

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai

Step 1 ending part:

```
##### Installation on Windows #####
dist:
pre-run-deploy:
Distributing C:\Users\User\Desktop\year2_semester_1_stuff\cos30041\all_labs\week4_lab\ED-JEE-DTO\ED-JEE-DTO-appclient\dist\ED-JEE-DTO-appclient.jar
post-run-deploy:
run-deploy:
Copying 1 file to C:\Users\User\Desktop\year2_semester_1_stuff\cos30041\all_labs\week4_lab\ED-JEE-DTO\ED-JEE-DTO-appclient\dist
Copying 2 files to C:\Users\User\Desktop\year2_semester_1_stuff\cos30041\all_labs\week4_lab\ED-JEE-DTO\ED-JEE-DTO-appclient\dist\ED-JEE-DTO-appclientClient
Warning: C:\Users\User\Desktop\year2_semester_1_stuff\cos30041\all_labs\week4_lab\ED-JEE-DTO\ED-JEE-DTO-appclient\dist\gfdeploy\ED-JEE-DTO-appclient does not exist.
Record with primary key 000001 could not be created in the database table!
Record with primary key 000007 could not be created in the database table!
Type the number to perform the desired action:
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Get records at an Address
5: Exit
```

Step 2 all parts:

```
private Myuser myDTO2DAO(MyuserDTO myuserDTO) {
    Myuser myuser = new Myuser();
    myuser.setUserId(myuserDTO.getUserId());
    myuser.setName(myuserDTO.getName());
    myuser.setPassword(myuserDTO.getPassword());
    myuser.setEmail(myuserDTO.getEmail());
    myuser.setPhone(myuserDTO.getPhone());
    myuser.setAddress(myuserDTO.getAddress());
    myuser.setSecqn(myuserDTO.getSecqn());
    myuser.setSecans(myuserDTO.getSecans());
    return myuser;
}

private MyuserDTO myDAO2DTO(Myuser myuser) {
    MyuserDTO smth = new MyuserDTO(myuser.getUserId(), myuser.getName(), myuser.getPassword(), myuser.getEmail(), myuser.getPhone(), myuser.getAddress(), myuser.getSecqn(), myuser.getSecans());
    return smth;
}

@Override
public MyuserDTO getRecord(String userId) {
    Myuser myuser = find(userId);
    if (myuser != null) {
        return new MyuserDTO(myuser.getUserId(), myuser.getName(), myuser.getPassword(), myuser.getEmail(), myuser.getPhone(), myuser.getAddress(), myuser.getSecqn(), myuser.getSecans());
    } else {
        return null;
    }
}

@Override
public boolean updateRecord(MyuserDTO myuserDTO) {
    Myuser myuser = find(myuserDTO.getUserId());
    if (myuser != null) {
        try {
            edit(myDTO2DAO(myuserDTO)); // to add an object to database
            return true;
        } catch (Exception ex) {
            throw ex;
        }
    } else {
        // ...
    }
}
```

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai

```
        } else {
            return false;
        }
    }

    @Override
    public boolean deleteRecord(String userId) {
        Myuser myuser = find(userId);
        if (myuser != null) {
            try {
                remove(myuser);
                return true;
            } catch (Exception ex) {
                throw ex;
            }
        } else {
            return false;
        }
    }

    @Override
    public ArrayList<MyuserDTO> getRecordsByAddress(String address) {
        javax.persistence.Query query;
        query = em.createNamedQuery("Myuser.findByAddress").setParameter("address", address);
        List<Myuser> daoList = query.getResultList();
        ArrayList<MyuserDTO> dtoList = new ArrayList<>();
        for (Myuser user : daoList) {
            dtoList.add(myDAO2DTO(user));
        }
        return dtoList;
    }
}
```

