματα με τον constructor της κλάσης User και καλεί τον constructor της γονικής κλάσης User για να αρχικοποιήσει τα κοινά στοιχεία.

- Μέθοδοι:

createClient(Client client): Δημιουργεί ένα νέο πελάτη στο σύστημα και εμφανίζει ένα μήνυμα με το όνομα χρήστη του πελάτη και το όνομα χρήστη του διαχειριστή που πραγματοποίησε τη δημιουργία.

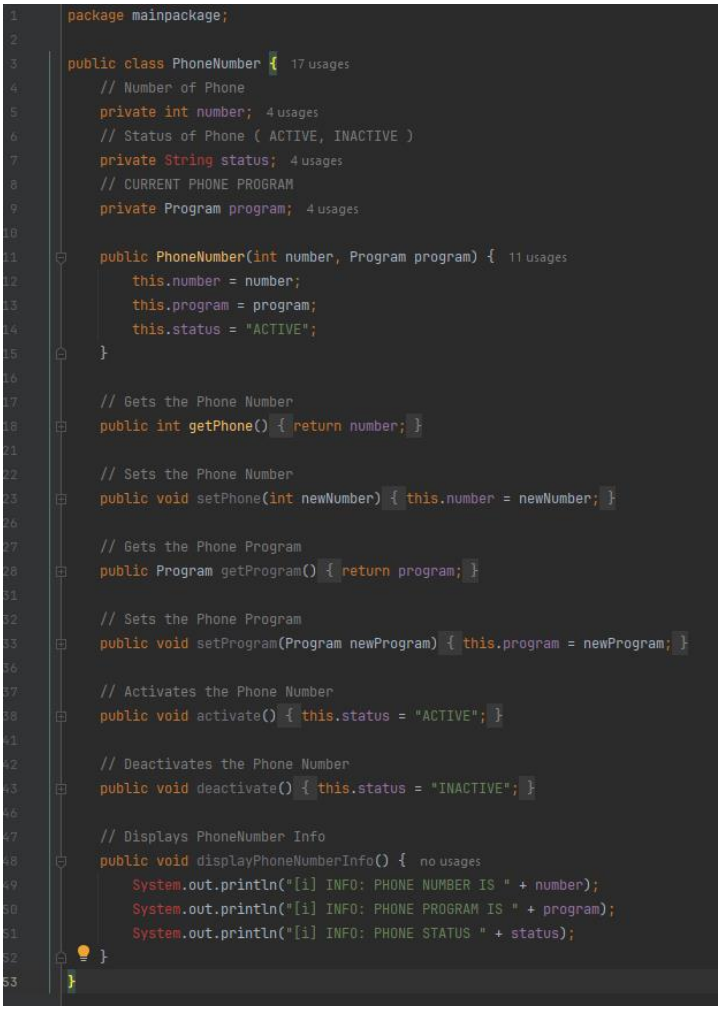
editClient(Client client): Επεξεργάζεται έναν υπάρχοντα πελάτη στο σύστημα και εμφανίζει ένα μήνυμα με το όνομα χρήστη του πελάτη και το όνομα χρήστη του διαχειριστή που πραγματοποίησε την επεξεργασία.

deleteClient(Client client): Διαγράφει έναν υπάρχοντα πελάτη από το σύστημα και εμφανίζει ένα μήνυμα με το όνομα χρήστη του πελάτη και το όνομα χρήστη του διαχειριστή που πραγματοποίησε τη διαγραφή.

Αντίστοιχες μέθοδοι για (δημιουργία, επεξεργασία και διαγραφή) πωλητών και προγραμμάτων.

Αυτή η υποκλάση παρέχει τις λειτουργίες διαχείρισης χρηστών και προγραμμάτων στο σύστημα, που είναι προνομιακές και μόνο για τους διαχειριστές.

