

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 postgres) και η μεταξέ 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.

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

- 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.
- Με τη βοήθεια του Μοντέλου Οντοτήτων-Σχέσεων, να δημιουργήσετε και να εικονογραφήσετε τη βάση στον database server. Μπορείτε να χρησιμοποιήσετε οποιοδήποτε βοηθητικό εργαλείο για την εξήγηση της βάσης από το μοντέλο (π.χ. mysql Workbench για mysql).
- 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 παικτα κλάσεων, τα οποία θα περιλαμβάνουν τις βασικές κλάσεις που έχουν υλοποιήσει στην προηγούμενη άσκηση.

3.3 Δημιουργήστε ένα ή περισσότερα γίντα κλάσεων, τα οποία 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 υλοποιήσετε τις εξής μεθόδους:

- Λειτουργία σύνδεσης (login) και αποσύνδεσης (logout).
- Προβολή όλων των διαθέσιμων προγραμμάτων/παικτων τηλεφωνίας.
- Εισαγωγή νέου πελάτη.
- Αντιστοίχιση 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=IwAR1sDpaB2Y74bAPTY5AaQGdktdir8ys0n6I6L6P28rxS3MsfuPcnmOnGomY>

[3] Κατηγορίες για επλοή.

<https://faker.readthedocs.io/en/master/providers.html?fbclid=IwAR1Vtrjll-SUXG7e3HNC70L60u476GKM4S1AAEiWw7P3q8WFnoCfey4>

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 .pwb αρχείο εάν χρησιμοποιείτε το Workbench) και (γ) την τεμαχίωση αντίστοιχα

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1	Add Client Servlet.....	Error! Bookmark not defined.
2	Add Exception Handler.....	5
3	Clients Servlet.....	6
4	Contact Servlet.....	7
5	Edit Client Program Servlet	8
6	Home Servlet.....	9
7	Login Servlet.....	10
8	Program Servlet.....	11
9	Client Dao	12
10	Phone Number Dao.....	14
11	Program Dao	16
12	User Dao.....	17
13	User	19
	13.1 Υποκλάση Client.....	20
	13.2 Υποκλάση Seller	22
	13.3 Υποκλάση Admin	23
14	PhoneNumber	2Error! Bookmark not defined.
15	Program.....	25
16	Call.....	26
17	Bill.....	27
18	Role.....	28
19	Βιβλιογραφικές Πηγές.....	29

1 Add Client Servlet

```
@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");
    }
}
```

```
// 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

```
J AppExceptionHandler.java X
src > main > java > main > servlets > J AppExceptionHandler.java
1  package main.servlets;
2
3  import java.io.IOException;
4  import jakarta.servlet.RequestDispatcher;
5  import jakarta.servlet.ServletException;
6  import jakarta.servlet.annotation.WebServlet;
7  import jakarta.servlet.http.HttpServlet;
8  import jakarta.servlet.http.HttpServletRequest;
9  import jakarta.servlet.http.HttpServletResponse;
10
11 @WebServlet("/AppExceptionHandler")
12 public class AppExceptionHandler extends HttpServlet {
13     private static final long serialVersionUID = -2938347283864271477L;
14
15     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16         String address = "/pages/404.jsp";
17         RequestDispatcher dispatcher = request.getRequestDispatcher(address);
18         dispatcher.forward(request, response);
19     }
20
21     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
22
23         String address = "/pages/404.jsp";
24         RequestDispatcher dispatcher = request.getRequestDispatcher(address);
25         dispatcher.forward(request, response);
26     }
27 }
```

- 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

```
src > main > java > main > servlets > ClientServlet.java
1 package main.servlets;
2 import java.io.IOException;
3
4 import jakarta.servlet.RequestDispatcher;
5 import jakarta.servlet.ServletException;
6 import jakarta.servlet.annotation.WebServlet;
7 import jakarta.servlet.http.HttpServlet;
8 import jakarta.servlet.http.HttpServletRequest;
9 import jakarta.servlet.http.HttpServletResponse;
10
11
12 @WebServlet("/clients")
13 public class ClientServlet extends HttpServlet {
14     private static final long serialVersionUID = -2785102612786892472L;
15
16     @Override
17     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
18
19         // Check if the SIGNEDIN cookie exists
20         boolean signedIn = false;
21         jakarta.servlet.http.Cookie[] cookies = request.getCookies();
22         if (cookies != null) {
23             for (jakarta.servlet.http.Cookie cookie : cookies) {
24                 if (cookie.getName().equals("SIGNEDIN")) {
25                     signedIn = true;
26                     break;
27                 }
28             }
29         }
30
31         if (!signedIn) {
32             // Redirect to login page if not signed in
33             response.sendRedirect(request.getContextPath() + "/login");
34             return;
35         }
36
37         String address = "/pages/clients.jsp";
38         RequestDispatcher dispatcher = request.getRequestDispatcher(address);
39         dispatcher.forward(request, response);
40     }
41 }
```

- 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

