Tutorial 01 **SWT 22022 – Practical for Internet Application Development**

Title: Java EE Project with Java Database

Submission Date: 22nd April 2020

Type of Submission: ZIP the project and send through the below link

https://www.dropbox.com/request/HGH94chk9pw1vWn1bA0D

Lab Sheet Covered: Lab sheet 10 & 11

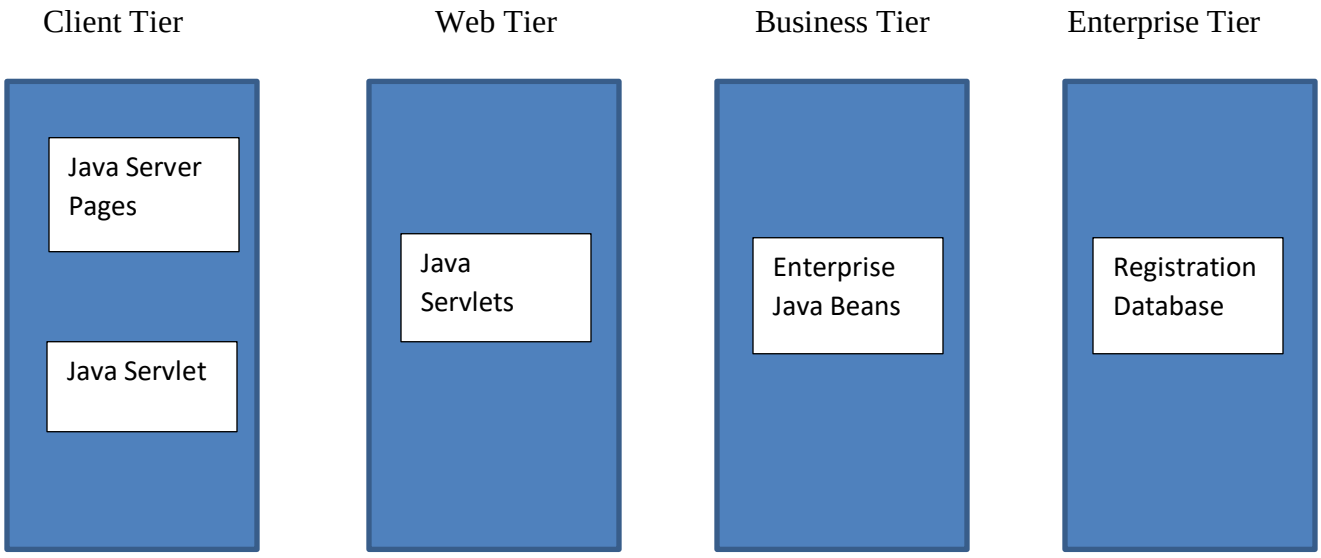
Overview:

In this tutorial we are going to create a simple Java Enterprise Edition (J2EE) project using Netbeans and Glassfish.

There are four tiers:–

1. Client Tier
2. Web Tier
3. Business Tier
4. Enterprise Tier

and this project will put something in each of these tiers.

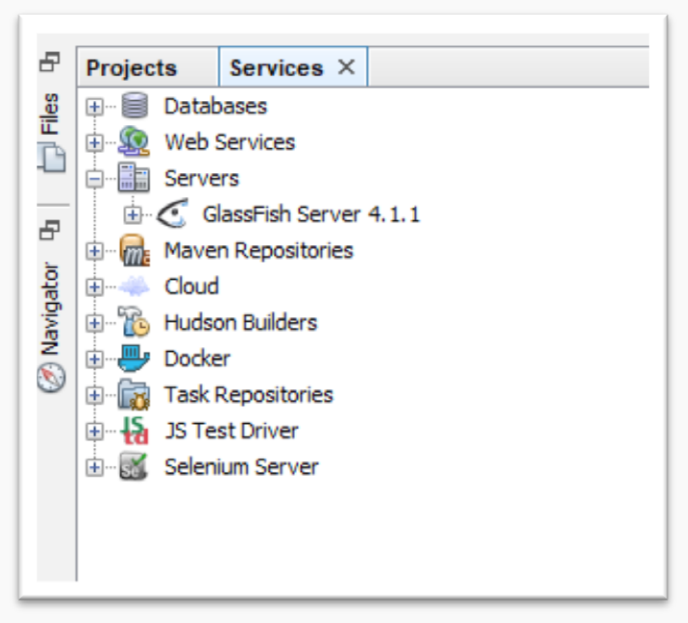


The task is to have a user register their information into a web page. This web page will update a database, and then show a confirmation page that the client is registered. There will also be another web page that will display all the information on the table. The final completed project is starting