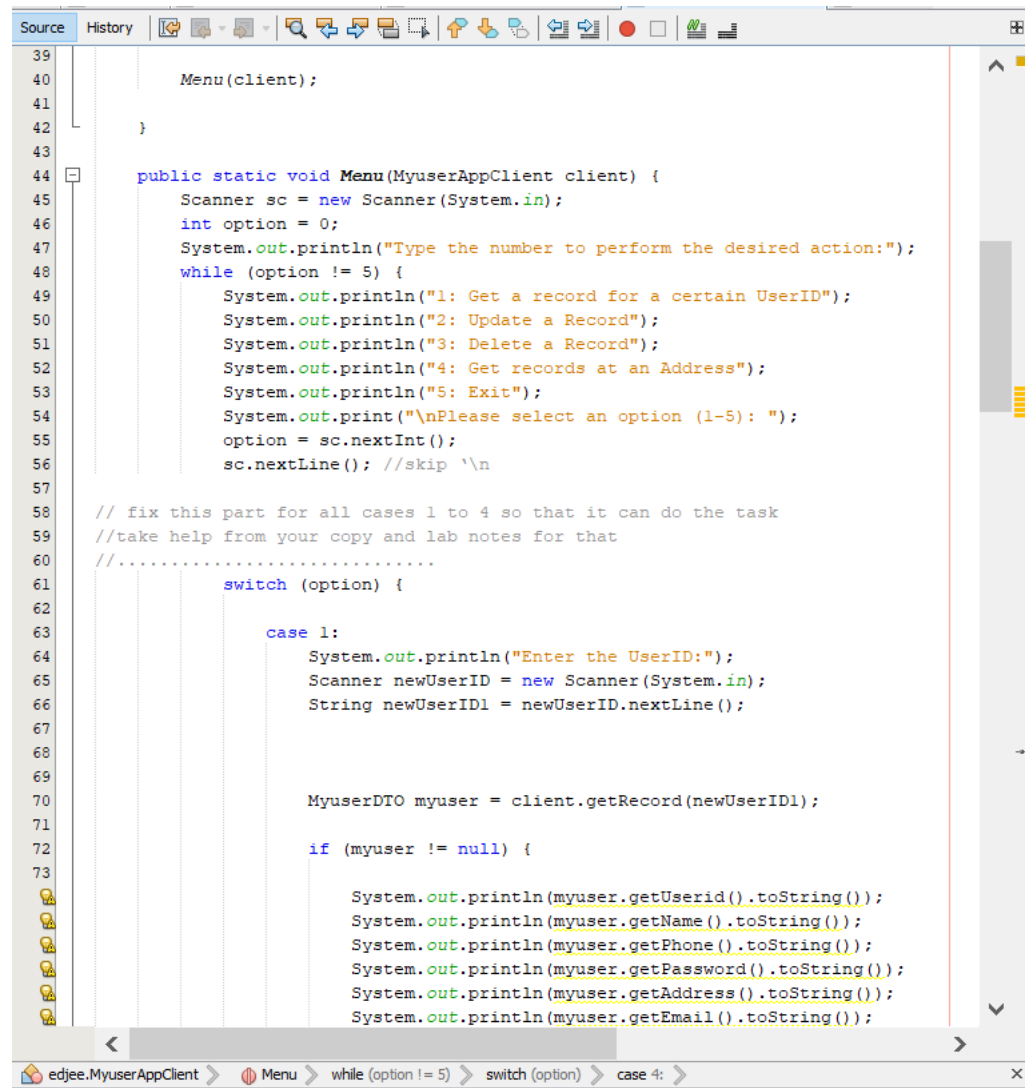
{Note: here my mistake has been I had written findby here instead of findBy for the address part. Thus the program couldn't understand what I was asking it to do as the rest of the named queries given in myuser were using findBy and not findby}

Step 3 all code parts:

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai



The screenshot shows an IDE window with a Java source file. The code defines a `Menu` method that takes a `MyuserAppClient` object. It uses a `Scanner` to read user input and a `while` loop to display a menu. The menu options are: 1: Get a record for a certain UserID, 2: Update a Record, 3: Delete a Record, 4: Get records at an Address, and 5: Exit. The code includes comments for fixing the menu options and taking help from a copy and lab notes. The `switch` statement handles the menu options, and the `case 1:` block shows how to retrieve and print user details from the `MyuserDTO` object.

```
39
40     Menu(client);
41
42 }
43
44 public static void Menu(MyuserAppClient client) {
45     Scanner sc = new Scanner(System.in);
46     int option = 0;
47     System.out.println("Type the number to perform the desired action:");
48     while (option != 5) {
49         System.out.println("1: Get a record for a certain UserID");
50         System.out.println("2: Update a Record");
51         System.out.println("3: Delete a Record");
52         System.out.println("4: Get records at an Address");
53         System.out.println("5: Exit");
54         System.out.print("\nPlease select an option (1-5): ");
55         option = sc.nextInt();
56         sc.nextLine(); //skip '\n'
57
58         // fix this part for all cases 1 to 4 so that it can do the task
59         //take help from your copy and lab notes for that
60         //.....
61         switch (option) {
62
63             case 1:
64                 System.out.println("Enter the UserID:");
65                 Scanner newUserID = new Scanner(System.in);
66                 String newUserID1 = newUserID.nextLine();
67
68
69                 MyuserDTO myuser = client.getRecord(newUserID1);
70
71                 if (myuser != null) {