```
J ContactServlet.java X
src > main > java > main > servlets > J ContactServlet.java
1 package main.servlets;
2 import java.io.IOException;
3
4 import jakarta.servlet.RequestDispatcher;
5 import jakarta.servlet.ServletException;
6 import jakarta.servlet.annotation.WebServlet;
7 import jakarta.servlet.http.HttpServlet;
8 import jakarta.servlet.http.HttpServletRequest;
9 import jakarta.servlet.http.HttpServletResponse;
10
11
12 @WebServlet("/contact")
13 public class ContactServlet extends HttpServlet {
14     private static final long serialVersionUID = 1076048530718946589L;
15
16     @Override
17     public void doGet(HttpServletRequest request,
18                       HttpServletResponse response)
19         throws ServletException, IOException {
20
21         String address = "/contact.jsp";
22         RequestDispatcher dispatcher =
23             request.getRequestDispatcher(address);
24         dispatcher.forward(request, response);
25     }
26 }
```

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

```
18 @WebServlet("/edit-client-program")
19 public class EditClientProgramServlet extends HttpServlet {
20     private static final long serialVersionUID = -2785182612786892472L;
21
22     @Override
23     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2
```



```

12 @WebServlet("/home")
13 public class HomeServlet extends HttpServlet {
14     private static final long serialVersionUID = -2785102612786892472L;
15
16     @Override
17     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
18
19         // Check if the SIGNEDIN cookie exists
20         boolean signedIn = false;
21         jakarta.servlet.http.Cookie[] cookies = request.getCookies();
22         if (cookies != null) {
23             for (jakarta.servlet.http.Cookie cookie : cookies) {
24                 if (cookie.getName().equals("SIGNEDIN")) {
25                     signedIn = true;
26                     break;
27                 }
28             }
29         }
30
31         if (!signedIn) {
32             // Redirect to login page if not signed in
33             response.sendRedirect(request.getContextPath() + "/login");
34             return;
35         }
36
37         String address = "/pages/home.jsp";
38         RequestDispatcher dispatcher =
39             request.getRequestDispatcher(address);
40         dispatcher.forward(request, response);
41     }
42 }

```

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

```
16 @WebServlet("/login")
17 public class LoginServlet extends HttpServlet {
18     private static final long serialVersionUID = 1076048530718946589L;
19
20     // Handle the GET Requests
21     @Override
22     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23         // Returns the Login page from the pages
24         RequestDispatcher dispatcher = request.getRequestDispatcher("/pages/login.jsp");
25         dispatcher.forward(request, response);
26     }
27
28     @Override
29     public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
30         // Set encodings for requests and responses
31         response.setContentType("text/html; charset=UTF-8");
32         response.setCharacterEncoding("UTF-8");
33         request.setCharacterEncoding("UTF-8");
34
35         // Get the parameters from the POST request
36         String username = request.getParameter("username");
37         String password = request.getParameter("password");
38
39         // Creates a new UserDao to perform the Login
40         UserDao login = new UserDao();
41
42         // Checks if there is a user and stores its id on the variable else it defaults
43         // to 0
44         int userId = login.findUser(username, password);
45
46         // If authentication succeeded
47         if (userId != 0) {
48             // Set cookie with called SIGNEDIN with userId
49             Cookie cookie = new Cookie("SIGNEDIN", String.valueOf(userId));
50
51             // Set its duration to 1 day
52             cookie.setMaxAge(24 * 60 * 60);
53
54             // Adds the cookie to the response
55             response.addCookie(cookie);
56
57             // Redirect to /home
58             response.sendRedirect(request.getContextPath() + "/home");
59             // If authentication failed
60         } else {
61             // Handle Login failure (e.g., show error message, redirect back to login page)
62             request.setAttribute("loginError", "Incorrect Username or Password!");
63
64             // Forward the request back to the login page
65             RequestDispatcher dispatcher = request.getRequestDispatcher("/pages/login.jsp");
66             dispatcher.forward(request, response);
67         }
68     }
69 }
70 }
```

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

```
12 @WebServlet("/program")
13 public class ProgramServlet extends HttpServlet {
14     private static final long serialVersionUID = -2785102612786892472L;
15
16     @Override
17     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
18
19         // Check if the SIGNEDIN cookie exists
20         boolean signedIn = false;
21         jakarta.servlet.http.Cookie[] cookies = request.getCookies();
22         if (cookies != null) {
23             for (jakarta.servlet.http.Cookie cookie : cookies) {
24                 if (cookie.getName().equals("SIGNEDIN")) {
25                     signedIn = true;
26                     break;
27                 }
28             }
29         }
30
31         if (!signedIn) {
32             // Redirect to login page if not signed in
33             response.sendRedirect(request.getContextPath() + "/login");
34             return;
35         }
36
37         String address = "/pages/program.jsp";
38         RequestDispatcher dispatcher = request.getRequestDispatcher(address);
39         dispatcher.forward(request, response);
40     }
41 }
```

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

