Department of Computer Science and Engineering

SDM College of Engineering and Technology, Dharwad-02

Lab Manual

Advanced Object-Oriented Programming

IV Semester (A & B div)

Academic Year: **2023-24**

Course Instructors

Prof. Indira R Umarji – A Division

Contents

1. Course objective and outcomes:

1.1 Course Learning Objectives (CLOs):

This laboratory course focuses on the following learning perspectives:

Develop applications in business, scientific and engineering domain, which includes mandatory exercise on each of the following features as separate exercises or a single system.

1.2 Course Outcomes (COs):

COs	Description of course outcomes	Mapping to Program Outcomes at different levels		
		Substantial (3)	Moderate (2)	Low (1)
CO1	**Develop** applications through CORE JAVA feature like: Events, Exceptions, built-in Java objects, Streams, Threads and Frameworks.	2,3	14,15	1
CO2	Build appropriate graphical user interface for a given problem specification.	2,3	14,15	1
CO3	**Apply** generics in Java to create classes for different algorithms.	14	3	1
CO4	Apply appropriate driver classes to connect back end databases and perform database operations required as per problem specification.	14	13	15,1
CO5	Write Java programs to explore networking capabilities and build applications	3	5,14	1,2
CO6	Explain the Java language features like: Servlets, JSP, AJAX and JavaScript to develop we-based applications.	3,14	2,13	1,5,15
CO7	Write Test-cases/scripts to verify the correctness of the program.	4,15	-	1

POs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Mapping Level	1.0	2.3	2.8	3.0	2.5	-	-	-	-	-	-	-	2.0	2.5	1.8	-

2. List of Experiments:

SL. No	Problem Description	COs
1	**Using** State Diagram Design and Describe the behavior of STACK which contains maximum of FOUR integer elements. Implement the above design in JAVA Programming Language. Design the TEST-DRIVER class to include minimum number of TEST CASES to test the complete features of STACK class designed. **Expected Learning:** How to define the class , Use of Instance Variables, data types, operators, control structures, Understanding of access specifies, Declaring methods, parameterized methods, constructor, Interface, finalize() method, Compilation procedures, use of package, class path and other basic features.	**1,3 & 7**
2	**Create** an appropriate GUI which allows the user to select an item from the menu. When draw menu item is selected, draws the selected shape(Allowed shapes are: Circle, Rectangle & Triangle) in drawing area by getting appropriate dimensions of the selected shape from the user through key Board entry, using the concept of ABSTRACT CLASS, INHERITANCE and DYNAMIC DISPATCH features of JAVA Programming Language. Code must be robust for all possible erroneous input conditions, displaying appropriate error messages in message window specially designed for them. **Expected Learning**: Abstract class, Inheritance, Runtime polymorphism, AWT, Event Handling, Exception Handling.	**1,2 & 7**
3	**Write** Java program to simulate LOST UPDATE or INCONSISTENT READ Transaction issues of database using MULTITHREADING features of JAVA and also Write java program to control the above concurrency issue. Output of the program to be displayed on the screen as well as to be written in a file of user choice. **Expected Learning**: Multithreading and Streams of java language	**1,4 & 7**
4	**Design** and **Implement** an APPLET for any computer game of your choice. Store the user name and the score of each game session in the database (MySQL). **Expected Learning**: Applet life cycle, Java Database connectivity, events, AWT/SWING components, Application Design and Implementation.	**1,2,4 & 7**
5	**Write** Java program to implement 1-1 chatting (text) using Networking features. **Expected Learning**: Networking, Application Design and Implementation.	**5,7**
6	**Using** SERVLETS, JSP and Database, JavaScript, AJAX (any database on cloud) connectivity features of JAVA language, implement a web based search tool that facilitates the searching of all possible books available in the department for a given subject. Search for a single book at a time is allowed. Results are to be displayed in the TABLE form. Any suitable assumption that is convenient for system development may be done. **Expected Learning:** Servlets, JSP, JavaScript, AJAX and Database Connectivity.	**4,6 & 7**

3. Software's required

3.1 Software's :

In this course, following open source software's are used to design and develop the applications given in the experiments list:

3.1.1 **Windows 7** operating system.

3.1.2 **JDK and JRE**, these software's can be downloaded from the following link- https://www.oracle.com/java/technologies/javase-downloads.html. Install JDK and follow the installation wizard.

3.1.3 **Eclipse IDE,** it can be downloaded from the following link- https://www.eclipse.org/downloads/packages/installer

3.1.4 **NetBeans IDE**, it can be downloaded from the following link- https://netbeans.org/downloads/6.5/index.html

Install any one IDE to practice Java Programs

3.1.5 **WampServer**, it can be downloaded from the following link- http://www.wampserver.com/

3.1.6 **XAMPP server**, it can be downloaded from the following link- https://www.apachefriends.org/download.html

Install any one Server

3.1.7 **MySQL connector (API) jar file,** it can be downloaded from the following link- https://dev.mysql.com/downloads/windows/installer/8.0.html

3.1.8 **Apache Tomcat server**, this software can be downloaded from the following link- https://tomcat.apache.org/whichversion.html

Note:

This manual demonstrates simple examples using the following software's:

i. NetBeans IDE - v7.1.1
ii. Eclipse Juno
iii. WampServer - v3. 2. 0
iv. MySQL Connector API- v5.1.23
v. Apache Tomcat- v7.0.59

4. Installation and system configurations:

4.1 Installing the JDK and the JRE on 64-Bit Windows Platform:

The **JDK** and the **JRE** have minimum processor, disk space, and memory requirements for 64-bit Windows platform. Before installing the JDK or the JRE on your 64-bit Windows platform, you must verify that it meets the following minimum processor, disk space, and memory requirements.

4.1.1 Processor Requirements:

Both the JDK and JRE require a minimum **a Pentium-II 266 MHz processor**.

4.1.2 Disk Space Requirements:

For JDK 10, you are given the option of installing the following features:

- Development Tools
- Source Code
- Public Java Runtime Environment

When you install 64-bit JDK, then 64-bit public JRE also gets installed. The following table provides the disk requirements for the installed features:

JDK	Installed Image
Development Tools: 64-bit platform	500 MB
Source Code	54.2 MB

JRE	Installed Image
Public Java Runtime Environment	200 MB
Java Update	2 MB

4.1.3 Memory Requirements:

On Windows 64-bit operating systems, the Java runtime requires a minimum of 128 MB of memory.

4.1.4 JDK Installation Instructions for Windows:

You run a self-installing executable file to unpack and install the JDK on Windows computers.

Install JDK on Windows computers by performing the actions described in the following topics:

- Downloading the JDK Installer
- Running the JDK Installer
- Setting the PATH Environment Variable

- **Downloading the JDK Installer: In a browser, go to the Java SE Development Kit 10 Downloads page and click** Accept License Agreement**. Under the** Download **menu, click the** Download **link that corresponds to the .exe for your version of Windows.**

  Example- Download the file jdk-10.*interim.update.patch*_windows-x64_bin.exe. (Similarly .exe file can be downloaded for Window 32 bit machine.)

- **Running the JDK Installer:** You must have administrator privilege to install the JDK on Microsoft Windows. To run the JDK installer:
    1. Start the JDK 10 installer by double-clicking the installer's icon or file name in the download location.
    2. Follow the instructions provided by the Installation wizard.
       The JDK includes the JavaFX SDK, a private JRE, and the Java Mission Control tools suite. The installer integrates the JavaFX SDK into the JDK installation directory.
    3. After the installation is complete, delete the downloaded file to recover the disk space.

- **Setting the PATH Environment Variable:** It is useful to set the PATH variable permanently for JDK so that it is persistent after rebooting. If you do not set the PATH variable, then you must specify the full path to the executable file every time that you run it.

  For example:

```
C :\> "C:\Program Files\Java\jdk-10\bin\javac" MyClass.java
```

  To set the PATH variable permanently, add the full path of the jdk-10\bin directory to the PATH variable. Typically, the full path is:

```
C:\Program Files\Java\jdk-10\bin
```

  To set the **PATH** variable on Microsoft Windows:
    1. Select **Control Panel** and then **System**.
    2. Click **Advanced** and then **Environment Variables**.
    3. Add the location of the bin folder of the JDK installation to the PATH variable in **System Variables**.