72                     System.out.println(myuser.getUserid().toString());
73                     System.out.println(myuser.getName().toString());
74                     System.out.println(myuser.getPhone().toString());
75                     System.out.println(myuser.getPassword().toString());
76                     System.out.println(myuser.getAddress().toString());
77                     System.out.println(myuser.getEmail().toString());
78                 }
79             }
80     }
81 }
```

edjee.MyuserAppClient > Menu > while (option != 5) > switch (option) > case 4: >

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai

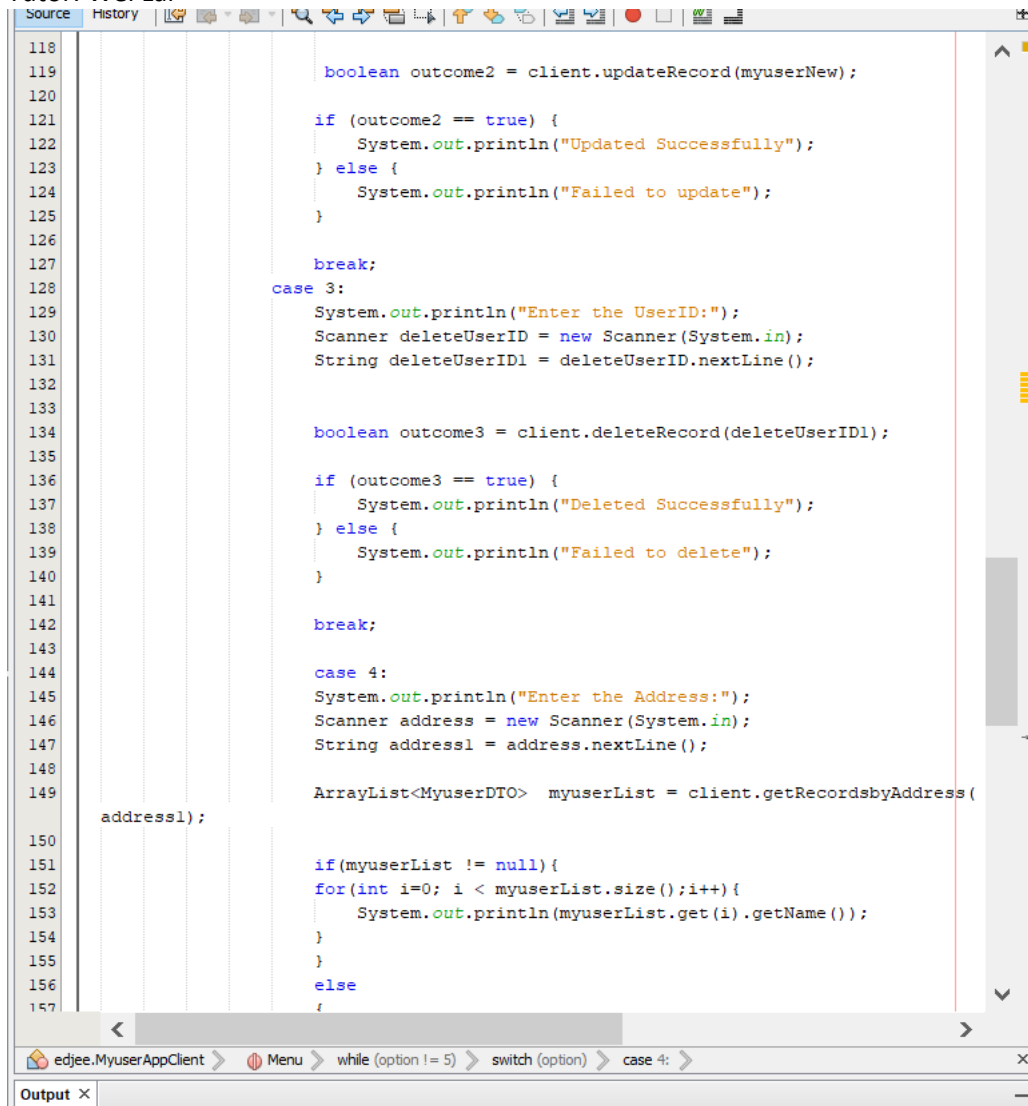
```
82      System.out.println(myuser.getSecAns().toString());
83  }
84  else{
85      System.out.println("UserID is not found");
86  }
87  }
88
89  break;
90
91  case 2:
92      System.out.println("Enter record details to be updated:");
93      Scanner newData = new Scanner(System.in);
94
95      System.out.println("UserId:");
96      String newUserId2 = newData.nextLine();
97
98      System.out.println("Name:");
99      String newName = newData.nextLine();
100     System.out.println("Password:");
101     String newPassword = newData.nextLine();
102     System.out.println("Email:");
103     String newEmail = newData.nextLine();
104     System.out.println("Phone:");
105     String newPhone = newData.nextLine();
106     System.out.println("Address:");
107     String newAddress = newData.nextLine();
108     System.out.println("SECQN:");
109     String newSecQn = newData.nextLine();
110     System.out.println("SECAns:");
111     String newSecAns = newData.nextLine();
112
113
114
115
116     MyuserDTO myuserNew = new MyuserDTO(newUserId2, newName,newPas
sword,newEmail,newPhone,newAddress,newSecQn,newSecAns);
117
118
119     boolean outcome2 = client.updateRecord(myuserNew);
120
```