```
16 public class ClientDao {
17     Connection connection;
18     static String err = "n/a";
19
20     public ClientDao() {
21         connection = Database.getConnection();
22     }
23
24     // Adds the Program
25     public int addClient(int user_id,String am, String phonenumber) {
26         try {
27             PreparedStatement statement = connection.prepareStatement("INSERT INTO program (user_id, am, phonenumber) values (?, ?, ?);");
28             statement.setInt(1, user_id);
29             statement.setString(2, am);
30             statement.setString(3, phonenumber);
31             statement.executeUpdate();
32             statement.close();
33         } catch (Exception sqle) {
34             err = sqle.toString();
35             sqle.printStackTrace();
36             return 1;
37         }
38         return 0;
39     }
40 }
```

```
57 public List<Client> getAllClients() {
58     List<Client> clients = new ArrayList<Client>();
59     try {
60         Statement statement = connection.createStatement();
61         ResultSet result_set = statement.executeQuery(
62             "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"
63             + "FROM \\'user\\' u\r\n" + "JOIN client c ON u.id = c.user_id\r\n"
64             + "JOIN phoneNumber pn ON c.phoneNumber = pn.number\r\n"
65             + "JOIN program p ON pn.program_id = p.id;");
66         while (result_set.next()) {
67             String roleString = result_set.getString("role");
68             Role role = Role.valueOf(roleString.toUpperCase());
69
70             String phoneNumberString = result_set.getString("phoneNumber");
71             int phoneIdInt = result_set.getInt("phone_id");
72             int programIdInt = result_set.getInt("program_id");
73             String phoneStatus = result_set.getString("phone_status");
74             PhoneNumber phoneNumber = new PhoneNumber(phoneIdInt, phoneNumberString, programIdInt, phoneStatus);
75             Client client = new Client(result_set.getInt("id"), result_set.getString("username"),
76                 result_set.getString("name"), result_set.getString("surname"), result_set.getString("email"),
77                 role, result_set.getString("AM"), phoneNumber);
78             clients.add(client);
79         }
80     } catch (Exception sqle) {
81         sqle.printStackTrace();
82     }
83     return clients;
84 }
```

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

```

86 public Client getClientById(int clientId) {
87     Client client = null;
88     try {
89         PreparedStatement preparedStatement = connection.prepareStatement(
90             "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"
91             + "FROM \user\ u\r\n" + "JOIN client c ON u.id = c.user_id\r\n"
92             + "JOIN phoneNumber pn ON c.phoneNumber = pn.number\r\n"
93             + "JOIN program p ON pn.program_id = p.id\r\n" + "WHERE u.id = ?;");
94         preparedStatement.setInt(1, clientId);
95         ResultSet result_set = preparedStatement.executeQuery();
96
97         if (result_set.next()) {
98             String roleString = result_set.getString("role");
99             Role role = Role.valueOf(roleString.toUpperCase());
100
101             String phoneNumberString = result_set.getString("phoneNumber");
102             int phoneIdInt = result_set.getInt("phone_id");
103             int programIdInt = result_set.getInt("program_id");
104             String phoneStatus = result_set.getString("phone_status");
105             PhoneNumber phoneNumber = new PhoneNumber(phoneIdInt, phoneNumberString, programIdInt, phoneStatus);
106             client = new Client(result_set.getInt("id"), result_set.getString("username"),
107                 result_set.getString("name"), result_set.getString("surname"), result_set.getString("email"),
108                 role, result_set.getString("AM"), phoneNumber);
109         }
110     } catch (Exception sqle) {
111         sqle.printStackTrace();
112     }
113     return client;
114 }
115 }

```

- **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