14 PhoneNumber

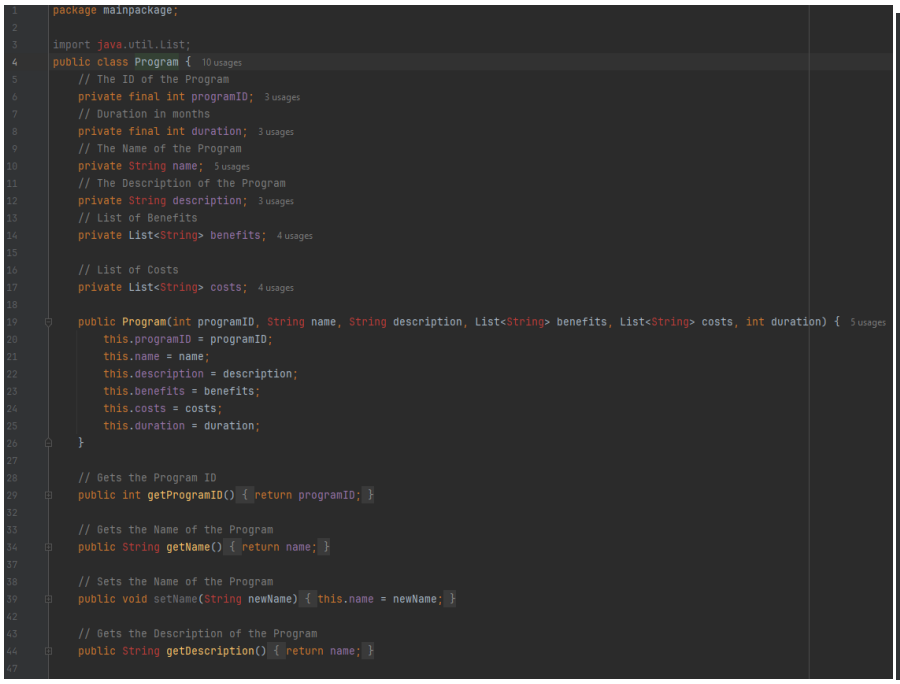


```java
package mainpackage;

public class PhoneNumber {  17 usages
    // Number of Phone
    private int number;  4 usages
    // Status of Phone ( ACTIVE, INACTIVE )
    private String status;  4 usages
    // CURRENT PHONE PROGRAM
    private Program program;  4 usages

    public PhoneNumber(int number, Program program) {  11 usages
        this.number = number;
        this.program = program;
        this.status = "ACTIVE";
    }

    // Gets the Phone Number
    public int getPhone() { return number; }

    // Sets the Phone Number
    public void setPhone(int newNumber) { this.number = newNumber; }

    // Gets the Phone Program
    public Program getProgram() { return program; }

    // Sets the Phone Program
    public void setProgram(Program newProgram) { this.program = newProgram; }

    // Activates the Phone Number
    public void activate() { this.status = "ACTIVE"; }

    // Deactivates the Phone Number
    public void deactivate() { this.status = "INACTIVE"; }

    // Displays PhoneNumber Info
    public void displayPhoneNumberInfo() {  no usages
        System.out.println("[i] INFO: PHONE NUMBER IS " + number);
        System.out.println("[i] INFO: PHONE PROGRAM IS " + program);
        System.out.println("[i] INFO: PHONE STATUS " + status);
    }
}
```