edjee.MyuserAppClient > Menu > while (option != 5) > switch (option) > case 4: >

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai



```
118
119
120     boolean outcome2 = client.updateRecord(myuserNew);
121
122     if (outcome2 == true) {
123         System.out.println("Updated Successfully");
124     } else {
125         System.out.println("Failed to update");
126     }
127
128     break;
129
130 case 3:
131     System.out.println("Enter the UserID:");
132     Scanner deleteUserID = new Scanner(System.in);
133     String deleteUserID1 = deleteUserID.nextLine();
134
135     boolean outcome3 = client.deleteRecord(deleteUserID1);
136
137     if (outcome3 == true) {
138         System.out.println("Deleted Successfully");
139     } else {
140         System.out.println("Failed to delete");
141     }
142
143     break;
144
145 case 4:
146     System.out.println("Enter the Address:");
147     Scanner address = new Scanner(System.in);
148     String address1 = address.nextLine();
149
150     ArrayList<MyuserDTO> myuserList = client.getRecordsbyAddress(
151         address1);
152
153     if(myuserList != null){
154         for(int i=0; i < myuserList.size();i++){
155             System.out.println(myuserList.get(i).getName());
156         }
157     }
158     else
159     {
160     }
```

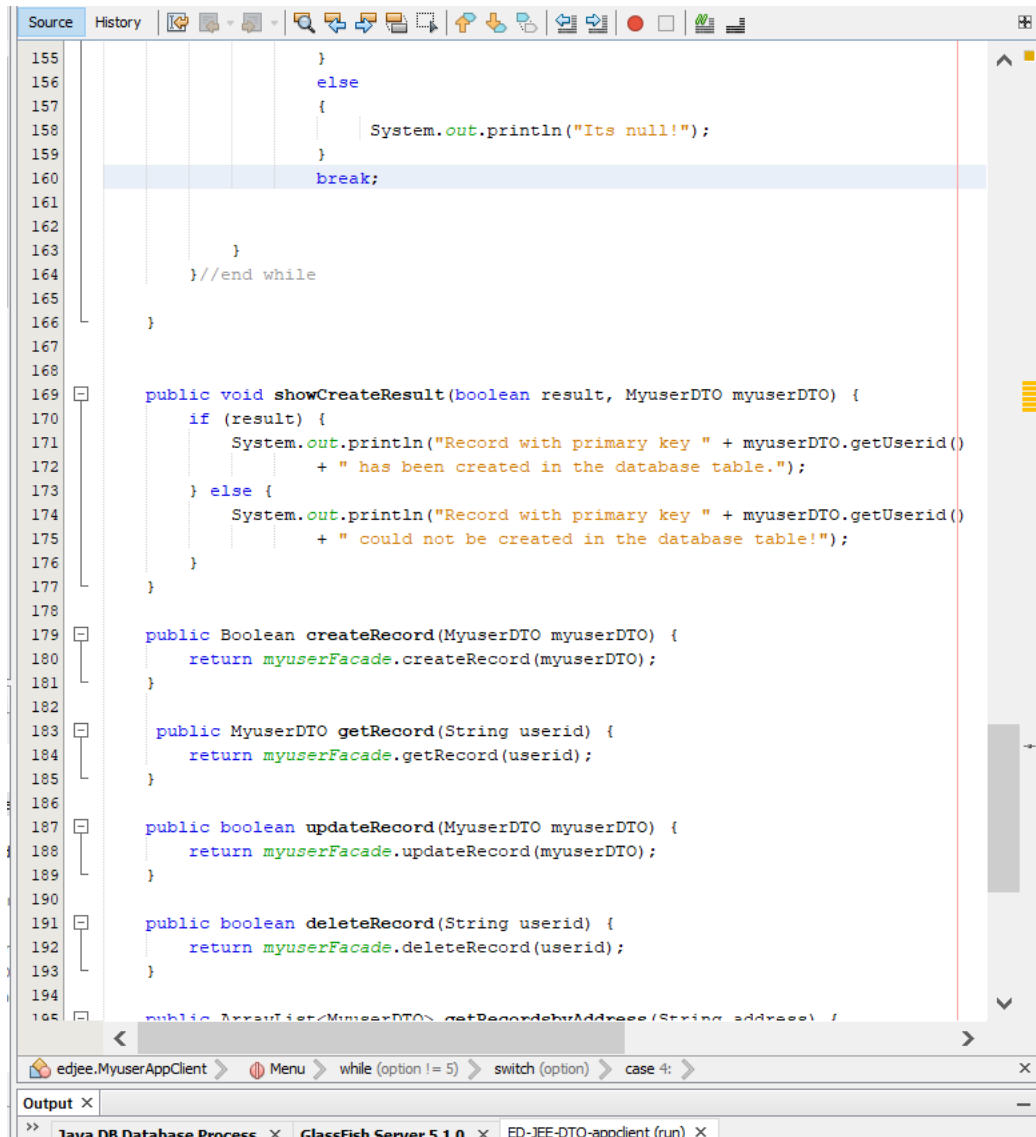
edjee.MyuserAppClient > Menu > while (option != 5) > switch (option) > case 4: >