```
1 public class PhoneNumberDao {
2     Connection connection;
3     static String err = "n/a";
4
5     public PhoneNumberDao() {
6         connection = Database.getConnection();
7     }
8
9     // Adds the PhoneNumber
10    public int addPhoneNumber(String number, int programId) {
11        try {
12
13            PreparedStatement statement = connection.prepareStatement("INSERT INTO phoneNumber (number, program_id, status) values (?, ?, ?);");
14            statement.setString(1, number);
15            statement.setInt(2, programId);
16            statement.setString(3, "ACTIVE");
17            statement.executeUpdate();
18            statement.close();
19        } catch (Exception sqle) {
20            err = sqle.toString();
21            sqle.printStackTrace();
22            return 1;
23        }
24        return 0;
25    }
26
27    public int updatePhoneNumber(PhoneNumber phoneNumber) {
28        try {
29            PreparedStatement statement = connection.prepareStatement("UPDATE phoneNumber SET id = ?, number= ?, program_id = ?, status = ? WHERE id = ?;");
30            statement.setInt(1, phoneNumber.getId());
31            statement.setString(2, phoneNumber.getNumber());
32            statement.setInt(3, phoneNumber.getProgramId());
33            statement.setString(4, phoneNumber.getStatus());
34            statement.setInt(5, phoneNumber.getId());
35            statement.executeUpdate();
36        } catch (Exception sqle) {
37            sqle.printStackTrace();
38            return 1;
39        }
40        return 0;
41    }
42
43    public List<PhoneNumber> getAllPhoneNumbers(){
44        List<PhoneNumber> phoneNumbers = new ArrayList<PhoneNumber>();
45        try {
46            Statement statement = connection.createStatement();
47            ResultSet result_set = statement.executeQuery("SELECT * FROM phoneNumber;");
48            while (result_set.next()) {
49                PhoneNumber phoneNumber = new PhoneNumber(
50                    result_set.getInt("id"),
51                    result_set.getString("number"),
52                    result_set.getInt("program_id"),
53                    result_set.getString("status")
54                );
55                phoneNumbers.add(phoneNumber);
56            }
57        } catch (Exception sqle) {
58            sqle.printStackTrace();
59        }
60        return phoneNumbers;
61    }
62
63    // Returns a Specific PhoneNumber from the Database which has the id given
64    public PhoneNumber getPhoneNumberById(int phoneId) {
65        try {
66            PreparedStatement statement = connection.prepareStatement("SELECT * FROM PhoneNumber WHERE id = ?;");
67            statement.setInt(1, phoneId);
68            ResultSet result_set = statement.executeQuery();
69            while (result_set.next()) {
70                PhoneNumber pack = new PhoneNumber(
71                    result_set.getInt("id"),
72                    result_set.getString("number"),
73                    result_set.getInt("program_id"),
74                    result_set.getString("status")
75                );
76                return pack;
77            }
78        } catch (Exception sqle) {
79            sqle.printStackTrace();
80        }
81        return new PhoneNumber(0, "6935558090", 1, "INACTIVE");
82    }
83
84    public String giveErr() {
85        return err;
86    }
87 }
```

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

```
10 public class ProgramDao {
11     static String err = "n/a";
12
13     public ProgramDao() {
14         connection = Database.getConnection();
15     }
16
17     // Adds the Program
18     public int addProgram(Program newProgram) {
19         try {
20             PreparedStatement statement = connection.prepareStatement("INSERT INTO program (name, description, costs, benefits, duration) values (?, ?, ?, ?, ?);");
21             statement.setString(1, newProgram.getName());
22             statement.setString(2, newProgram.getDescription());
23             statement.setString(3, newProgram.getCosts());
24             statement.setString(4, newProgram.getBenefits());
25             statement.setInt(5, newProgram.getDuration());
26             statement.executeUpdate();
27             statement.close();
28         } catch (Exception sqle) {
29             err = sqle.toString();
30             sqle.printStackTrace();
31             return 1;
32         }
33         return 0;
34     }
35
36     public int updateProgram(Program pack) {
37         try {
38             PreparedStatement statement = connection.prepareStatement("UPDATE program SET name = ?, description= ?, costs= ?, benefits = ?, duration = ? WHERE id = ?;");
39             statement.setInt(1, pack.getProgramID());
40             statement.setString(2, pack.getName());
41             statement.setString(3, pack.getDescription());
42             statement.setString(4, pack.getCosts());
43             statement.setInt(5, pack.getDuration());
44             statement.executeUpdate();
45         } catch (Exception sqle) {
46             sqle.printStackTrace();
47             return 1;
48         }
49         return 0;
50     }
51 }
```

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.