4.1.5 JRE Installation Instructions for Windows:

- When installing JRE on Windows computers, you must select the JRE installer that is appropriate for your Windows system.
- The 64-bit Windows operating systems come with a 64-bit Internet Explorer (IE) browser as the standard (default) for viewing web pages.
- Install JRE on Windows computers by performing the actions described in the following topics:
    - Downloading the JRE Installer
    - Running the JRE Installer

- **Downloading the JRE Installer:** The JRE Installer is located on the Java SE Runtime Environment 10 Downloads page
    - In a browser, go to the Java SE Runtime Environment 10 Downloads page.
      The following JRE installers are available for you to download:
        - **Windows Offline**: jre-10.*interim.update.patch*_windows-x64_bin.exe
        - **Windows Tar**: jre-10.*interim.update.patch*_windows-x64_bin.tar.gz
        - Download the JRE installer according to your requirement.

- Click Accept License Agreement, and then, under the Download menu, click the link that corresponds to the installer for your version of Windows.
- Note the file size specified on the download page and, after the download has completed, verify that you have downloaded the complete file.

- **Downloading the JRE Installer:** You must have Administrative privileges in order to install the JRE on Microsoft Windows.

To run the JRE installer:

- Start the JRE 10 Installer by double-clicking the installer's icon or file name in the download location.
- Follow the instructions provided by the Installation wizard.
  - The installer notifies you if Java content is disabled in web browsers and provides instructions for enabling it. If you previously chose to hide some of the security prompts for applets and Java Web Start applications, then the installer provides an option for restoring the prompts.
- After the installation is complete, delete the downloaded file to recover disk space.

4.1.6 Installation of Eclipse IDE: The required software can be downloaded from the link given the section-3.1.3. Following are the steps used to install required version of eclipse IDE in windows platform:

1. Download the **Eclipse Installer**.
2. Start the **Eclipse Installer** executable. ...
3. Select the package to **install**. ...
4. Select your **installation** folder. ...
5. Launch **Eclipse**. (Click windows button from your keyboard→All programs→select eclipse folder from the list→click eclipse application to launch the IDE)

4.1.7 Setting JRE to the Eclipse IDE: Open Eclipse IDE, load the project into **package explorer** window and then follow the below given steps:

1. Select Project in the eclipse IDE→Right click on project→ properties.
2. Select "Java Build Path" on left, then "**JRE** System Library", click Edit…
3. Select "Workspace Default **JRE**"
4. Click "Installed JREs"
5. If you see **JRE** you want in the list, select it (selecting a JDK is OK)

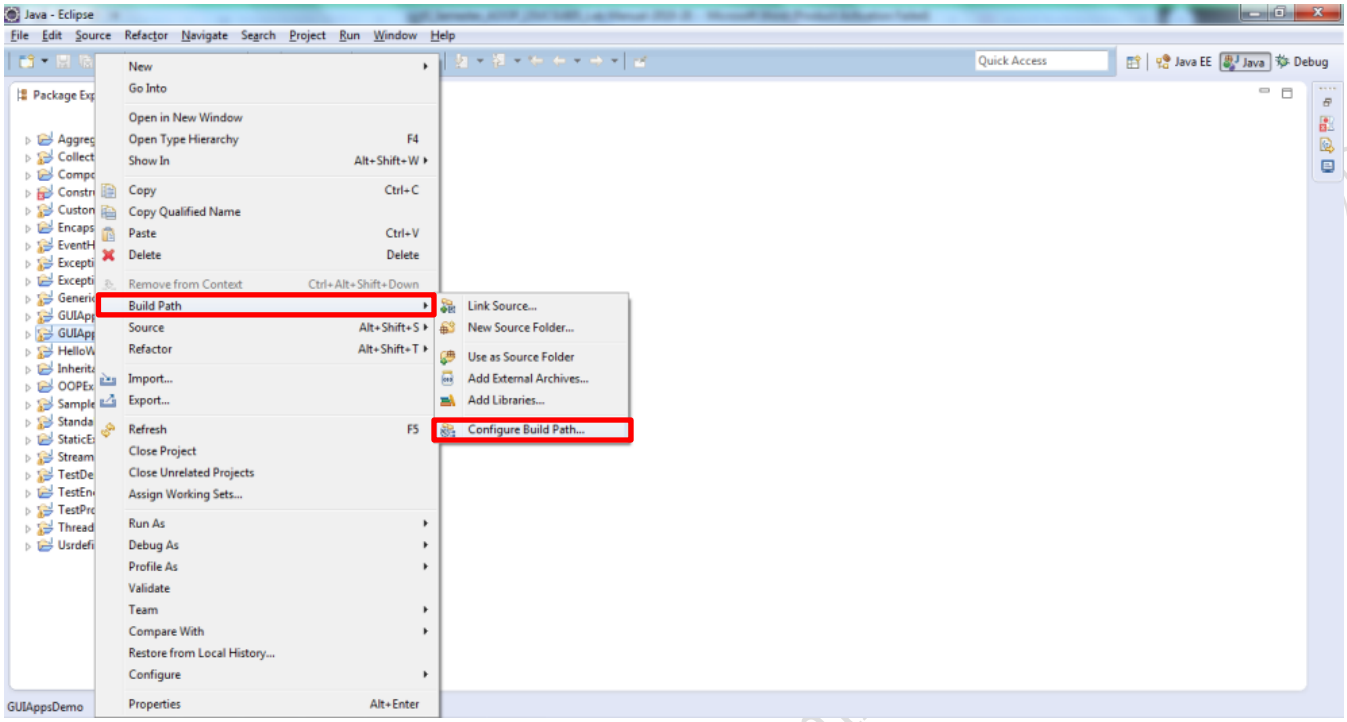

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Debug
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Team
Compare With
Restore from Local History...
Configure
Properties Alt+Enter
Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...
GUIAppsDemo