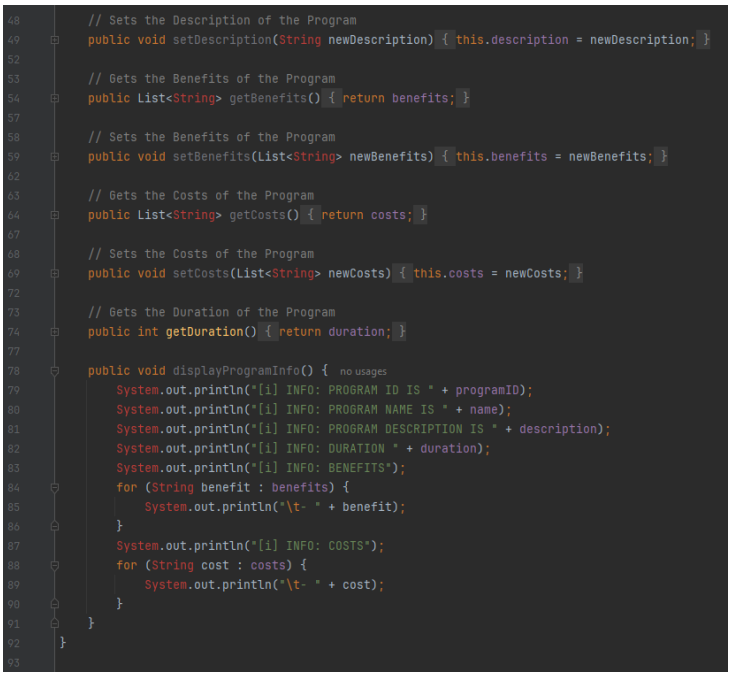
Η κλάση PhoneNumber παριστάνει έναν αριθμό τηλεφώνου στο σύστημά σας και περιλαμβάνει πληροφορίες σχετικά με τον αριθμό τηλεφώνου, την κατάστασή του (ενεργός ή ανενεργός) και το πρόγραμμα που σχετίζεται με αυτόν τον αριθμό.

- Μεταβλητές:
  number: Ο αριθμός τηλεφώνου.
  status: Η κατάσταση του τηλεφώνου, η οποία μπορεί να είναι "ACTIVE" ή "INACTIVE".
  program: Το πρόγραμμα που σχετίζεται με τον αριθμό τηλεφώνου.
- Constructor:
  Ο constructor δέχεται έναν αριθμό τηλεφώνου και ένα αντικείμενο προγράμματος και αρχικοποιεί τις μεταβλητές ιδιότητες. Κατά τη δημιουργία ενός αριθμού τηλεφώνου, θεωρείται ότι είναι ενεργός.
- Μέθοδοι:
  getPhone(), setPhone(int newNumber): Οι μέθοδοι getter και setter για τον αριθμό τηλεφώνου.
  getProgram(), setProgram(Program newProgram): Οι μέθοδοι getter και setter για το πρόγραμμα που σχετίζεται με τον αριθμό τηλεφώνου.
  activate(), deactivate(): Οι μέθοδοι για την ενεργοποίηση και απενεργοποίηση του τηλεφώνου αντίστοιχα.
  displayPhoneNumberInfo(): Εμφανίζει πληροφορίες σχετικά με τον αριθμό τηλεφώνου, το πρόγραμμα που σχετίζεται με αυτόν και την κατάστασή του.

15 Program



```java
package mainpackage;

import java.util.List;
public class Program {  10 usages
    // The ID of the Program
    private final int programID;  3 usages
    // Duration in months
    private final int duration;  3 usages
    // The Name of the Program
    private String name;  5 usages
    // The Description of the Program
    private String description;  3 usages
    // List of Benefits
    private List<String> benefits;  4 usages

    // List of Costs
    private List<String> costs;  4 usages

    public Program(int programID, String name, String description, List<String> benefits, List<String> costs, int duration) {  5 usages
        this.programID = programID;
        this.name = name;
        this.description = description;
        this.benefits = benefits;
        this.costs = costs;
        this.duration = duration;
    }

    // Gets the Program ID
    public int getProgramID() { return programID; }

    // Gets the Name of the Program
    public String getName() { return name; }

    // Sets the Name of the Program
    public void setName(String newName) { this.name = newName; }

    // Gets the Description of the Program
    public String getDescription() { return name; }
```



```java
    // Sets the Description of the Program
    public void setDescription(String newDescription) { this.description = newDescription; }

    // Gets the Benefits of the Program
    public List<String> getBenefits() { return benefits; }

    // Sets the Benefits of the Program
    public void setBenefits(List<String> newBenefits) { this.benefits = newBenefits; }

    // Gets the Costs of the Program
    public List<String> getCosts() { return costs; }

    // Sets the Costs of the Program
    public void setCosts(List<String> newCosts) { this.costs = newCosts; }

    // Gets the Duration of the Program
    public int getDuration() { return duration; }

    public void displayProgramInfo() {  no usages
        System.out.println("[i] INFO: PROGRAM ID IS " + programID);
        System.out.println("[i] INFO: PROGRAM NAME IS " + name);
        System.out.println("[i] INFO: PROGRAM DESCRIPTION IS " + description);
        System.out.println("[i] INFO: DURATION " + duration);
        System.out.println("[i] INFO: BENEFITS");
        for (String benefit : benefits) {
            System.out.println("\t- " + benefit);
        }
        System.out.println("[i] INFO: COSTS");
        for (String cost : costs) {
            System.out.println("\t- " + cost);
        }
    }
}
```