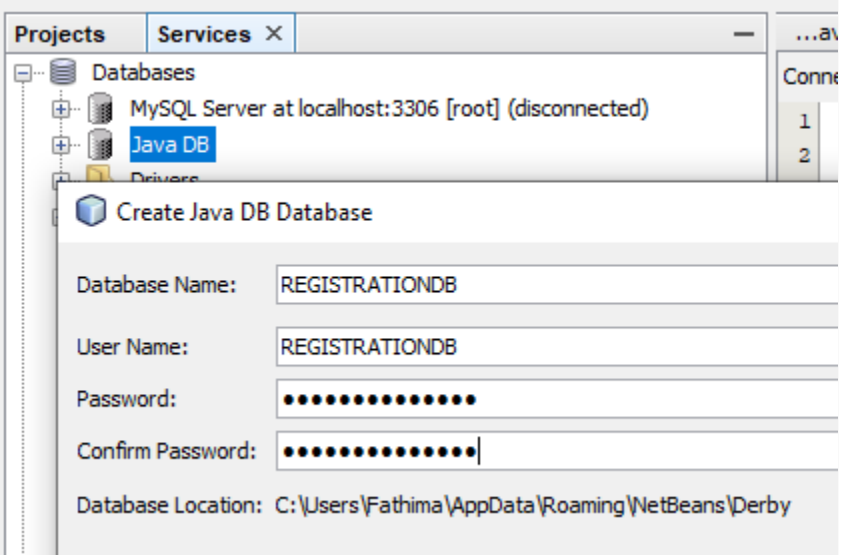
with the registration page, a new person is successfully registered displaying the confirmation page, then we look at the web page that displays the information on the registration database table and we see the new person was added successfully.

We are going to create the entire project using the Netbeans IDE 8.0.2 and Glassfish 4.1 server. We create the database for our project (this will be in the Enterprise Tier). First we start the Glassfish 4.1 server.

Go to Services tab, under the Servers start the Glassfish by right click.