```
56 public class ProgramDao {
57     public List<Program> getAllPrograms(){
58         List<Program> packs = new ArrayList<Program>();
59         try {
60             Statement statement = connection.createStatement();
61             ResultSet result_set = statement.executeQuery("SELECT * FROM program;");
62             while (result_set.next()) {
63                 Program pack = new Program(
64                     result_set.getInt("id"),
65                     result_set.getString("name"),
66                     result_set.getString("description"),
67                     result_set.getString("benefits"),
68                     result_set.getString("costs"),
69                     result_set.getInt("duration")
70                 );
71                 packs.add(pack);
72             }
73         } catch (Exception sqle) {
74             sqle.printStackTrace();
75         }
76         return packs;
77     }
78
79     // Returns a Specific Program from the Database which has the id given
80     public Program getProgramById(int packid) {
81         try {
82             PreparedStatement statement = connection.prepareStatement("SELECT * FROM program WHERE id = ?;");
83             statement.setInt(1, packid);
84             ResultSet result_set = statement.executeQuery();
85             while (result_set.next()) {
86                 Program pack = new Program(
87                     result_set.getInt("id"),
88                     result_set.getString("name"),
89                     result_set.getString("description"),
90                     result_set.getString("benefits"),
91                     result_set.getString("costs"),
92                     result_set.getInt("duration")
93                 );
94                 return pack;
95             }
96         } catch (Exception sqle) {
97             sqle.printStackTrace();
98         }
99         return new Program(0, "Fake Data Plan", "This is a Fake Data Plan", "No Benefits Included", "Maximum Cost Applied", 60 );
100     }
101 }
102
103 public class UserDao {
104     Connection connection;
105     static String err = "n/a";
106
107     public UserDao() {
108         connection = Database.getConnection();
109     }
110
111     public int addUser(String username, String password, String name, String surname, String email, Role role) {
112         int userId = -1; // Default value indicating failure
113         try {
114             // Prepare the SQL statement with RETURN_GENERATED_KEYS to retrieve the new ID
115             String query = "INSERT INTO user (username, password, name, surname, email, role) values (?, ?, ?, ?, ?, ?);";
116             PreparedStatement statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
117             statement.setString(1, username);
118             statement.setString(2, password);
119             statement.setString(3, name);
120             statement.setString(4, surname);
121             statement.setString(5, email);
122             statement.setString(6, role.toLowerCase());
123
124             // Execute the statement
125             statement.executeUpdate();
126
127             // Retrieve the generated keys
128             ResultSet generatedKeys = statement.getGeneratedKeys();
129             if (generatedKeys.next()) {
130                 userId = generatedKeys.getInt(1); // Assuming the ID is the first column
131             }
132
133             // Close the resources
134             generatedKeys.close();
135             statement.close();
136         } catch (Exception sqle) {
137             err = sqle.toString();
138             sqle.printStackTrace();
139         }
140         return userId;
141     }
142 }
```

12 User Dao

```

13 public class UserDao {
55     public int findUser(String username, String password) {
56         try {
57             // The prepared query
58             String query = "SELECT * FROM \"user\" WHERE username = ? AND password = ?";
59
60             // Prepare the Statement and set username and password
61             PreparedStatement statement = connection.prepareStatement(query);
62             statement.setString(1, username);
63             statement.setString(2, password);
64
65             // Executes the result set
66             ResultSet result_set = statement.executeQuery();
67             if (result_set.next()) {
68                 // Gets the id of the user
69                 int userId = result_set.getInt("id");
70                 System.out.println("User found with ID: " + userId);
71             }
72             // Return the user ID
73         }
74     }
75
76     public User getUser(int id) {
77         try {
78             // The prepared query
79             String query = "SELECT * FROM \"user\" WHERE id = ?";
80
81             // Prepare the Statement and set username and password
82             PreparedStatement statement = connection.prepareStatement(query);
83             statement.setInt(1, id);
84
85             // Executes the result set
86             ResultSet result_set = statement.executeQuery();
87             if (result_set.next()) {
88                 // Gets the user and returns it
89                 String roleString = result_set.getString("role");
90                 Role role = Role.valueOf(roleString.toUpperCase());
91                 User user_to_return = new User(result_set.getInt("id"), result_set.getString("username"), result_set.getString("name"), result_set.getString("surname"), result_set.getString("email"), role);
92                 return user_to_return;
93             } else {
94                 // User not found
95                 return new User(0, "jdoe", "John", "Doe", "johndoe@gmail.com", Role.CLIENT);
96             }
97         } catch (Exception sqle) {
98             sqle.printStackTrace();
99         }
100         // Return Fake User if not found anything
101         return new User(0, "jdoe", "John", "Doe", "johndoe@gmail.com", Role.CLIENT);
102     }
103 }

```

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

```
1 package mainpackage;
2
3 public class User { 10 usages 3 inheritors
4     // username of User ( unique )
5     private String username; 6 usages
6     // name of User
7     private String name; 3 usages
8     // surname of User
9     private String surname; 3 usages
10    // email of User
11    private String email; 3 usages
12    // Role of User ( CLIENT, SELLER, ADMIN )
13    private Role role; 3 usages
14    // Total Users
15    private static int userCounter = 0; 2 usages
16
17    public User(String username, String name, String surname, String email, Role role) { 9 usages
18        this.username = username;
19        this.name = name;
20        this.surname = surname;
21        this.email = email;
22        this.role = role;
23        userCounter++;
24    }
25
26    public void register() { System.out.println("[+] User " + username + " has been registered."); }
27
28    public void login() { System.out.println("[+] User " + username + " has logged in."); }
29
30    public void logout() { System.out.println("[+] User " + username + " has logged out."); }
31
32    // Gets the User Counter
33    public static int getUserCounter() { return userCounter; }
34
35    // Gets the Username of the User
36    public String getUsername() { return username; }
37
38    // Sets the Username of the User using a String
39    public void setUsername(String newUsername) { this.username = newUsername; }
```