Output ×

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai



```
155     }
156     else
157     {
158         System.out.println("Its null!");
159     }
160     break;
161
162
163     }
164     } //end while
165
166 }
167
168
169 public void showCreateResult(boolean result, MyuserDTO myuserDTO) {
170     if (result) {
171         System.out.println("Record with primary key " + myuserDTO.getUserid()
172             + " has been created in the database table.");
173     } else {
174         System.out.println("Record with primary key " + myuserDTO.getUserid()
175             + " could not be created in the database table!");
176     }
177 }
178
179 public Boolean createRecord(MyuserDTO myuserDTO) {
180     return myuserFacade.createRecord(myuserDTO);
181 }
182
183 public MyuserDTO getRecord(String userid) {
184     return myuserFacade.getRecord(userid);
185 }
186
187 public boolean updateRecord(MyuserDTO myuserDTO) {
188     return myuserFacade.updateRecord(myuserDTO);
189 }
190
191 public boolean deleteRecord(String userid) {
192     return myuserFacade.deleteRecord(userid);
193 }
194
195 public ArrayList<MyuserDTO> getRecordsByAddress(String address) {
```

edjee.MyuserAppClient > Menu > while (option != 5) > switch (option) > case 4: >

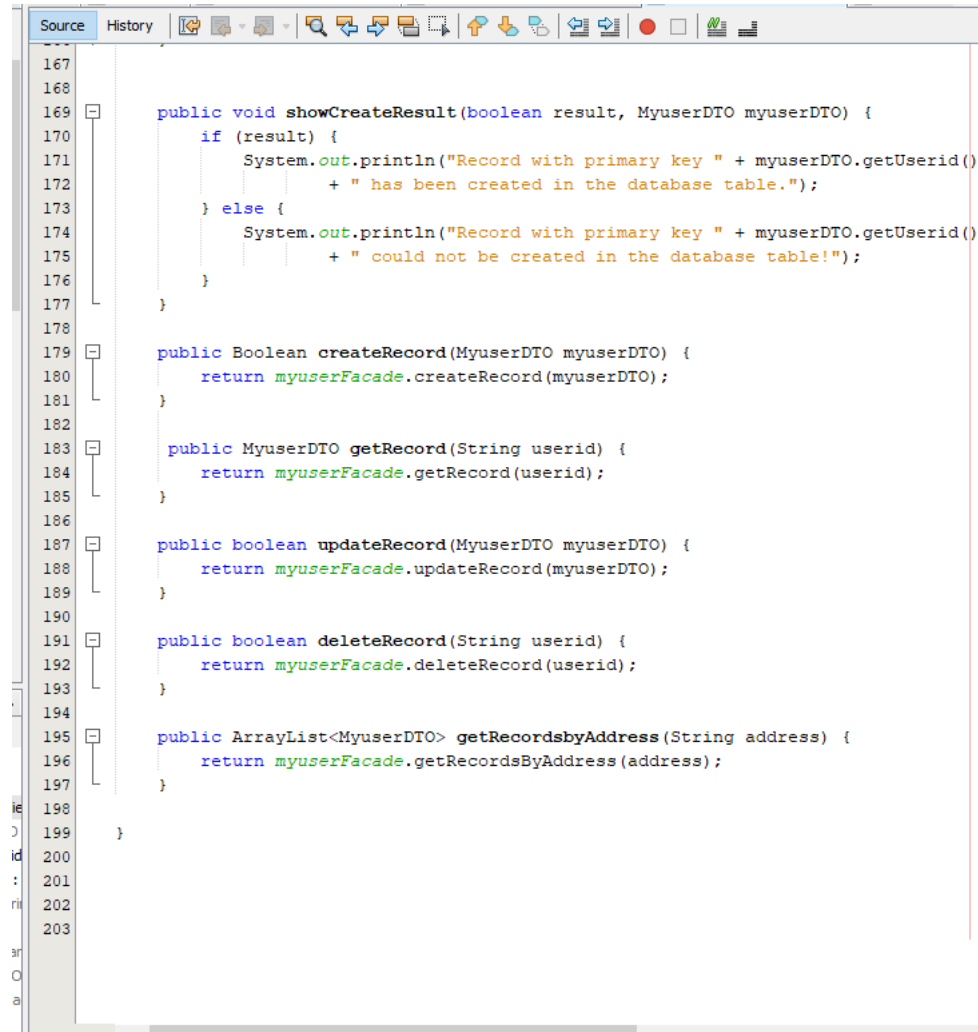
Output X

>> Java DB Database Process X GlassFish Server 5.1.0 X ED-JEE-DTO-appclient (run) X

