Cyrus Edgren 101606991
Tutor: Wei Lai

4.1P

Task 1.

The lab was completed and produced the following output:

```
Record with primary key 000001 could not be created in the database table!
Record with primary key 000007 has been created in the database table.
run:
BUILD SUCCESSFUL (total time: 12 seconds)
```

Task 2.

1. The function 'myDAO2DTO()' was created:

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

2. The function 'getRecord()' was created:

```
@Override
public MyuserDTO getRecord(String userId) {
    Myuser myuser = find(userId);

    if (myuser == null) {
        return null;
    } else {
        return myDAO2DTO(myuser);
    }
}
```

3. The function 'updateRecord()' was created:

```
@Override
public boolean updateRecord(MyuserDTO myuserDTO) {
    if (find(myuserDTO.getUserid()) == null) {
        return false;
    }

    try {
        Myuser myuser = this.myDTO2DAO(myuserDTO);
        this.edit(myuser);
        return true;
    } catch (Exception ex) {
        return false;
    }
}
```

Cyrus Edgren 101606991
Tutor: Wei Lai

4. The function ‘deleteRecord()’ was created:

```
@Override
public boolean deleteRecord(String userId) {
    Myuser myuser = find(userId);

    if (myuser == null) {
        return false;
    }

    try {
        this.remove(myuser);
        return true;
    } catch (Exception ex) {
        return false;
    }
}
```

5. The function ‘getRecordsByAddress()’ was created:

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

        return dtoList;
    }
}
```

Cyrus Edgren 101606991
Tutor: Wei Lai

Task 3.

The same code for the simple menu was taken from previous tasks and modified slightly to work with the new features of this task.

Creating and getting a record:

```
Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

2

 Please enter userId:
000009

 Please enter name:
Max

 Please enter password:
pass12

 Please enter email:
max@mail.com

 Please enter phone:
0983748374

 Please enter address:
Swinburne EN510f

 Please enter secret question:
Dogs name

 Please enter secret answer:
Ruf
Record created successfully.



Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

1

 Please enter the UserId of the user you would like to retrieve:
000009
UserId: 000009
Name: Max
Password: pass12
Email: max@mail.com
Phone: 0983748374
Address: Swinburne EN510f
Secret Qustion: Dogs name
Secret Answer: Ruf
```

Cyrus Edgren 101606991
Tutor: Wei Lai

Updating a record:

```
Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

3

 Please enter userId:
000009

 Please enter name:
John

 Please enter password:
12pass

 Please enter email:
john@mail.com

 Please enter phone:
0938473842

 Please enter address:
Swinburne EN511a

 Please enter secret question:
Cats name

 Please enter secret answer:
Snowball

 Record updated successfully.


Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

1

 Please enter the UserId of the user you would like to retrieve:
000009
UserId: 000009
Name: John
Password: 12pass
Email: john@mail.com
Phone: 0938473842
Address: Swinburne EN511a
Secret Qustion: Cats name
Secret Answer: Snowball
```

Cyrus Edgren 101606991
Tutor: Wei Lai

Getting users by address:

```
Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

5

 Please enter the Address that you would like to find:
Swinburne EN5lla


UserId: 000002
Name: James T. Kirk
Password: 234567
Email: jkirk@swin.edu.au
Phone: 8765432109
Address: Swinburne EN5lla
Secret Qustion: What is my name?
Secret Answer: James



UserId: 000009
Name: John
Password: l2pass
Email: john@mail.com
Phone: 0938473842
Address: Swinburne EN5lla
Secret Qustion: Cats name
Secret Answer: Snowball
```

Deleting a record:

```
Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

4

 Please enter the UserId of the user you would like to delete:
000009

 User deleted succussfully.


Please choose what operation you would like to perform:
1: Get
2: Create
3: Update
4: Remove
5: Get by address
6: Exit

1

 Please enter the UserId of the user you would like to retrieve:
000009

No user with that id could be found.
```

Task 4.

4.1

Myuser is the class that is responsible for the ORM work, as it is the object which holds the data and is used to directly manipulate the database with the values it contains.

4.2

Stateless session bean pooling is the process of having pre-prepared instances of an object that can each be used for a single transaction. As each transaction uses a different bean, there is no memory of what occurred in each transaction and separate beans can not act on values that were passed with other beans. In this task, 'MyuserFacade' was the class that defined the bean and what it could do. Bean instance pooling can help scalability through performance optimization, and also through the split multi-layer approach it requires when dealing with a client, database and business layer.

Code can be found for further review at:
https://github.com/CyrusEdgren/Secure_Scalable_Software/tree/master/4.1P