```
52
53    // Gets the name of the User
54    public String getName() { return name; }
55
56    // Sets the name of the User
57    public void setName(String newName) { this.name = newName; }
58
59    // Gets the surname of the User
60    public String getSurname() { return surname; }
61
62    // Sets the surname of the User
63    public void setSurname(String newSurname) { this.surname = newSurname; }
64
65    // Gets the email of the User
66    public String getEmail() { return email; }
67
68    // Sets the email of the User
69    public void setEmail(String newEmail) { this.email = newEmail; }
70
71    // Gets the role of the User
72    public Role getRole() { return role; }
73
74    // Sets the role of the User
75    public void setRole(Role newRole) { this.role = newRole; }
76
77    // The enum of Role to ensure only specific values
78    public enum Role { 11 usages
79        CLIENT, 2 usages
80        SELLER, 1 usage
81        ADMIN 1 usage
82    }
83
84 }
```

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

```
1 package mainpackage;
2
3 public class Client extends User { 9 usages
4     private final int AM; 2 usages
5     private PhoneNumber phoneNumber; 3 usages
6
7     public Client(String username, String name, String surname, String email, Role role, int AM, PhoneNumber phoneNumber) { 3 usages
8         super(username, name, surname, email, role);
9         this.AM = AM;
10        this.phoneNumber = phoneNumber;
11    }
12
13    public void showBill() { 1 usage
14        System.out.println("[+] Showing bill for Client " + getUsername());
15        // Implement logic to show bill
16    }
17
18    public void showHistory() { 1 usage
19        System.out.println("[+] Showing history for Client " + getUsername());
20        // Implement logic to show history
21    }
22
23    public void payBill() { 1 usage
24        System.out.println("[+] Payment successful for Client " + getUsername());
25        // Implement logic to pay bill
26    }
27
28    // Gets the Unique AM of the Client
29    public int getAM() { return AM; }
30
31
32    // Gets the phoneNumber of the Client
33    public PhoneNumber getPhoneNumber() { return phoneNumber; }
34
35
36    // Sets the phoneNumber of the Client
37    public void setPhoneNumber(PhoneNumber newPhoneNumber) { this.phoneNumber = newPhoneNumber; }
38
39
40
41
42 }
43
```

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

```
1 package mainpackage;
2
3 public class Seller extends User { 6 usages
4     private final int AM; 2 usages
5
6     public Seller(String username, String name, String surname, String email, Role role, int AM) { 3 usages
7         super(username, name, surname, email, role);
8         this.AM = AM;
9     }
10
11 @ public void addNewClient(Client client) { 1 usage
12     System.out.println("[+] New client " + client.getUsername() + " added by seller " + getUsername());
13     // Implement logic to add new client
14 }
15
16 @ public void changeClientProgram(Client client) { 1 usage
17     System.out.println("[+] Program changed for client " + client.getUsername() + " by seller " + getUsername());
18     // Implement logic to change client program
19 }
20
21 @ public void issueClientBill(Client client) { 1 usage
22     System.out.println("[+] Bill issued for client " + client.getUsername() + " by seller " + getUsername());
23     // Implement logic to issue client bill
24 }
25
26 public int getAM() { return AM; }
27
28 }
29
30 }
```

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

```
1 package mainpackage;
2
3 public final class Admin extends User { 3 usages
4
5     public Admin(String username, String name, String surname, String email, Role role) { 3 usages
6         super(username, name, surname, email, role);
7     }
8
9     @
10    public void createClient(Client client) { 1 usage
11        System.out.println("[+] Client " + client.getUsername() + " created by admin " + getUsername());
12    }
13
14    @
15    public void editClient(Client client) { 1 usage
16        System.out.println("[+] Client " + client.getUsername() + " edited by admin " + getUsername());
17    }
18
19    @
20    public void deleteClient(Client client) { 1 usage
21        System.out.println("[+] Client " + client.getUsername() + " deleted by admin " + getUsername());
22    }
23
24    @
25    public void createSeller(Seller seller) { 1 usage
26        System.out.println("[+] Seller " + seller.getUsername() + " created by admin " + getUsername());
27    }
28
29    @
30    public void editSeller(Seller seller) { 1 usage
31        System.out.println("[+] Seller " + seller.getUsername() + " edited by admin " + getUsername());
32    }
33
34    @
35    public void deleteSeller(Seller seller) { 1 usage
36        System.out.println("[+] Seller " + seller.getUsername() + " deleted by admin " + getUsername());
37    }
38
39    public void createProgram(Program program) { System.out.println("[+] Program created by admin " + getUsername()); }
40
41    public void editProgram(Program program) { System.out.println("[+] Program edited by admin " + getUsername()); }
42
43    public void deleteProgram(Program program) { System.out.println("[+] Program deleted by admin " + getUsername()); }
44
45 }
```

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 προνομακές και μόνο για τους διαχειριστές.

14 PhoneNumber