Η κλάση Program παριστάνει ένα πρόγραμμα στο σύστημα και περιλαμβάνει πληροφορίες όπως το ID του προγράμματος, τη διάρκειά του, το όνομά του, την περιγραφή του, τα οφέλη και το κόστος του.

- Μεταβλητές:

programID: Το ID του προγράμματος.
duration: Η διάρκεια του προγράμματος σε μήνες.
name: Το όνομα του προγράμματος.
description: Η περιγραφή του προγράμματος.
benefits: Μια λίστα με τα οφέλη του προγράμματος.
costs: Μια λίστα με τα κόστη του προγράμματος.

- Constructor:

Ο constructor δέχεται τα στοιχεία που χρειάζονται για τη δημιουργία ενός προγράμματος και αρχικοποιεί τις μεταβλητές ιδιότητες.

- Μέθοδοι:

Υπάρχουν μεθόδοι getter και setter για κάθε ιδιότητα του προγράμματος, ώστε να επιτρέπεται η πρόσβαση και η τροποποίησή τους.

displayProgramInfo(): Εμφανίζει πληροφορίες για το πρόγραμμα, όπως το ID, το όνομα, την περιγραφή, τη διάρκεια, τα οφέλη και τα κόστη του.

16 Call

```java
package mainpackage;

public class Call {  5 usages
    // Unique Call ID
    private final int callID;  3 usages
    // The PhoneNumber of the Caller
    private final PhoneNumber callerNumber;  3 usages
    // The PhoneNumber of the Receiver
    private final PhoneNumber receiverNumber;  3 usages
    // Duration of Call in seconds
    private int duration;  4 usages
    // The Call type between INCOMING, OUTGOING and MISSED
    private final CallType callType;  3 usages

    public Call(int callID, PhoneNumber callerNumber, PhoneNumber receiverNumber, int duration, CallType callType) {
        this.callID = callID;
        this.callerNumber = callerNumber;
        this.receiverNumber = receiverNumber;
        this.duration = duration;
        this.callType = callType;
    }

    // Gets the Call ID
    public int getCallID() { return callID; }

    // Gets the Caller Number
    public PhoneNumber getCallerNumber() { return callerNumber; }

    // Gets the Receiver Number
    public PhoneNumber getReceiverNumber() { return receiverNumber; }

    // Gets the duration of the Call
    public long getDuration() { return duration; }

    // Sets the duration of the Call
    public void setDuration(int newDuration) { this.duration = newDuration; }

    // Returns the Call type
    public CallType getCallType() { return callType; }

    // Displays call information
    public void displayCallInfo() {  no usages
        System.out.println("[i] INFO: CALL ID IS " + callID);
        System.out.println("[i] INFO: CALLER NUMBER IS " + callerNumber);
        System.out.println("[i] INFO: RECEIVER NUMBER IS " + receiverNumber);
        System.out.println("[i] INFO: DURATION " + duration);
        System.out.println("[i] INFO: CALL TYPE " + callType);
    }

    // Call Type Enumeration to ensure that it has specific values
    public enum CallType {  3 usages
        INCOMING,  no usages
        OUTGOING,  no usages
        MISSED  no usages
    }
}
```

Η κλάση Call αντιπροσωπεύει μία κλήση στο σύστημά και περιλαμβάνει πληροφορίες όπως το μοναδικό ID της κλήσης, τον αριθμό του καλούντα, τον αριθμό του δέκτη, τη διάρκεια της κλήσης και τον τύπο της κλήσης (εισερχόμενη, εξερχόμενη ή αναπάντητη).