Fig:1

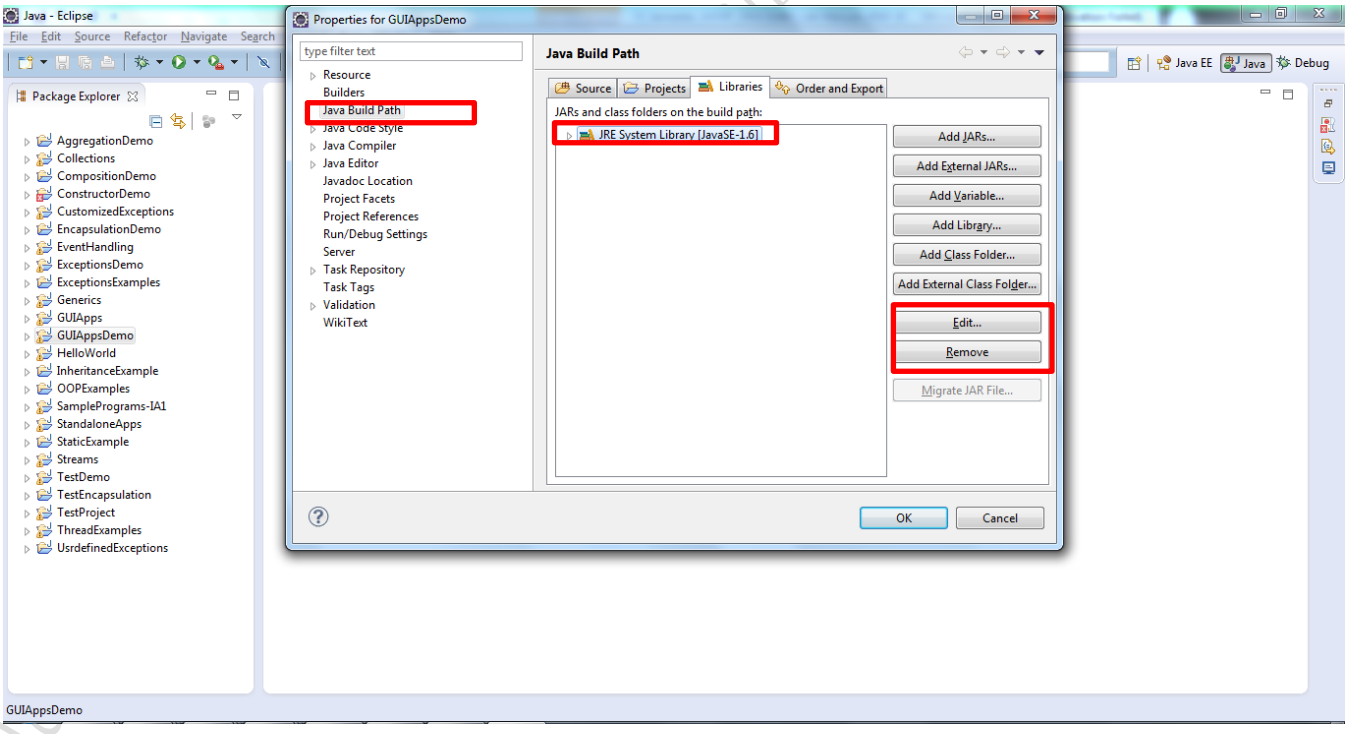

Java - Eclipse
Properties for GUIAppsDemo
type filter text
Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project References
Run/Debug Settings
Server
Task Repository
Task Tags
Validation
WikiText
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
JRE System Library [JavaSE-1.6]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
OK
Cancel
Package Explorer
AggregationDemo
Collections
CompositionDemo
ConstructorDemo
CustomizedExceptions
EncapsulationDemo
EventHandling
ExceptionsDemo
ExceptionsExamples
Generics
GUIApps
GUIAppsDemo
HelloWorld
InheritanceExample
OOPExamples
SamplePrograms-IA1
StandaloneApps
StaticExample
Streams
TestDemo
TestEncapsulation
TestProject
ThreadExamples
UsrdefinedExceptions
GUIAppsDemo

Fig: 2

4.1.8 Installation of NetBeans IDE: Java program can be edited and compiled using NetBeans IDE also.

- Minimum Hardware Configurations:

  Microsoft Windows Vista SP1/Windows 7 Professional:
  Processor: 800MHz Intel Pentium III or equivalent.
  Memory: 512 MB.
  Disk space: 750 MB of free disk space.

To install this software, follow the below given steps:

1. To use NetBeans for Java programming, you need to install Java Development Kit (JDK).
2. Download "NetBeans IDE" installer from link given in the section 3.1.4
3. Run the installer and follow the installation wizard. JRE will be detected while installing the IDE.
4. Click Windows key from your machine keyboard→select all programs→select netbeans from list→click NetBeans application to launch the IDE.
5. NetBeans IDE is provided with GlassFish Application Server.

4.1.9 Installation of Wamp/Xampp Servers: Any one software can be used. This software is used to host web applications. Follow the below given steps to install Wamp Server the software's:

1. Download the WAMP Server from the link given the section 3.1.5 or 3.1.6.
2. Initiate WAMP Server Install Process
3. Select Location/Destination to Install WAMP
4. Select Start Menu Folder to Install WAMP
5. Ready to Install WAMP window will be displayed.
6. To launch the wamp or xampp server click start button from the keyboard→select all programs→select wampserver folder from the program list→click wampserver application to launch the server→click Yes to start the server

4.1.10 Configuration of Eclipse Workspace with Apache Tomcat Server: Eclipse is a very powerful development environment for Java. Mainly for Web Development project you need Web Server. Apache Tomcat is the best production ready web container. By default when you download Eclipse IDE, it doesn't come with Tomcat install with it. Use the following steps to configure the Eclipse IDE:

1. Download Apache Tomcat (any version) from the link given in the section 3.1.8.
2. Extract files to some folder path. **Example**: C:\apache-tomcat-7.0.59
3. Open the Eclipse and do the following settings:

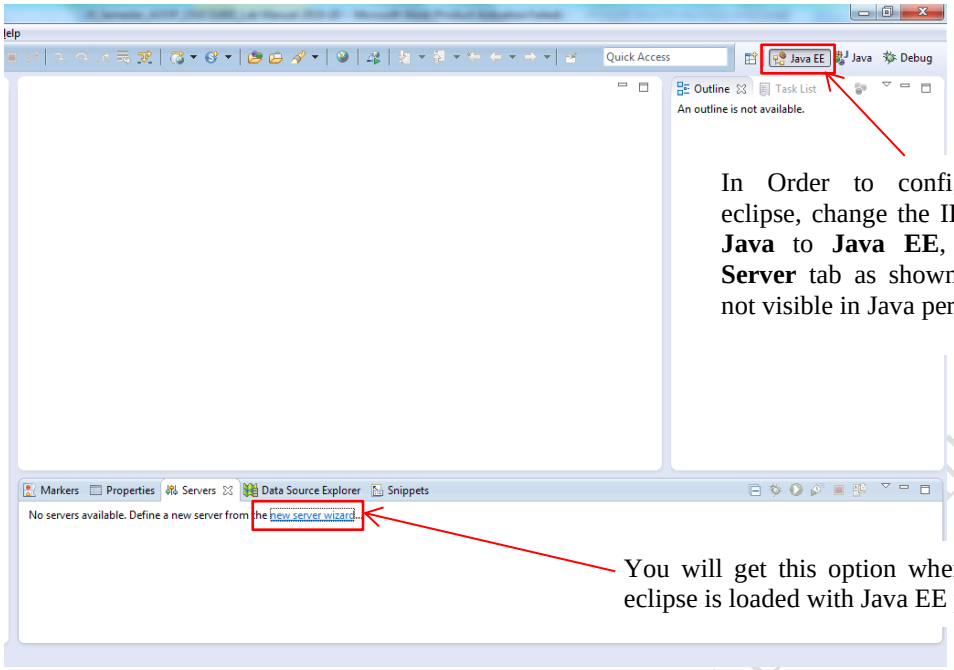


Fig: 3

4. Click no **new server wizard** link from the Server tab and it will prompt pop-up window as show below

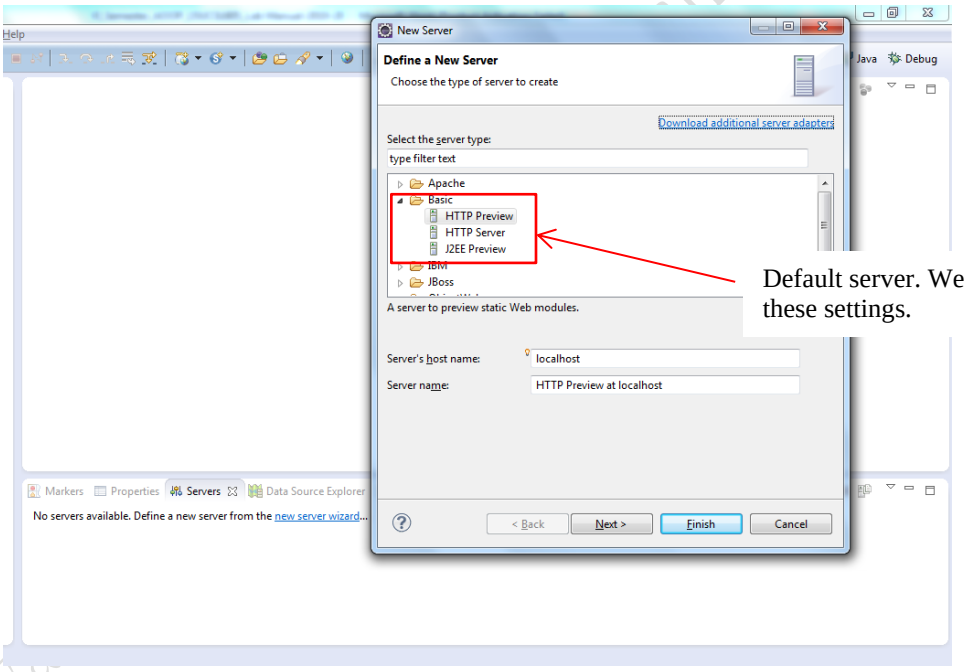
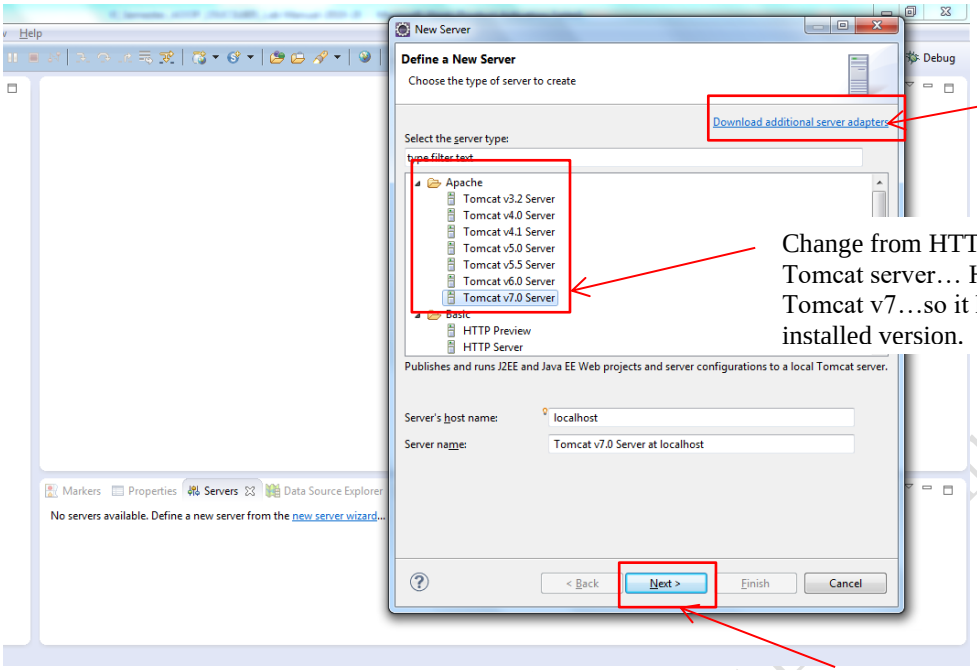