```
1 package mainpackage;
2
3 public class PhoneNumber { 17 usages
4     // Number of Phone
5     private int number; 4 usages
6     // Status of Phone ( ACTIVE, INACTIVE )
7     private String status; 4 usages
8     // CURRENT PHONE PROGRAM
9     private Program program; 4 usages
10
11     public PhoneNumber(int number, Program program) { 11 usages
12         this.number = number;
13         this.program = program;
14         this.status = "ACTIVE";
15     }
16
17     // Gets the Phone Number
18     public int getPhone() { return number; }
19
20
21
22     // Sets the Phone Number
23     public void setPhone(int newNumber) { this.number = newNumber; }
24
25
26
27     // Gets the Phone Program
28     public Program getProgram() { return program; }
29
30
31
32     // Sets the Phone Program
33     public void setProgram(Program newProgram) { this.program = newProgram; }
34
35
36
37     // Activates the Phone Number
38     public void activate() { this.status = "ACTIVE"; }
39
40
41
42     // Deactivates the Phone Number
43     public void deactivate() { this.status = "INACTIVE"; }
44
45
46
47     // Displays PhoneNumber Info
48     public void displayPhoneNumberInfo() { no usages
49         System.out.println("[i] INFO: PHONE NUMBER IS " + number);
50         System.out.println("[i] INFO: PHONE PROGRAM IS " + program);
51         System.out.println("[i] INFO: PHONE STATUS " + status);
52     }
53 }
```

Η κλάση PhoneNumber παριστάνει έναν αριθμό τηλεφώνου στο σύστημά σας και περιλαμβάνει πληροφορίες σχετικά με τον αριθμό τηλεφώνου, την κατάσταση 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

```
1 package mainpackage;
2
3 import java.util.List;
4 public class Program { 10 usages
5     // The ID of the Program
6     private final int programID; 3 usages
7     // Duration in months
8     private final int duration; 3 usages
9     // The Name of the Program
10    private String name; 5 usages
11    // The Description of the Program
12    private String description; 3 usages
13    // List of Benefits
14    private List<String> benefits; 4 usages
15
16    // List of Costs
17    private List<String> costs; 4 usages
18
19    public Program(int programID, String name, String description, List<String> benefits, List<String> costs, int duration) { 5 usages
20        this.programID = programID;
21        this.name = name;
22        this.description = description;
23        this.benefits = benefits;
24        this.costs = costs;
25        this.duration = duration;
26    }
27
28    // Gets the Program ID
29    public int getProgramID() { return programID; }
30
31    // Gets the Name of the Program
32    public String getName() { return name; }
33
34    // Sets the Name of the Program
35    public void setName(String newName) { this.name = newName; }
36
37    // Gets the Description of the Program
38    public String getDescription() { return name; }
39
40    // Sets the Description of the Program
41    public void setDescription(String newDescription) { this.description = newDescription; }
42
43    // Gets the Benefits of the Program
44    public List<String> getBenefits() { return benefits; }
45
46    // Sets the Benefits of the Program
47    public void setBenefits(List<String> newBenefits) { this.benefits = newBenefits; }
48
49    // Gets the Costs of the Program
50    public List<String> getCosts() { return costs; }
51
52    // Sets the Costs of the Program
53    public void setCosts(List<String> newCosts) { this.costs = newCosts; }
54
55    // Gets the Duration of the Program
56    public int getDuration() { return duration; }
57
58    public void displayProgramInfo() { no usages
59        System.out.println("[i] INFO: PROGRAM ID IS " + programID);
60        System.out.println("[i] INFO: PROGRAM NAME IS " + name);
61        System.out.println("[i] INFO: PROGRAM DESCRIPTION IS " + description);
62        System.out.println("[i] INFO: DURATION " + duration);
63        System.out.println("[i] INFO: BENEFITS");
64        for (String benefit : benefits) {
65            System.out.println("\t- " + benefit);
66        }
67        System.out.println("[i] INFO: COSTS");
68        for (String cost : costs) {
69            System.out.println("\t- " + cost);
70        }
71    }
72 }
73 }
```

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 μέθοδοι 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