- Μεταβλητές:

callID: Το μοναδικό ID της κλήσης.
callerNumber: Ο αριθμός του καλούντα.
receiverNumber: Ο αριθμός του δέκτη.
duration: Η διάρκεια της κλήσης σε δευτερόλεπτα.
callType: Ο τύπος της κλήσης (εισερχόμενη, εξερχόμενη ή αναπάντητη).

- Constructor:

Ο constructor δέχεται τα στοιχεία που χρειάζονται για τη δημιουργία μίας κλήσης και αρχικοποιεί τις μεταβλητές ιδιότητες.

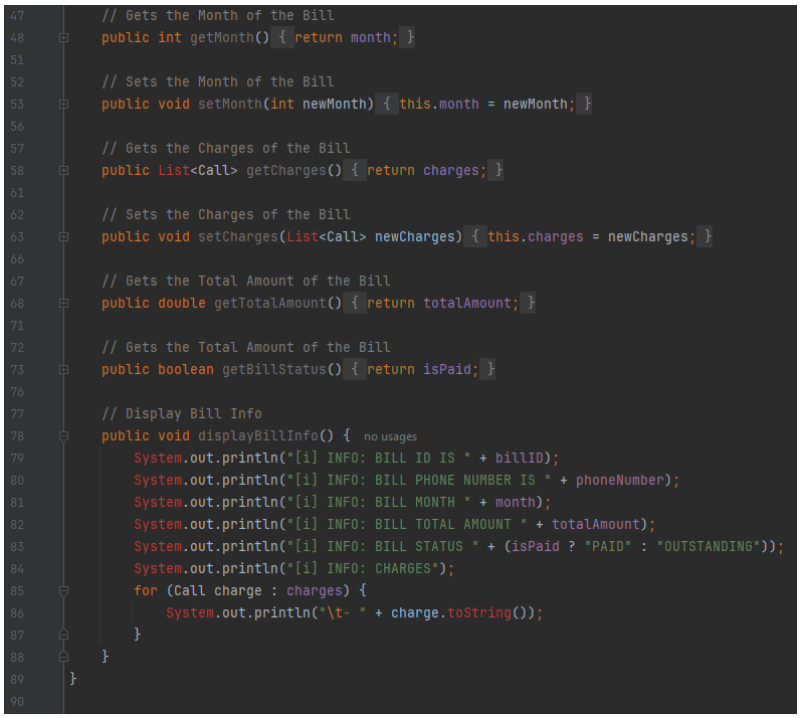
- Μέθοδοι:

Υπάρχουν μεθόδοι getter για τις ιδιότητες της κλήσης, ώστε να επιτρέπεται η πρόσβαση σε αυτές.

displayCallInfo(): Εμφανίζει πληροφορίες για την κλήση, όπως το ID της, τους αριθμούς του καλούντα και του δέκτη, τη διάρκεια της κλήσης και τον τύπο της κλήσης.