Fig: 4

5. Set the Apache to the Eclipse IDE as shown below.



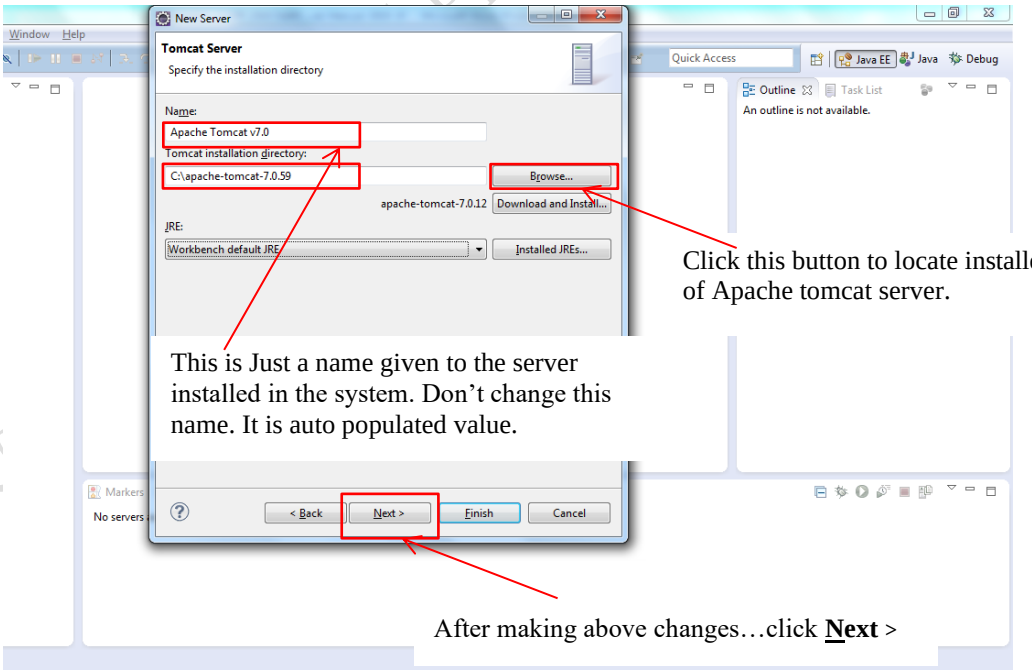
If server is not installed, it can be downloaded from here

Change from HTTP server to Apache Tomcat server… Here System contains Tomcat v7…so it has populated the installed version.

After making above changes…click **Next** >

Fig: 5

6. Locate the Apache tomcat server installed file path as shown below:



Click this button to locate installed version of Apache tomcat server.

This is Just a name given to the server installed in the system. Don't change this name. It is auto populated value.

After making above changes…click **Next** >

Fig: 6

7. If any project is already deployed in the server…below window will list out all the project which are deployed on the server and project to be deployed on the Apache tomcat server. As we are

using first time to develop and deploy applications on server, it will not display any list…it will be empty… if it is empty, just click finish button to finish configurations as shown below

8.

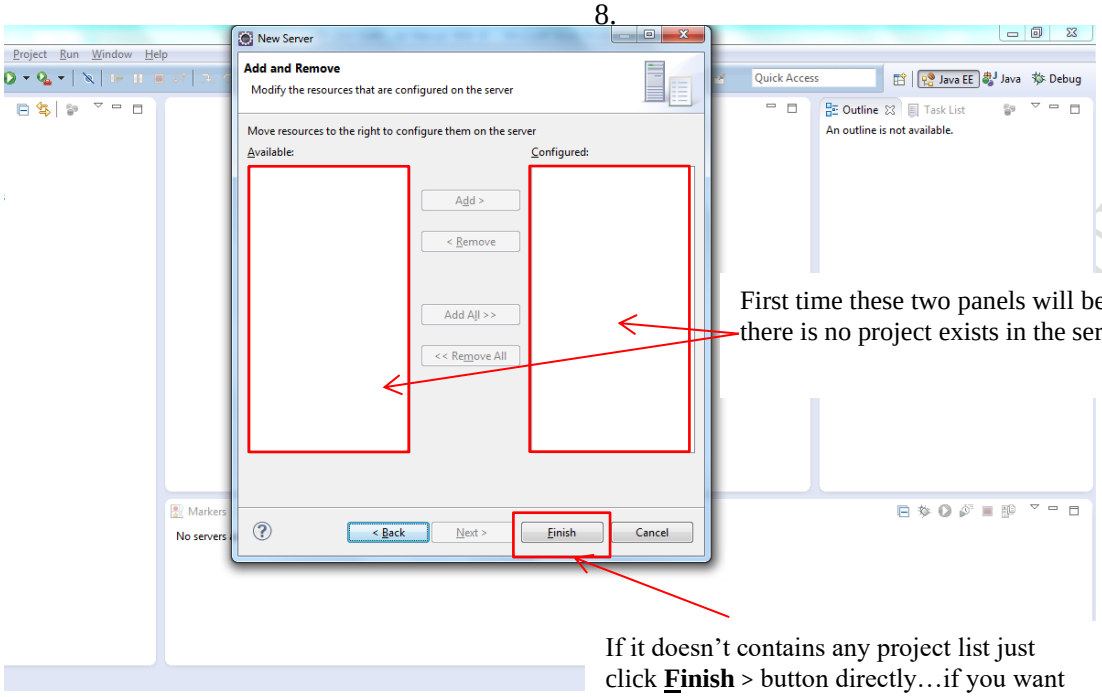


Fig: 7

8. Below is the screen shot which shows Eclipse IDE with Apache tomcat server configured: This screen can be cross verified with the screen shown in the step-3 of this section. That shows difference between unconfigured and configured version of Eclipse

