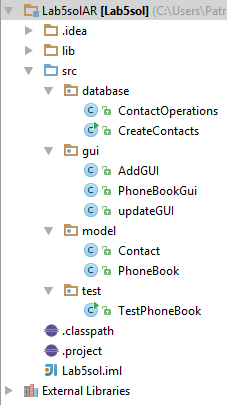
**Lab 5**

Download the file Lab5Skeleton.zip from Moodle. Extract and open this project in IntelliJ.

The project contains the following files:

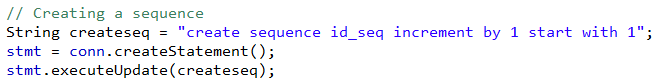


1. **Database Package**

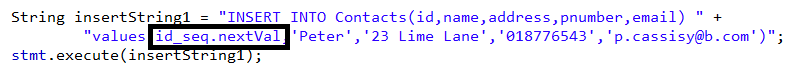
**CreateContacts Class**

Select the file CreateContacts.java and execute it. Make sure you have set your username and password in the code. This program when executed creates a Contacts table and inserts 4 rows into the table.

In this program the primary key has been set using a sequence. Here is the code from the program that creates a sequence called **id\_seq**. Line 73



In each INSERT statement note the use of the Oracle **NEXTVAL** function which is used to retrieve the next value in a sequence.



The method dropContactsTable() now contains code to remove the sequence when the table is dropped.

**Contact Operations Class**

Take a look at this class and make sure that you understand the methods - read the comments in the code

Next let's look at the two classes in the model.

1. **Model Package**

**Contact Class**

You will see that there are 2 constructors here one with 4 parameters and one with 5 parameters. The constructor with 4 parameters is needed when we add a new contact - the user will only supply 4 values because other value is supplied by the database using the sequence.

The constructor with 5 parameters is used when we retrieve the contacts from the database (including the id). We create Contact object references each with 5 parameters and these can then be added to an array list.

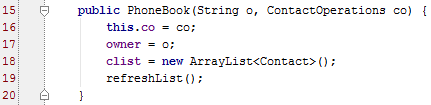
The rest of the code in the Contact class is comprised of regular getter and setter methods

**PhoneBook Class**

In this class note the two reference variables of type ResultSet and ContactOperations. We need these in order to communicate with the database class called ContactOperations.

Take a look at the constructor. The parameters represent the owner of the phone book (String) and the reference variable co (ContactOperations)

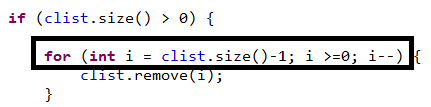
These two parameters are used to initialise the two member variables. The array list is also initialised here. There is a call to a method refreshList() which is covered next.



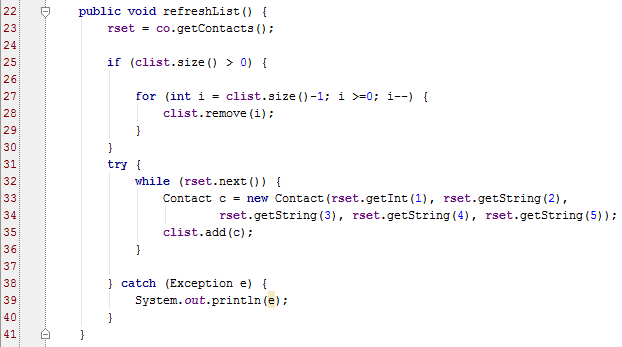
In order to ensure that our array list contains fresh data we need to write a method to retrieve the data from the Contacts table and use this data to create Contact object references and add them to the array list.

Looking at the code we see the first line initialises the rset variable by calling on the database method getContacts(). If you look at this method in the ContactOperations class you will see that it returns a ResultSet object reference which contains all the data in the Contacts table ordered by the id.