17 Bill

```java
package mainpackage;

import ...

public class Bill {  no usages
    // The Bill ID
    private final int billID;  3 usages
    // The PhoneNumber associated with the Bill
    private PhoneNumber phoneNumber;  4 usages
    // The Bill's Month
    private int month;  4 usages
    // The Bill's Charges
    private List<Call> charges;  4 usages
    // The Total Amount to pay the Bill
    private double totalAmount;  3 usages
    // Whether the Bill isPaid or not ( true or false )
    private boolean isPaid;  3 usages

    // Constructor
    public Bill(int billID, PhoneNumber phoneNumber, int month, List<Call> charges) {  2 usages
        this.billID = billID;
        this.phoneNumber = phoneNumber;
        this.month = month;
        this.charges = charges;
        // Initially, the bill is 0, then the cost is calculated
        this.totalAmount = 0.0;
        // Initially, the bill is not paid
        this.isPaid = false;
    }

    // Gets the ID of the Bill
    public int getBillID() { return billID; }

    // Gets the phoneNumber of the Bill
    public PhoneNumber getPhoneNumber() { return phoneNumber; }

    // Sets the phoneNumber of the Bill
    public void setPhoneNumber(PhoneNumber newPhoneNumber) { this.phoneNumber = newPhoneNumber; }
```



```java
    // Gets the Month of the Bill
    public int getMonth() { return month; }

    // Sets the Month of the Bill
    public void setMonth(int newMonth) { this.month = newMonth; }

    // Gets the Charges of the Bill
    public List<Call> getCharges() { return charges; }

    // Sets the Charges of the Bill
    public void setCharges(List<Call> newCharges) { this.charges = newCharges; }

    // Gets the Total Amount of the Bill
    public double getTotalAmount() { return totalAmount; }

    // Gets the Total Amount of the Bill
    public boolean getBillStatus() { return isPaid; }

    // Display Bill Info
    public void displayBillInfo() {  no usages
        System.out.println("[i] INFO: BILL ID IS " + billID);
        System.out.println("[i] INFO: BILL PHONE NUMBER IS " + phoneNumber);
        System.out.println("[i] INFO: BILL MONTH " + month);
        System.out.println("[i] INFO: BILL TOTAL AMOUNT " + totalAmount);
        System.out.println("[i] INFO: BILL STATUS " + (isPaid ? "PAID" : "OUTSTANDING"));
        System.out.println("[i] INFO: CHARGES");
        for (Call charge : charges) {
            System.out.println("\t- " + charge.toString());
        }
    }
}
```

Η κλάση Bill αντιπροσωπεύει μία τιμολόγηση για έναν συγκεκριμένο αριθμό τηλεφώνου σε έναν συγκεκριμένο μήνα. Περιέχει πληροφορίες όπως το μοναδικό ID του τιμολογίου, τον αριθμό τηλεφώνου που σχετίζεται με το τιμολόγιο, τον μήνα του τιμολογίου, τις χρεώσεις που περιλαμβάνονται σε αυτό και το συνολικό ποσό που πρέπει να πληρωθεί.

- Μεταβλητές:

billID: Το μοναδικό ID του τιμολογίου.
phoneNumber: Ο αριθμός τηλεφώνου που σχετίζεται με το τιμολόγιο.
month: Ο μήνας του τιμολογίου.
charges: Οι χρεώσεις που περιλαμβάνονται στο τιμολόγιο.
totalAmount: Το συνολικό ποσό του τιμολογίου.
isPaid: Εάν το τιμολόγιο έχει πληρωθεί ή όχι.

- Constructor:

Ο constructor δέχεται τα στοιχεία που χρειάζονται για τη δημιουργία ενός τιμολογίου και αρχικοποιεί τις μεταβλητές ιδιότητες.

- Μέθοδοι (Methods):

Υπάρχουν μεθόδοι getter για τις ιδιότητες του τιμολογίου, ώστε να επιτρέπεται η πρόσβαση σε αυτές.

displayBillInfo(): Εμφανίζει πληροφορίες για το τιμολόγιο, όπως το ID του, τον αριθμό τηλεφώνου, τον μήνα, το συνολικό ποσό και την κατάσταση πληρωμής.

18 Role

```java
package main.model;

// The enum of Role to ensure only specific values
public enum Role {
    CLIENT,
    SELLER,
    ADMIN;

     public String toLowercase() {
        return this.name().toLowerCase();
    }
}
```

Η παραπάνω κλάση Role είναι ένας enum τύπος στην Java που ορίζει τρεις συγκεκριμένους ρόλους για τους χρήστες ενός συστήματος: CLIENT, SELLER και ADMIN.
Ο enum τύπος εξασφαλίζει ότι οι ρόλοι που μπορούν να αντιστοιχιστούν στους χρήστες είναι μόνο αυτοί οι τρεις συγκεκριμένοι και όχι κάποιοι άλλοι.

Μέθοδος toLowercase

- Αυτή η μέθοδος επιστρέφει το όνομα της σταθεράς σε πεζά γράμματα.

19 Βιβλιογραφικές Πηγές

Για data insertion στις βασεις

https://www.mockaroo.com/