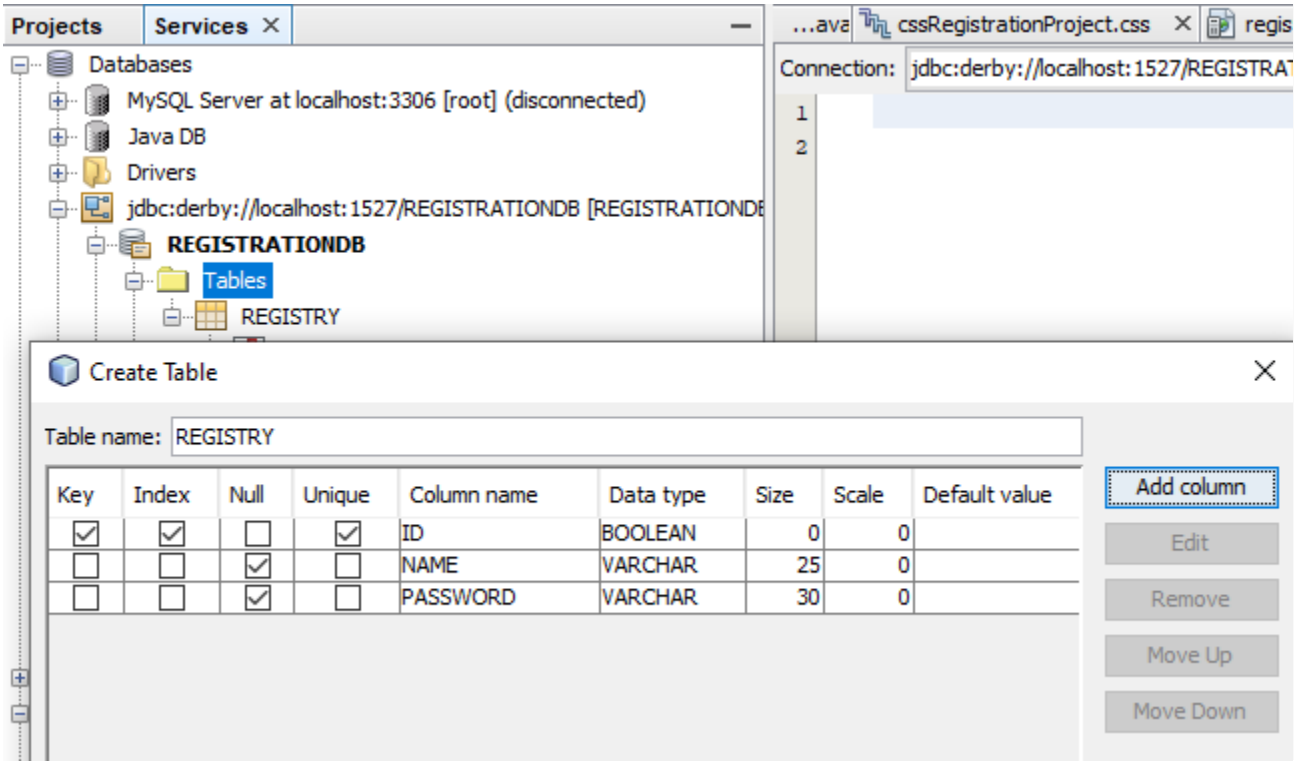


Next we create a database then we create a table on that database. So expand Databases and click on Java DB then create database.(Let's give the same wording for all the fields)