In order to update the arraylist one way to do this is to empty it and then fill it again with new data. The If statement below checks if the array list is not empty and then calls on the remove() method. Note that the counter i in the For loop doesn't start at 0. the reason for this is that if you are iterating from 0 to size inside the loop and deleting items this will reduce the size of the list which will fail when you try to access the indexes which are greater than the effective size(the size after the deleted items). This is the reason that the For loop looks like this:



The try catch block contains a while loop which iterates through the result set, creates a Contact object reference and adds it to the array list.



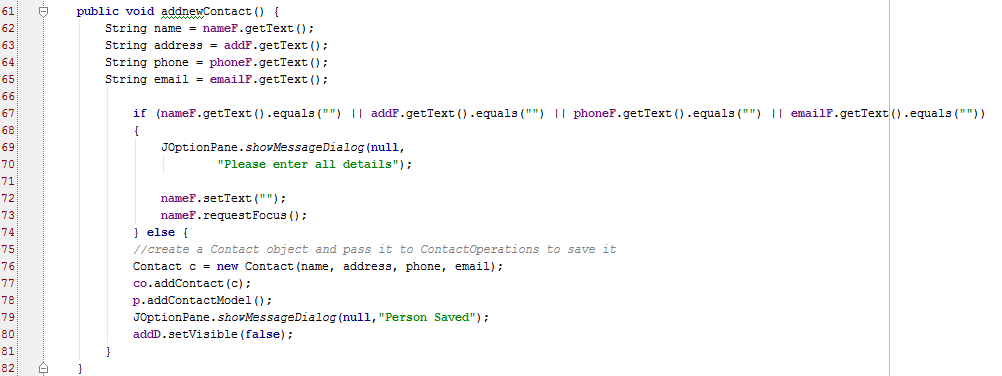
1. GUI & Model Packages

You will add code to the GUI classes in order to add, update and delete contacts from both the database and the array list.

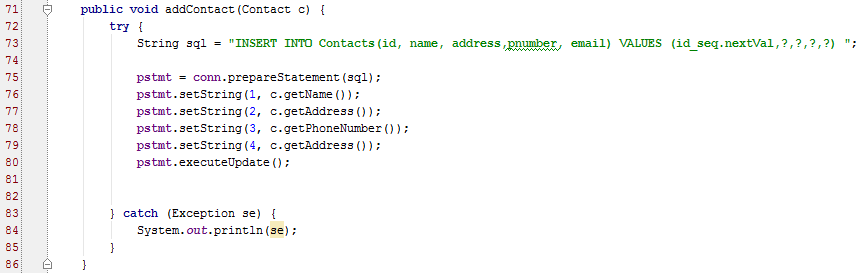
Let's start with adding a new contact

**Add Contact**

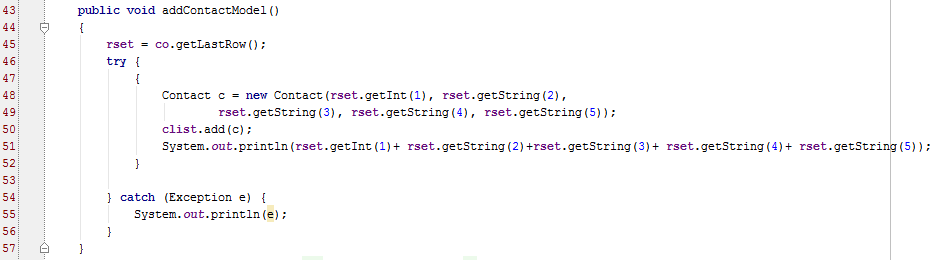
Switch to the AddGUI class and add the following code:

Add in the following code for the addNewContact() method 

Inside the ContactOperations class you need to code the following method:



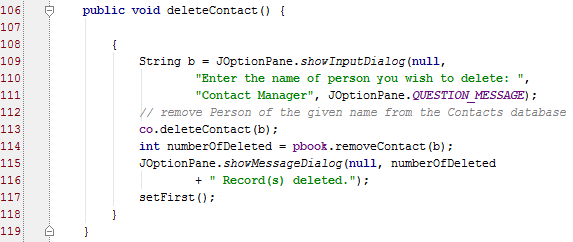
Now you need to code the method addContactModel() in the Phonebook class:



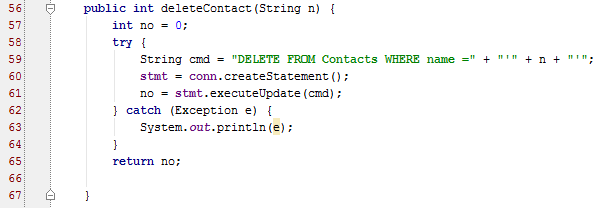
Note: Make sure the Add is working before you move on to the next part of the exercise.