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai



The screenshot shows an IDE window with a 'Source' tab and a toolbar. The code is a Java class with several methods. Line numbers 167 through 203 are visible on the left margin. The code defines a facade for user management operations, including creating, getting, updating, deleting records, and getting records by address. The methods use a facade object named 'myuserFacade' to delegate the actual database operations.

```
167
168
169 public void showCreateResult(boolean result, MyuserDTO myuserDTO) {
170     if (result) {
171         System.out.println("Record with primary key " + myuserDTO.getUserid()
172             + " has been created in the database table.");
173     } else {
174         System.out.println("Record with primary key " + myuserDTO.getUserid()
175             + " could not be created in the database table!");
176     }
177 }
178
179 public Boolean createRecord(MyuserDTO myuserDTO) {
180     return myuserFacade.createRecord(myuserDTO);
181 }
182
183 public MyuserDTO getRecord(String userid) {
184     return myuserFacade.getRecord(userid);
185 }
186
187 public boolean updateRecord(MyuserDTO myuserDTO) {
188     return myuserFacade.updateRecord(myuserDTO);
189 }
190
191 public boolean deleteRecord(String userid) {
192     return myuserFacade.deleteRecord(userid);
193 }
194
195 public ArrayList<MyuserDTO> getRecordsbyAddress(String address) {
196     return myuserFacade.getRecordsByAddress(address);
197 }
198
199 }
200
201
202
203
```

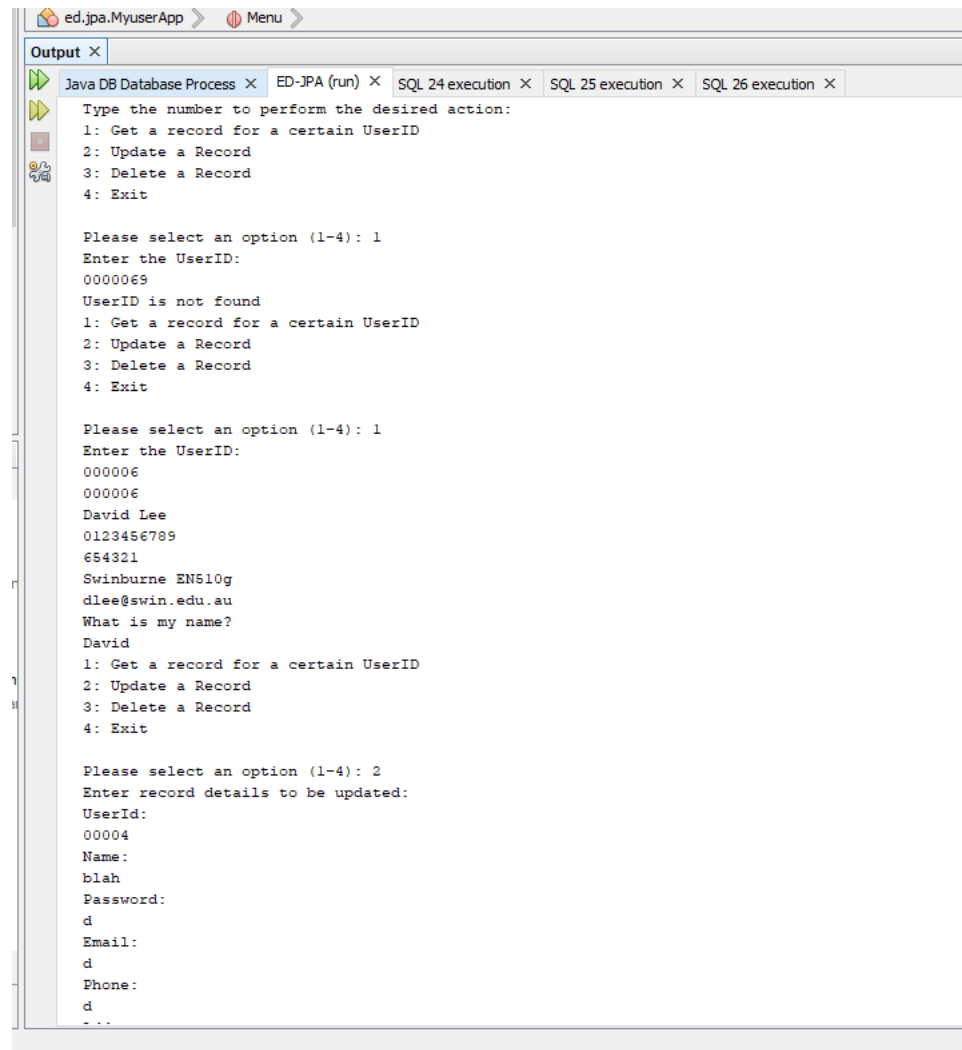
Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai

```
48 public void showCreateResult(boolean result, MyuserDTO myuserDTO) {
49     if (result) {
50         System.out.println("Record with primary key " + myuserDTO.getUserid()
51                             + " has been created in the database table.");
52     } else {
53         System.out.println("Record with primary key " + myuserDTO.getUserid()
54                             + " could not be created in the database table!");
55     }
56 }
57
58 public boolean createRecord(MyuserDTO myuserDTO) {
59     return mydb.createRecord(myuserDTO);
60 }
61
62 //can also directly call it with client in the main code, but doing it like this to follow the format given in createRecord for good practise:
63 public MyuserDTO getRecord(String userid) {
64     return mydb.getRecord(userid);
65 }
66
67 public boolean updateRecord(MyuserDTO myuserDTO) {
68     return mydb.updateRecord(myuserDTO);
69 }
70
71 public boolean deleteRecord(String userid) {
72     return mydb.deleteRecord(userid);
73 }
74 }
75 }
```