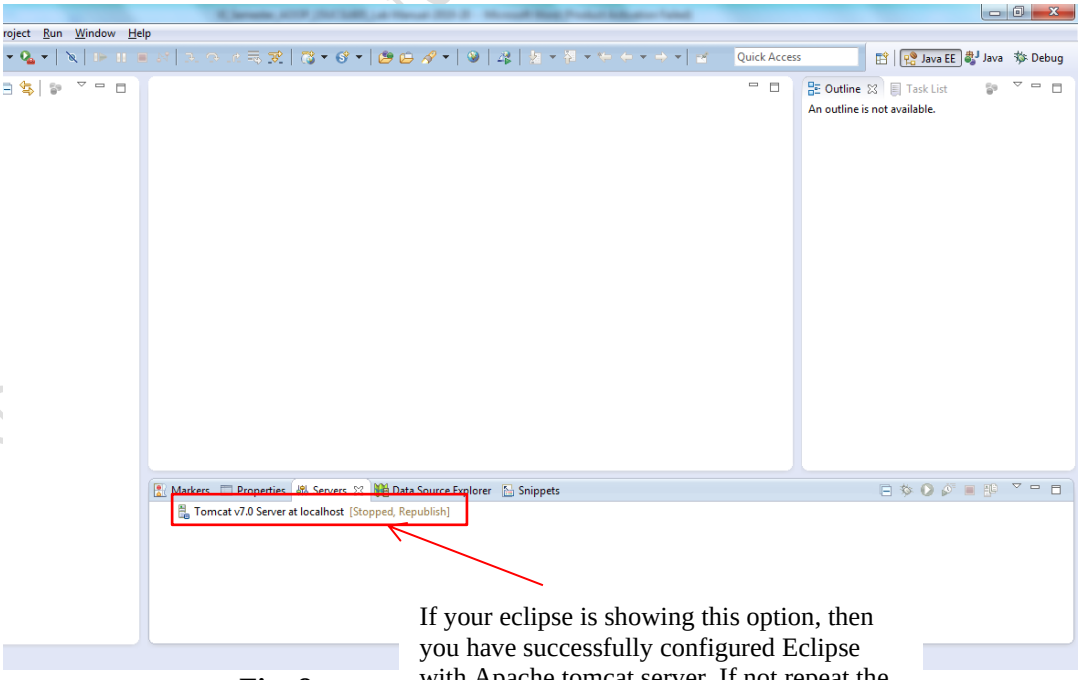


Fig: 8

9. Now it is time to test the server, to do so, select the Tomcat v7.0 server at localhost, right click on it→click start option as show below:

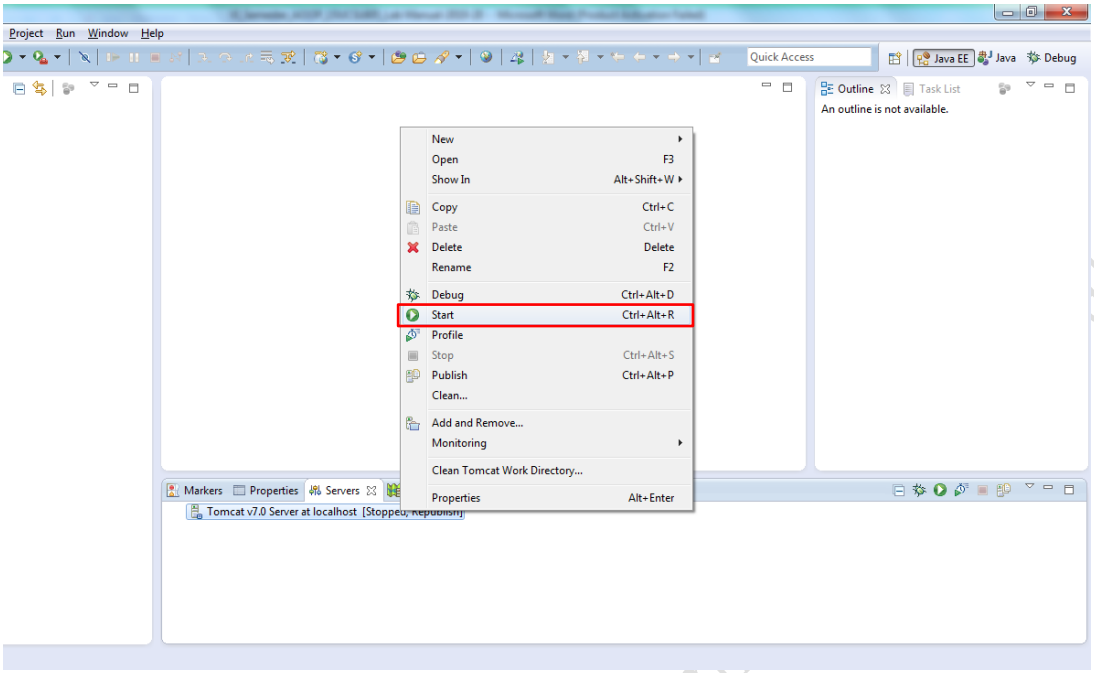


Fig: 9

10. Server is starting and it will take some time to run the server as shown below:

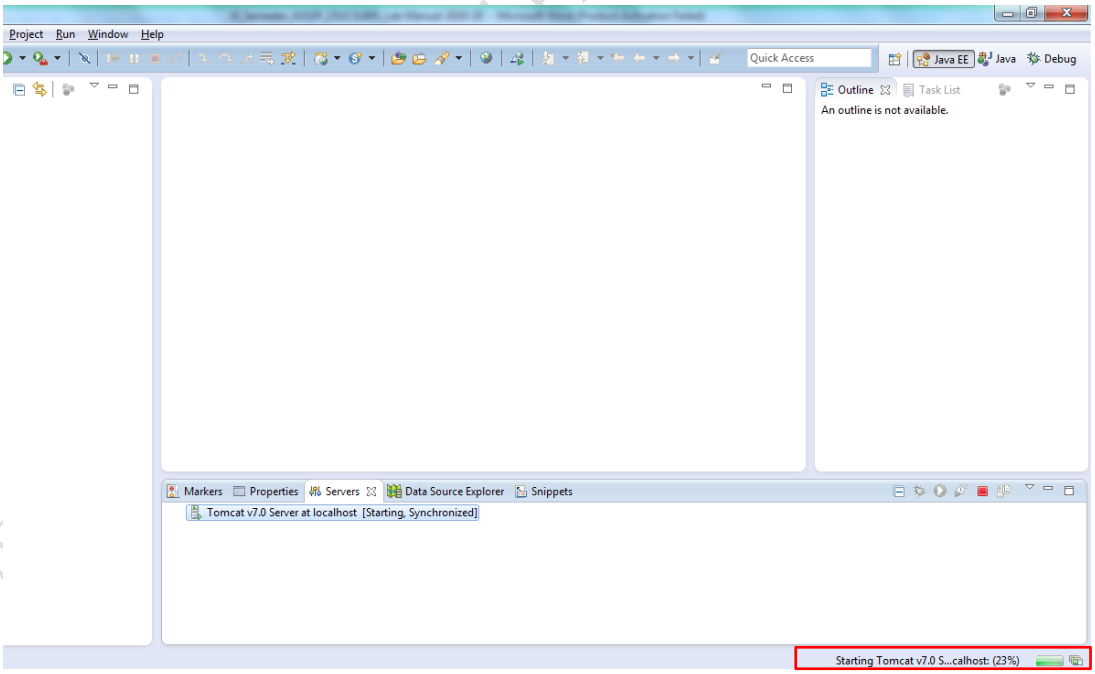


Fig: 10

11. Now server is up and ready to deploy the web applications.

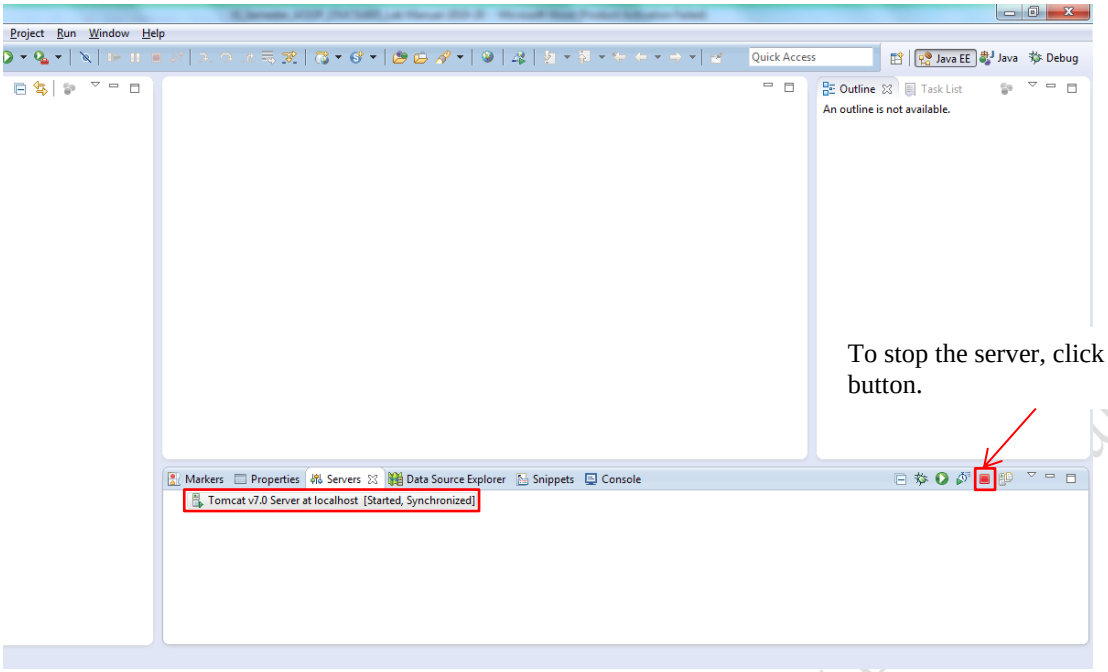


Fig: 11

4.1.11 Configuration of NetBeans IDE Workspace with Apache Tomcat Server: Glassfish is provided with **NetBeans IDE** as a reference implementation of Java Enterprise Edition (EE). It is not intended for use in production environments. As NetBeans provides GlassFish server, no need to add any external application servers.

5. How to build Java applications (Java SE) using Eclipse IDE

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License. It is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.

5.1 Steps to create simple Java application using Eclipse IDE:

Problem Statement: Write a Java program to implement simple calculator using Eclipse IDE.

Step-1: To write a Java program, launch the installed Eclipse IDE. Following method is one of the ways to create Java application. There are other ways to create project. Example: Creating .java file in a particular package can be done by File→New Project→Package option, but in the below given method both creation of .java and package is done in single window.

- Double click on shortcut icon of eclipse IDE 

from your desktop.
- It will open Eclipse:

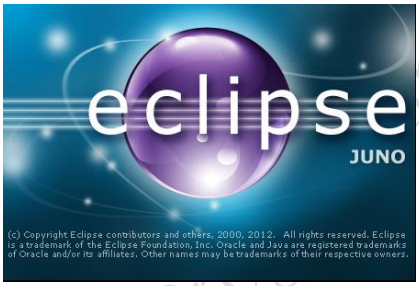
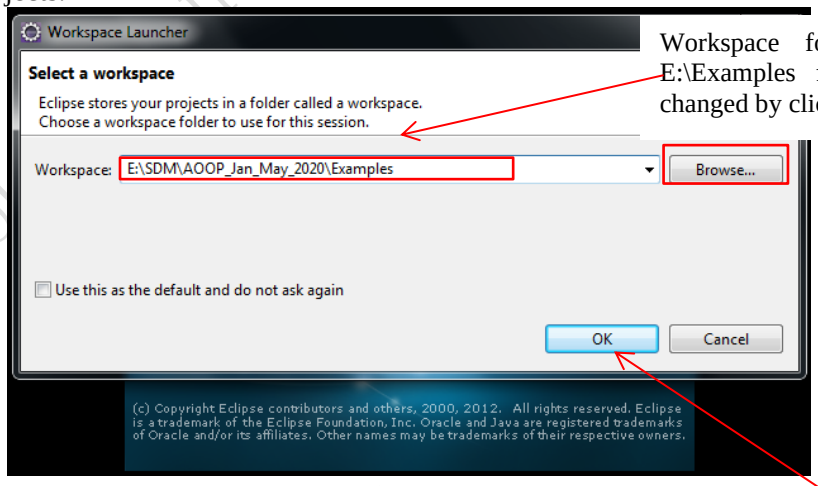


Fig: 12

- It will launch the Eclipse IDE. Before opening IDE, it will prompt workspace folder window. It can be changed or default Workspace folder path can be used to create projects.



Workspace folder path is changed to E:\Examples folder. Folder path can be changed by clicking **Browse** button

Fig: 13

Click **OK** to load IDE workbench

- Following window will Open:
  Here note the following marked fields.