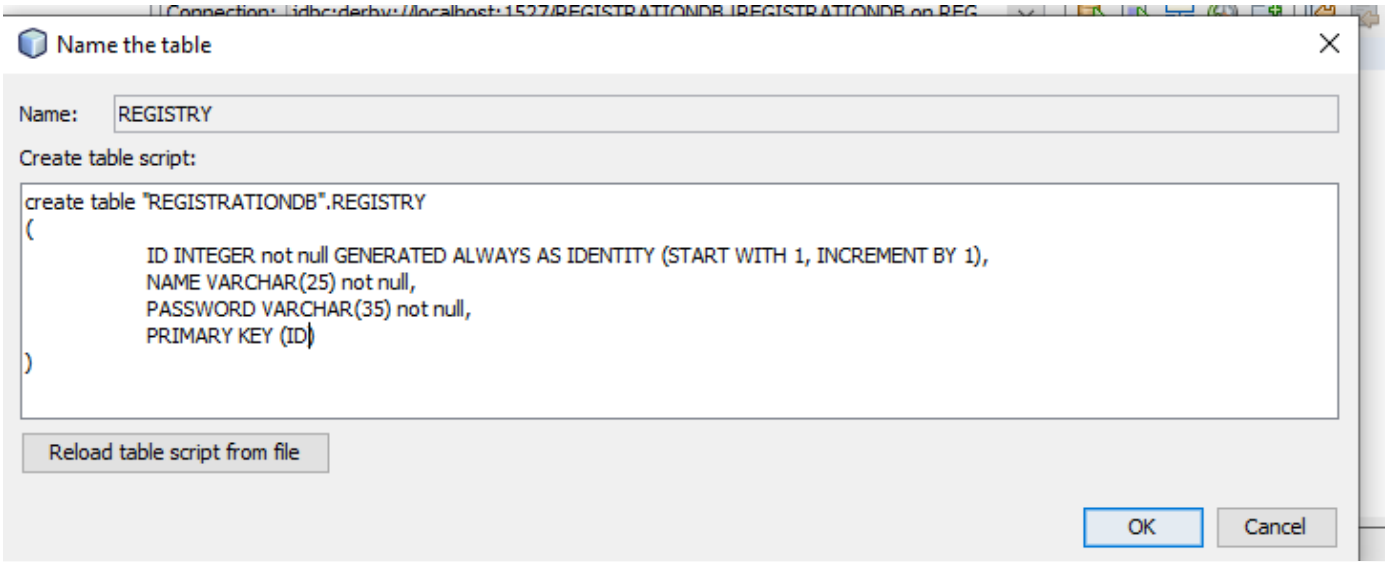


10.04.2020

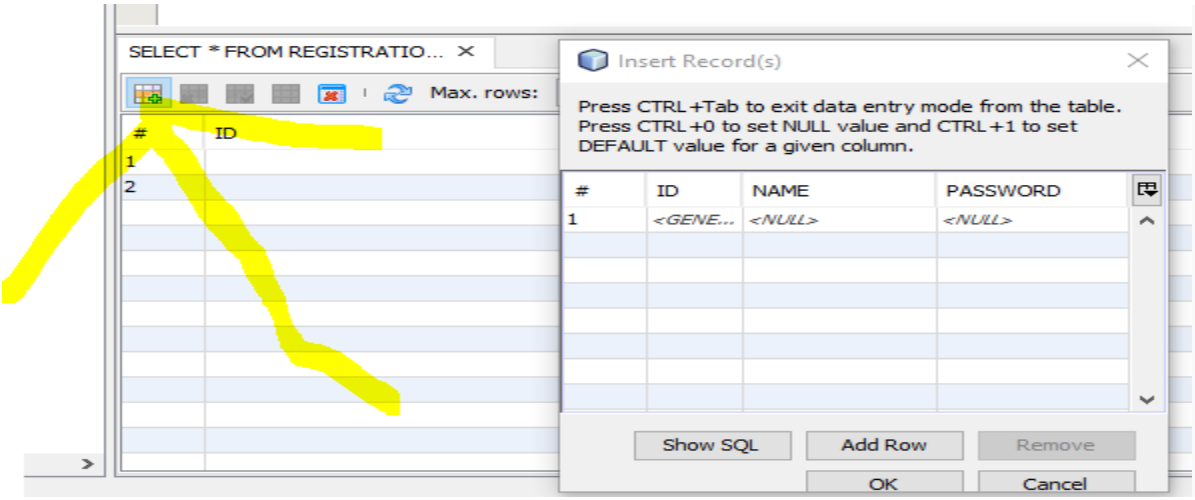
Create the table called REGISTRY and add columns as shown below.



Here we can see there is no feature available to make the ID to auto increment. So we do the following steps. Make a backup of the structure of that table using the Grab Structure feature in Netbeans and we delete the table. Then recreate the table (using the recreate feature of Netbeans) and before recreating the table, we want to edit the table script. We then add the code to auto increment the primary key for this table and we recreate the table.

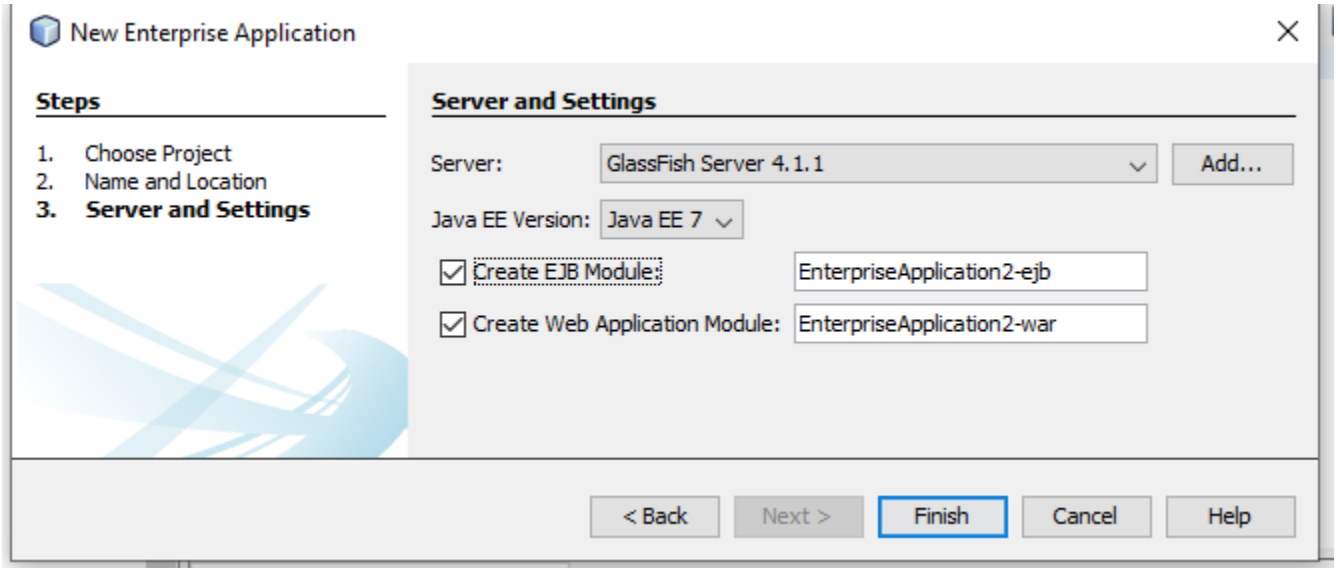


We then add 2 records showing the auto increment works for the table. Click on insert record and add data to the table on the window popup.