```
1 package mainpackage;
2
3 public class Call { 5 usages
4     // Unique Call ID
5     private final int callID; 3 usages
6     // The PhoneNumber of the Caller
7     private final PhoneNumber callerNumber; 3 usages
8     // The PhoneNumber of the Receiver
9     private final PhoneNumber receiverNumber; 3 usages
10    // Duration of Call in seconds
11    private int duration; 4 usages
12    // The Call type between INCOMING, OUTGOING and MISSED
13    private final CallType callType; 3 usages
14
15    public Call(int callID, PhoneNumber callerNumber, PhoneNumber receiverNumber, int duration, CallType callType) {
16        this.callID = callID;
17        this.callerNumber = callerNumber;
18        this.receiverNumber = receiverNumber;
19        this.duration = duration;
20        this.callType = callType;
21    }
22
23    // Gets the Call ID
24    public int getCallID() { return callID; }
25
26    // Gets the Caller Number
27    public PhoneNumber getCallerNumber() { return callerNumber; }
28
29    // Gets the Receiver Number
30    public PhoneNumber getReceiverNumber() { return receiverNumber; }
31
32    // Gets the duration of the Call
33    public long getDuration() { return duration; }
34
35    // Sets the duration of the Call
36    public void setDuration(int newDuration) { this.duration = newDuration; }
37
38    // Returns the Call type
39    public CallType getCallType() { return callType; }
40
41    // Displays call information
42    public void displayCallInfo() { no usages
43        System.out.println("[i] INFO: CALL ID IS " + callID);
44        System.out.println("[i] INFO: CALLER NUMBER IS " + callerNumber);
45        System.out.println("[i] INFO: RECEIVER NUMBER IS " + receiverNumber);
46        System.out.println("[i] INFO: DURATION " + duration);
47        System.out.println("[i] INFO: CALL TYPE " + callType);
48    }
49
50    // Call Type Enumeration to ensure that it has specific values
51    public enum CallType { 3 usages
52        INCOMING, no usages
53        OUTGOING, no usages
54        MISSED no usages
55    }
56 }
```

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 μέθοδοι 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

```
1 package mainpackage;
2
3 import java.util.*;
4
5
6 public class Bill { no usages
7     // The Bill ID
8     private final int billID; 3 usages
9     // The PhoneNumber associated with the Bill
10    private PhoneNumber phoneNumber; 4 usages
11    // The Bill's Month
12    private int month; 4 usages
13    // The Bill's Charges
14    private List<Call> charges; 4 usages
15    // The Total Amount to pay the Bill
16    private double totalAmount; 3 usages
17    // Whether the Bill isPaid or not ( true or false )
18    private boolean isPaid; 3 usages
19
20    // Constructor
21    public Bill(int billID, PhoneNumber phoneNumber, int month, List<Call> charges) { 2 usages
22        this.billID = billID;
23        this.phoneNumber = phoneNumber;
24        this.month = month;
25        this.charges = charges;
26        // Initially, the bill is 0, then the cost is calculated
27        this.totalAmount = 0.0;
28        // Initially, the bill is not paid
29        this.isPaid = false;
30    }
31
32    // Gets the ID of the Bill
33    public int getBillID() { return billID; }
34
35    // Gets the phoneNumber of the Bill
36    public PhoneNumber getPhoneNumber() { return phoneNumber; }
37
38    // Sets the phoneNumber of the Bill
39    public void setPhoneNumber(PhoneNumber newPhoneNumber) { this.phoneNumber = newPhoneNumber; }
40
41    // Gets the Month of the Bill
42    public int getMonth() { return month; }
43
44    // Sets the Month of the Bill
45    public void setMonth(int newMonth) { this.month = newMonth; }
46
47    // Gets the Charges of the Bill
48    public List<Call> getCharges() { return charges; }
49
50    // Sets the Charges of the Bill
51    public void setCharges(List<Call> newCharges) { this.charges = newCharges; }
52
53    // Gets the Total Amount of the Bill
54    public double getTotalAmount() { return totalAmount; }
55
56    // Gets the Total Amount of the Bill
57    public boolean getBillStatus() { return isPaid; }
58
59    // Display Bill Info
60    public void displayBillInfo() { no usages
61        System.out.println("[i] INFO: BILL ID IS " + billID);
62        System.out.println("[i] INFO: BILL PHONE NUMBER IS " + phoneNumber);
63        System.out.println("[i] INFO: BILL MONTH " + month);
64        System.out.println("[i] INFO: BILL TOTAL AMOUNT " + totalAmount);
65        System.out.println("[i] INFO: BILL STATUS " + (isPaid ? "PAID" : "OUTSTANDING"));
66        System.out.println("[i] INFO: CHARGES");
67        for (Call charge : charges) {
68            System.out.println("\t- " + charge.toString());
69        }
70    }
71}
```

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 μέθοδοι 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

```
1  package main.model;
2
3  // The enum of Role to ensure only specific values
4  public enum Role {
5      CLIENT,
6      SELLER,
7      ADMIN;
8
9      public String toLowerCase() {
10         return this.name().toLowerCase();
11     }
12 }
```

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 βάσεις

<https://www.mockaroo.com/>