Step 3 outputs:



```
ed.jpa.MyuserApp  Menu
Output
Java DB Database Process x ED-JPA (run) x SQL 24 execution x SQL 25 execution x SQL 26 execution x
Type the number to perform the desired action:
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 1
Enter the UserID:
0000069
UserID is not found
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 1
Enter the UserID:
000006
000006
David Lee
0123456789
654321
Swinburne ENS10g
dlee@swin.edu.au
What is my name?
David
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 2
Enter record details to be updated:
UserId:
00004
Name:
blah
Password:
d
Email:
d
Phone:
d
...
```


Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai

```
Output X
Java DB Database Process X ED-JPA (run) X SQL 24 execution X SQL 25 execution X SQL 26 execution X

Please select an option (1-4): 2
Enter record details to be updated:
UserId:
00004
Name:
blah
Password:
d
Email:
d
Phone:
d
Address:
d
SECQN:
d
SECAns:
d
Failed to update
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 2
Enter record details to be updated:
UserId:
000006
Name:
s
Password:
s
Email:
s
Phone:
s
Address:
s
SECQN:
s
SECAns:
s
```

```
Output X
Java DB Database Process X ED-JPA (run) X SQL 24 execution X SQL 25 execution X SQL 26 execution X

UserId:
000006
Name:
s
Password:
s
Email:
s
Phone:
s
Address:
s
SECQN:
s
SECAns:
s
Updated Successfully
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 3
Enter the UserID:
000000006
Failed to delete
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 3
Enter the UserID:
000001
Deleted Successfully
1: Get a record for a certain UserID
2: Update a Record
3: Delete a Record
4: Exit

Please select an option (1-4): 4
BUILD SUCCESSFUL (total time: 1 minute 43 seconds)
```

Name: S M Ragib Rezwan

ID: 103172423

Tutor: Wei Lai

4.1) Here myuser is the DAO object and thus is the one directly accessing the database when the object is created. On the other hand MyuserFacade is just a bean type class. So myuser is responsible for the ORM work

4.2) basically this means that for each session bean, there is always a certain number of bean instances available for use in the pool. So, when a client sends a request, the pool can quickly be used to fulfill the request, and when it's over, the instance can be returned back to pool for later reuse. So the same bean instance can be used to fulfill the request of lots of clients later on, which in turn make it scalable.