3.1P

1. METHOD

```
public MyuserDTO getRecord(String userid){

    Myuser myuser = findMyuser(userid);

    return new MyuserDTO(myuser.getUserid(),
            myuser.getName(),
            myuser.getPassword(),
            myuser.getEmail(),
            myuser.getPhone(),
            myuser.getAddress(),
            myuser.getSecqn(),
            myuser.getSecans()
    );
}

public boolean deleteRecord(String userid){

    Myuser toBeDeleted = findMyuser(userid);

    if (toBeDeleted != null)
    {
        em.getTransaction().begin();
        em.remove(toBeDeleted);
        em.getTransaction().commit();
        return true;
    }

    return false;
}
```

```java
public boolean updateRecord(MyuserDTO myuserDTO) {
    Myuser myuser = findMyuser(myuserDTO.getUserid());

    if (myuser != null)
    {
        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 true;
    }

    return false;
}
```

Test harness

```java
// GET RECORD TESTS
// Returns correct record
MyuserDTO getRecordResult = client.getRecord("000001");

if (myuserDTO.getUserid().equals(getRecordResult.getUserid()))
{
    System.err.println("SUCESS: getRecord returns correct DTO");
}
else{
    System.err.println("FAIL: getRecord returns wrong DTO");
}

// Returns null if record does not exist
try {
    MyuserDTO getRecordNull = client.getRecord("000010");
}
catch(NullPointerException ex){
    System.err.println("SUCESS: getRecord returns null when no user exists");
}
```

```
// RETURNS TRUE IF RECORD IS DELETED
boolean deleteResult = client.deleteRecord("000001");
boolean objectDeleted = false;

try {
    client.getRecord("000001");
}
catch (NullPointerException ex){
    objectDeleted = true;
}

if(deleteResult && objectDeleted){
    System.err.println("SUCESS: MyUser deleted");
} else
{
    System.err.println("FAILURE: Object not deleted");
}

// RETURNS FALSE ON INVLALID DELETE
deleteResult = client.deleteRecord("123456");

if(deleteResult){
    System.err.println("FAILURE: Invalid delete returns true");
} else
{
    System.err.println("SUCESS: Valid delete returns false");
}


MyuserDTO updatedDTO = new MyuserDTO("000006", "David Update", "12345",
        "update@swin.edu.au", "98765432", "Update EN510g",
        "What is my update?", "Update");

boolean updateSucceded = client.updateRecord(updatedDTO);
MyuserDTO updatedResult = client.getRecord(updatedDTO.getUserid());

if (updateSucceded &&
        updatedDTO.getUserid().equals(updatedResult.getUserid()) &&
        updatedDTO.getName().equals(updatedResult.getName()) &&
        updatedDTO.getPassword().equals(updatedResult.getPassword()) &&
        updatedDTO.getPhone().equals(updatedResult.getPhone()) &&
        updatedDTO.getAddress().equals(updatedResult.getAddress()) &&
        updatedDTO.getEmail().equals(updatedResult.getEmail()) &&
        updatedDTO.getSecAns().equals(updatedResult.getSecAns()) &&
        updatedDTO.getSecQn().equals(updatedResult.getSecQn())
        ) {
    System.err.println("SUCESS: Update Succeeded");
}
else
{
     System.err.println("FAILURE: Update Failed");
}
```

Questions

4.1. Which class is responsible for doing all the ORM work? MyuserDB / Myuser? Justify your answer

Myuser is responsible for the object relation mapping as it is an entity. Entities define the database tables using the object properties.

This is why myuser uses the @column tag on its properties

```
private static final long serialVersionUID = 1L;
@Id
@Basic(optional = false)
@Column(name = "USERID")
private String userid;
@Column(name = "NAME")
private String name;
@Column(name = "PASSWORD")
private String password;
@Column(name = "EMAIL")
private String email;
@Column(name = "PHONE")
private String phone;
@Column(name = "ADDRESS")
private String address;
@Column(name = "SECQN")
private String secqn;
@Column(name = "SECANS")
private String secans;
```

4.2. Is the role of the Myuser entity class here different from that of the “Myuser” class in the application in Lab 02. Why? Justify your answer.

It is significantly different here as an entities fields are persisted to the database. This means that altering a field from a Myuser object returned from the EntityManager will alter that field in the database.

This means that Myuser in lab 3 takes on the role of DAO whereas in lab 2 it was only a POJO.

4.3. Why Myuser entity class in this application domain is different from that in the Lab 02 (in the JDBC domain)?

Its role was different here as the javax persistence library requires us to define entity objects in order to interact with the database.