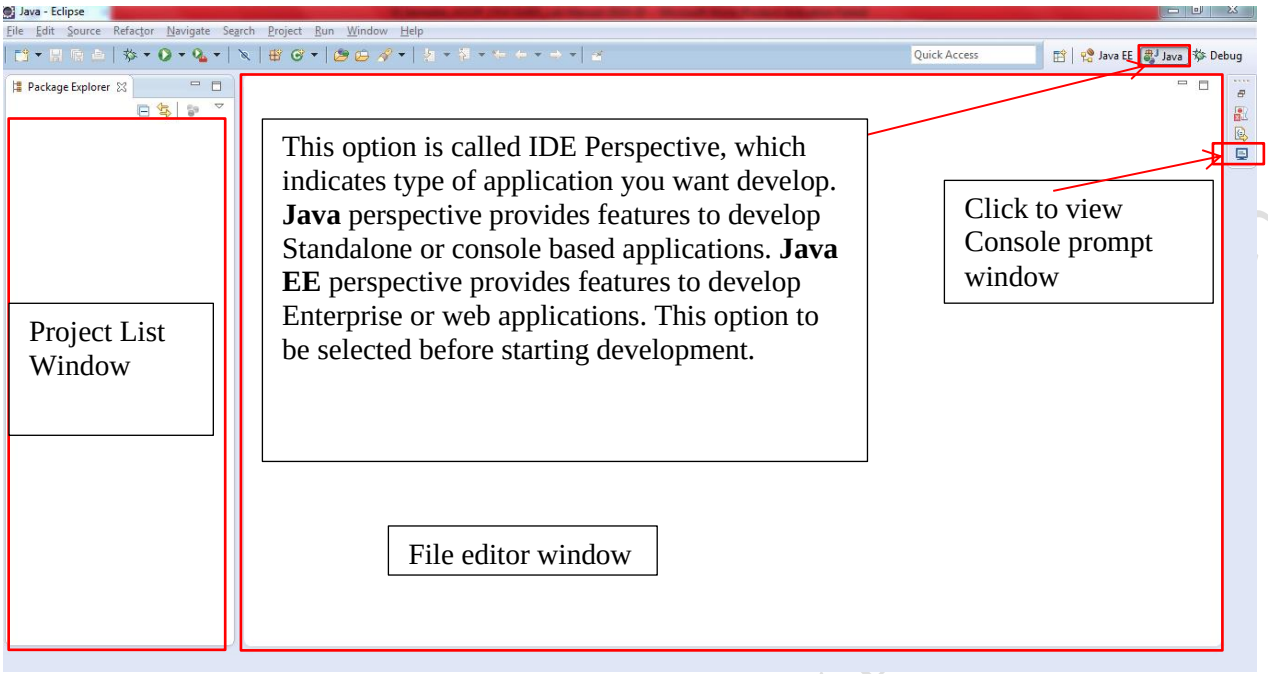


Fig: 14

- Eclipse IDE with console prompt window

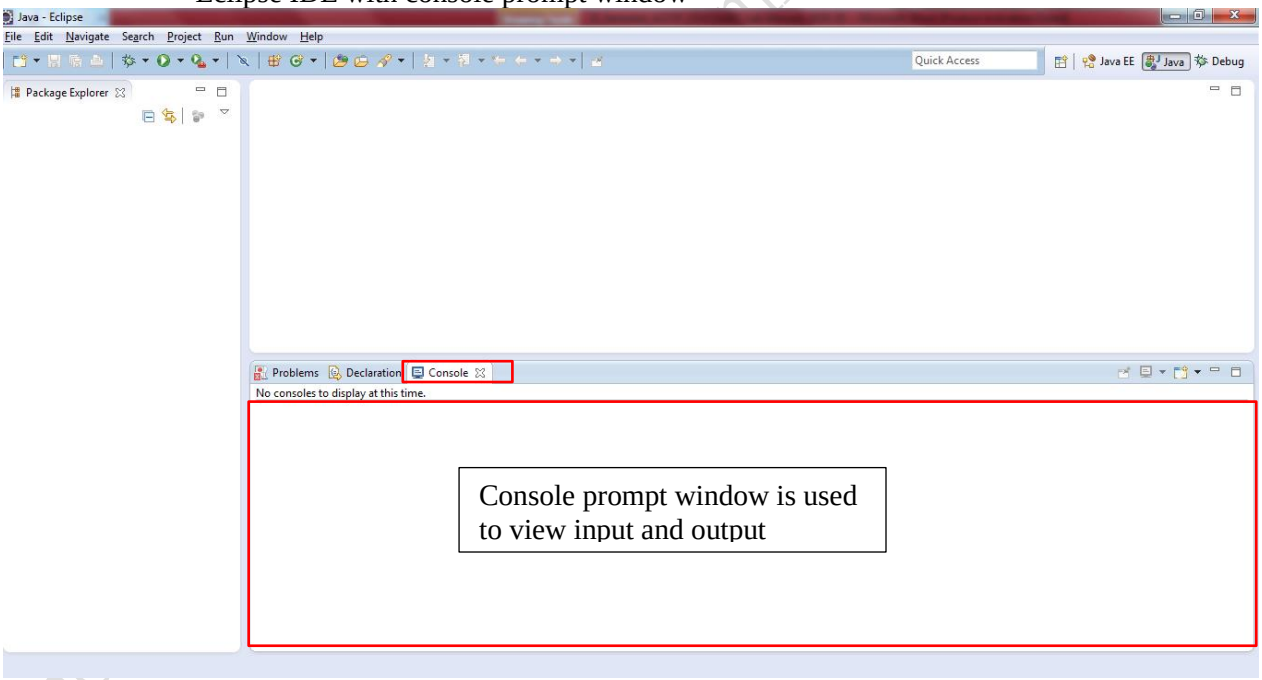


Fig: 15

- Create new Project as shown below:

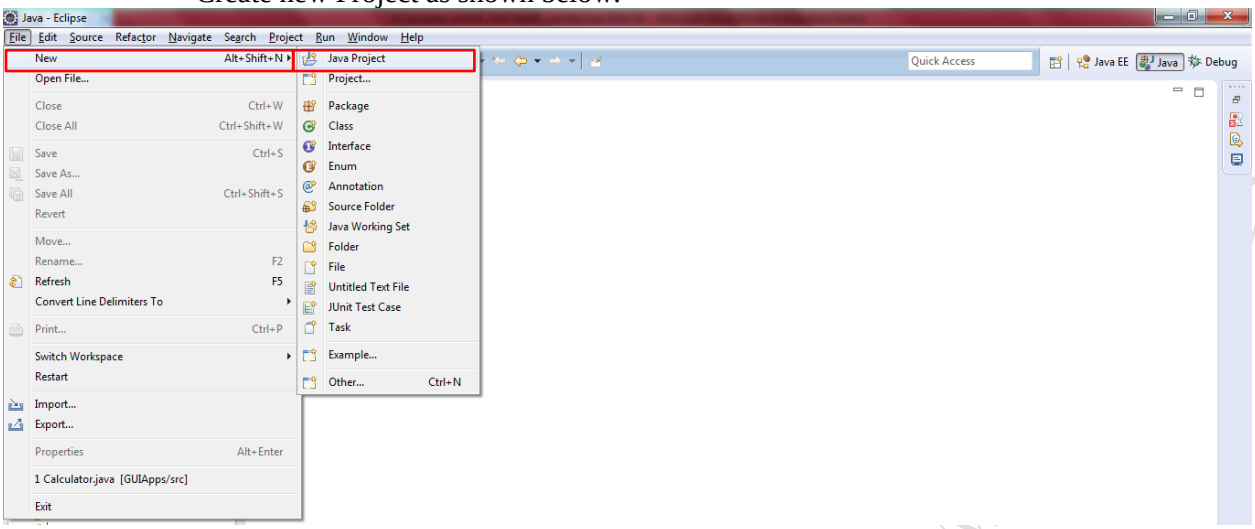


Fig: 16

- This window comes with **project name**, **default location of project**, **jre version** etc many options. Here only project name is entered as given below other default settings are not changed:

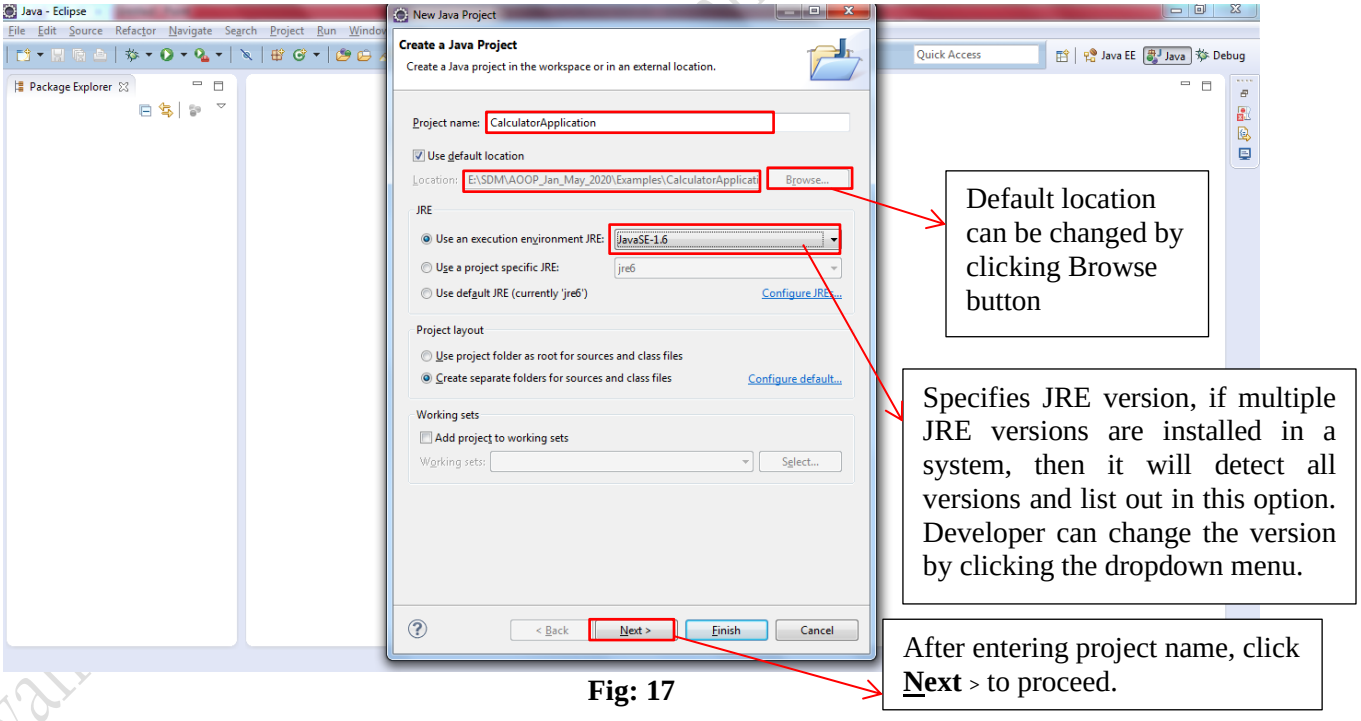


Fig: 17

- Another window with JRE libraries, build paths etc. This window will display project folder with required JRE libraries and other tabs.