**Delete Contact**

When the user presses the delete button, a window is displayed asking for the name of the user whose contact should be deleted. You need to write a method called deleteContact() in the PhoneBookGUI class. The code is given below:

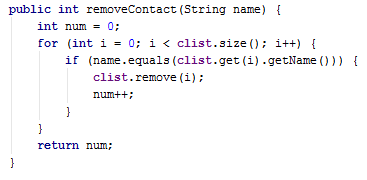


See Line 113 above – this is a call to a method deleteContact() which is part of the ContactOperations class (database package). The code for this method is shown below:



See Line 114 above – this is a call to a method removeContact

Code this method as part of the PhoneBook class (model package):

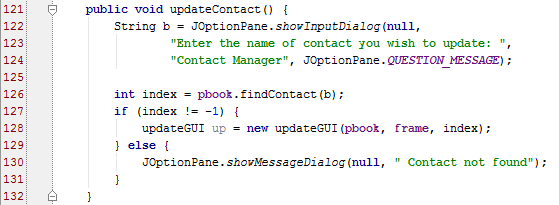


This method searches the array list for the contact and if found then removes the contact from the list.

**Update Contact**

All the code for Update is provided - read the code and make sure you understand it. An explanation is provided below.

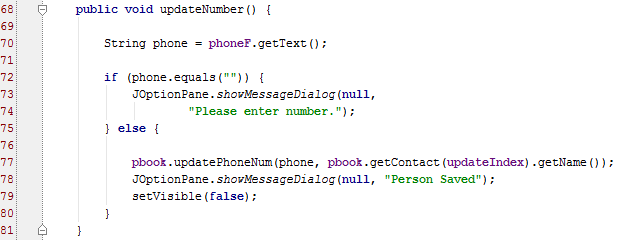
When the user presses the Update button, the following method is called – you need to add this method to your code. This method displays a window asking for the name of the user whose contact should be updated.



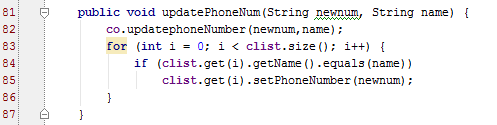
Line 126 - there is a method call to findContact(). This method returns the location of the contact in the arraylist if it exists, otherwise -1 is returned. if the contact is found then the following code is executed which launches the Update GUI



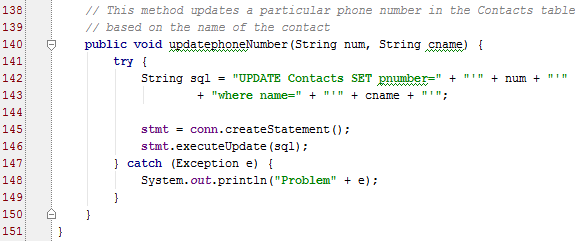
The user enters in the new phone number on the GUI and presses Save. When the Save button is pressed the following method is called – you need to add this code to the UpdateGUI class:



Looking at this method you will see a call to the updatePhoneNum() method at line 77 in the phoneBook class (model package). This method has two parameters and the code for the method is shown below – add this code to the Phonebook class.

.

There is a call to method updatephoneNumber() on line 82 which is part of the ContactOperations class (database package). The code for this method is shown below – add it to the ContactOperations class:



The rest of the code in the updatePhoneNum() class searches the array list for the contact name and if found then updates the phone number in the list.

**Question**

Can you think of a better way to update the array list when a new contact is added by the user. The way it is done here we empty the array list and fill it again with all the records from the database. All we need is the new record from the database.