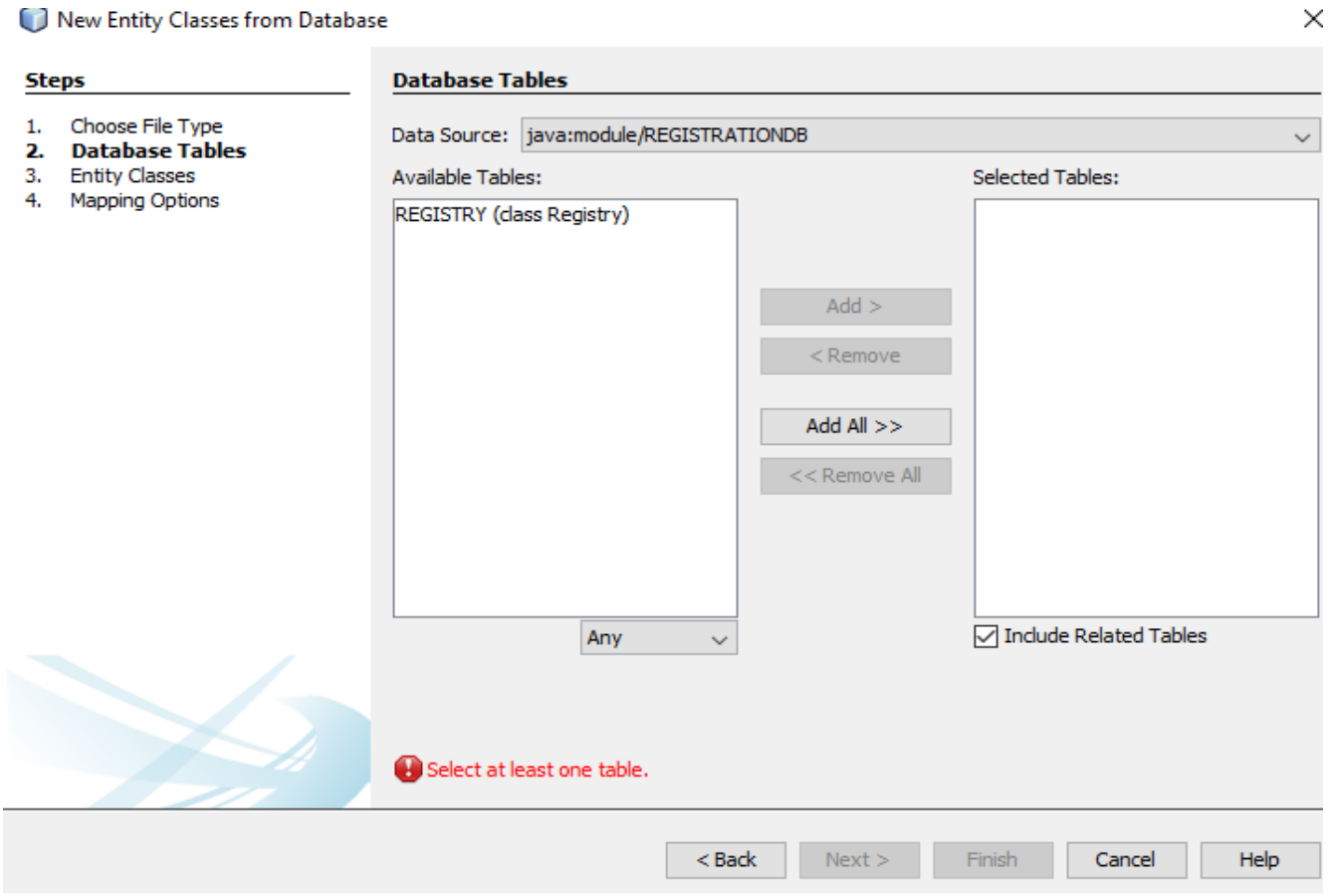


Now we create a project. File -> New Project -> Java EE –> Enterprise Application -> Next.

Project name “RegistrationProject” -> Next -> Glassfish Server + Java EE 7 + select 2 modules as follows and finish.



We are going to look at the Business Tier of the project. So we are going to create the Enterprise Java Beans in the Business Tier which is allow us to communicate with database that is Enterprise Tier of the project. Right click on the RegistrationProject-ejb and create a new – other – Persistence - Entity Classes from Database. Upon data source add our database using create data source by entering JNDI Name as REGISTRATIONDB.



Add and select table. Next give the package name as entityBeanPkg. Click next in Mapping Options select Fully Qualified Database Table Names.

This will create our java file with the queries to select the data from the database as well as the constraint annotations and the getter and setter methods. Then we again right click on the RegistrationProject-ejb module and select new – other – Enterprise JavaBeans – Session Bean for Entity Classes. We will create a local interface for this bean. This will create three more java files the abstractfacade, the facade, and the facadelocal. The facadelocal program will be the interface program we will connect from our servlets to send and receive data on the database. We need to deploy the RegrstrationProject-ejb module. If the build is not successful please check the Glassfish upgrade.

We create a Servlet (in the client tier) that uses the ejb local interface (in the Bisiness Tier) to display information from the database (in the Enterprise Tier). I right click on the RegistrationProject-war module to create a new – other – web – servlet. Name the servlet as DisplayRegistryServlet. Create the package name displayRegistryServletPkg. Click next, add web.xml and finish. I type in the annotation @EJB and private to access the interface FacadeLocal ejb created above. We then need to fix imports to prevent the error messages. I then remove the current HTML and paste in HTML code in attachment 1 (DisplayRegistryServlet.txt).

As we can see at the code pasted in we can add some CSS. Rright click on the RegistrationProject-war module to create a CSS, new – other – web – cascading style sheet (and paste in code from

attachment 2 cssRegistrationProject.txt). Right clicking on the RegistrationProject (triangle) clean and build project, then right click again on the triangle and run. We need to type in the servlet name to bring it up, and we see the results in our browser window.

We complete the project by creating the Registration Page and the Confirmation Page. We right click on the RegistrationProject-war module new – other – Web – Servlet. Name it as RegistrationServlet and the package is RegistrationServletPkg, and we are not going to add web.xml for this servlet. So leave it. Click finish. We want this servlet to talk to a Enterprise Java Bean, like in previous servlet we add the @EJB annotation and RegistryFacadeLocal. Right click on the error message and fix imports.

Like we add in our previous servlet remove the default html code and replace the code from the attachment 3 (RegistrationServlet.txt). In the code we can see that the data validation is available. If it is valid it will move to RegistrationConfirmation page. Not valid back to the form with the error message. Inside the valid data the registryFacade.create, it is in the RegistryFacadeLocal interface. Here we are using create and findAll() methods. So comment all the other. And save that.

Go back to the RegistrationServlet, after we create the item in the database, we set the attributes so that we can send through the RegistrationConfirmation page. This servlet is not in the client tier, it is in the web tier (It is not seen by the client).

We create two JSP webpages for the registration page and the confirmation page and paste in some code available from the attachment 4 & 5.Let's create the Registration page, right click on the RegistrationProject-war module new – other – Web – JSP. Next name it as RegistrationForm. The registration page input form asks the client to type in the information requested (note: no data validation is done in this program, but should be done in a real world environment) when the submit button is pressed, the information is passed to the servlet, which will check the data(note: all field should be validated for a real world environment). If there is invalid data, we go back to the registration page with an error message, if the data passes validation, the facadeLocal interface will insert the new record into the table. If the create is successful the information will be passed to the confirmation page and displayed on that webpage. So select the html code and replace it with the code from the attachment 4 (registrationForm.txt).

Create the confirmation page with the name of RegistrationConfirmation in the same way as we created the RegistrationForm. So replace the html from the attachment 5 (RegistrationConfirmation.txt).

Build and Run the code. ☺

As I have already demonstrated you the Lab Sheet 10 & 11 during the practical session hope you can do it without any errors.