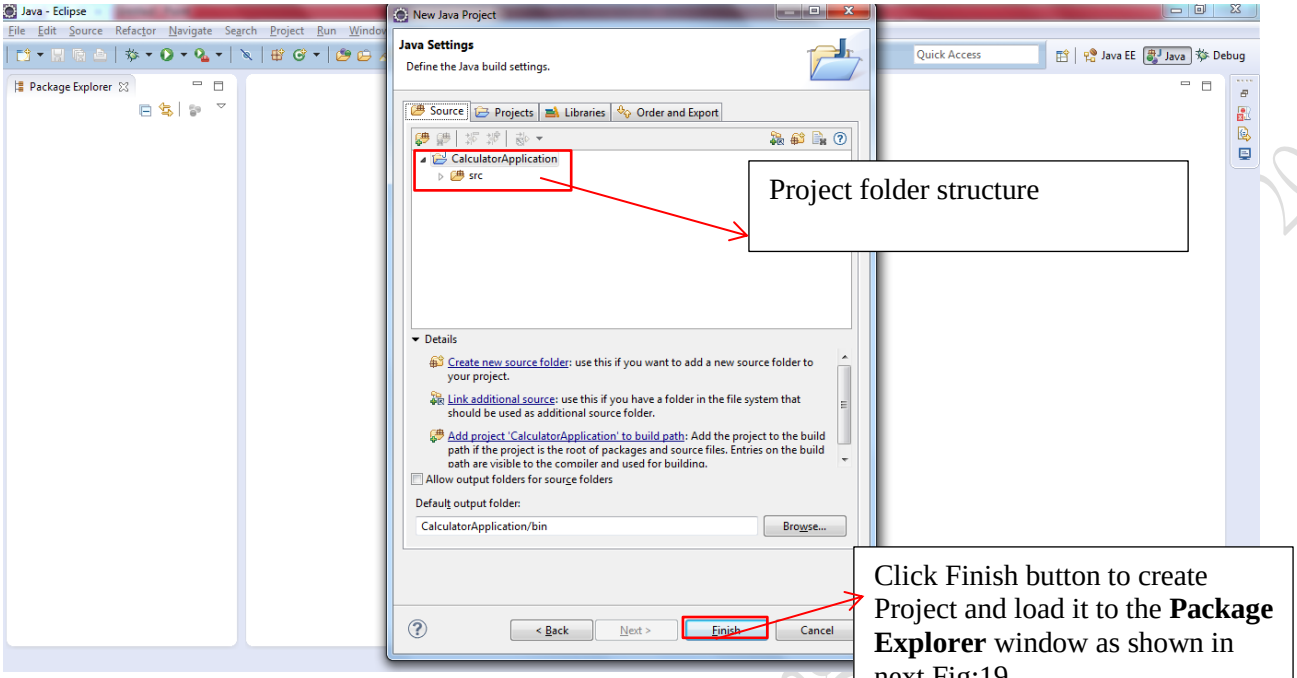


Fig: 18

- IDE is has created project and related files like JRE libraries, folders used to add multiple .java files.

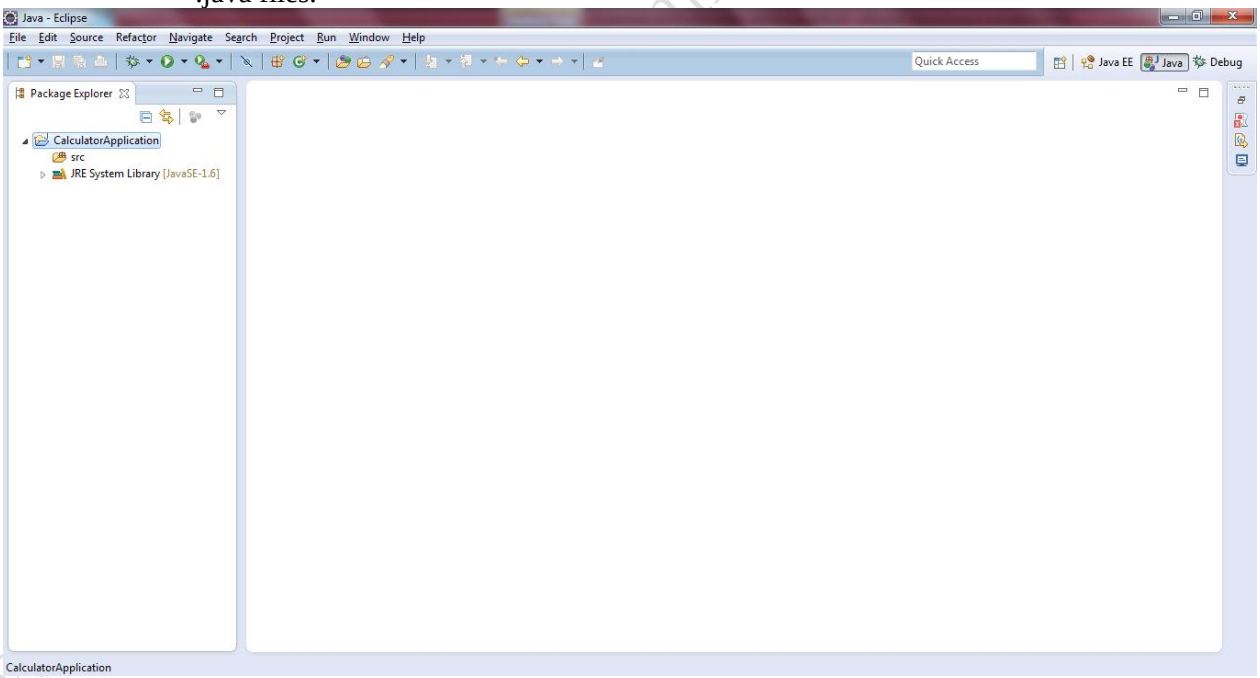


Fig: 19

- Adding class file i.e. .java file into project folder (under /src folder)

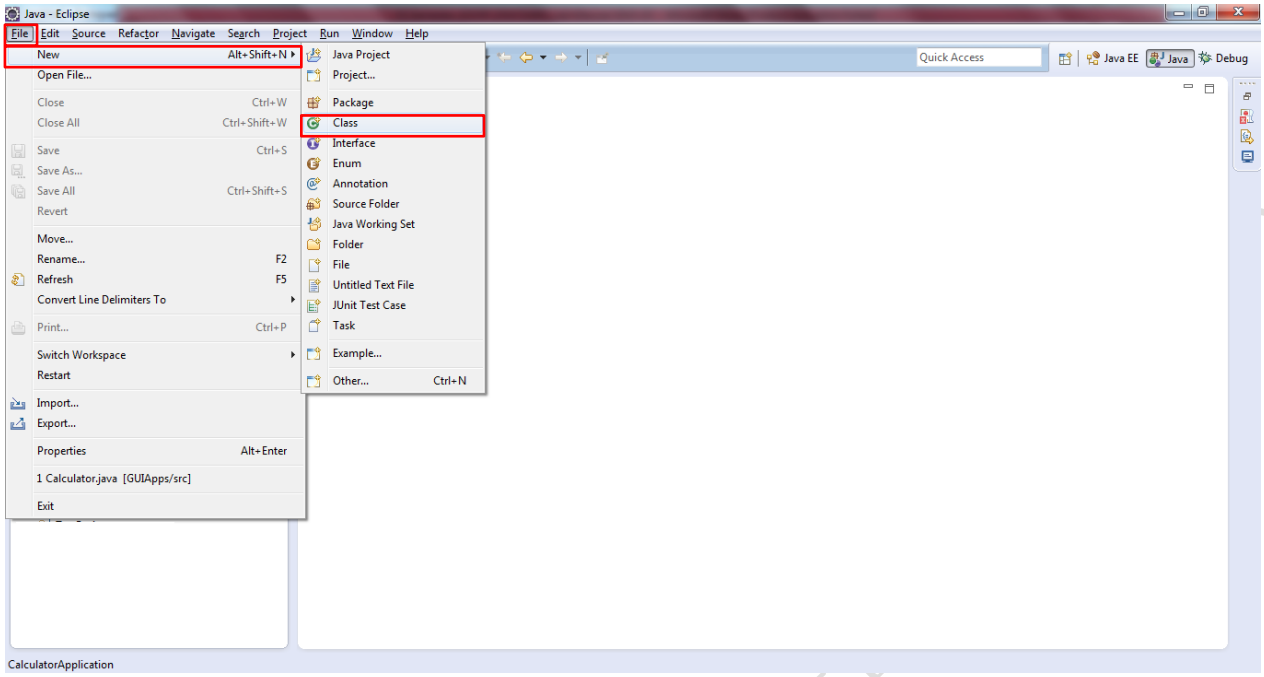


Fig: 20

- Enter Project name, package name as shown below. Check other settings like Modifiers, creating main method etc..

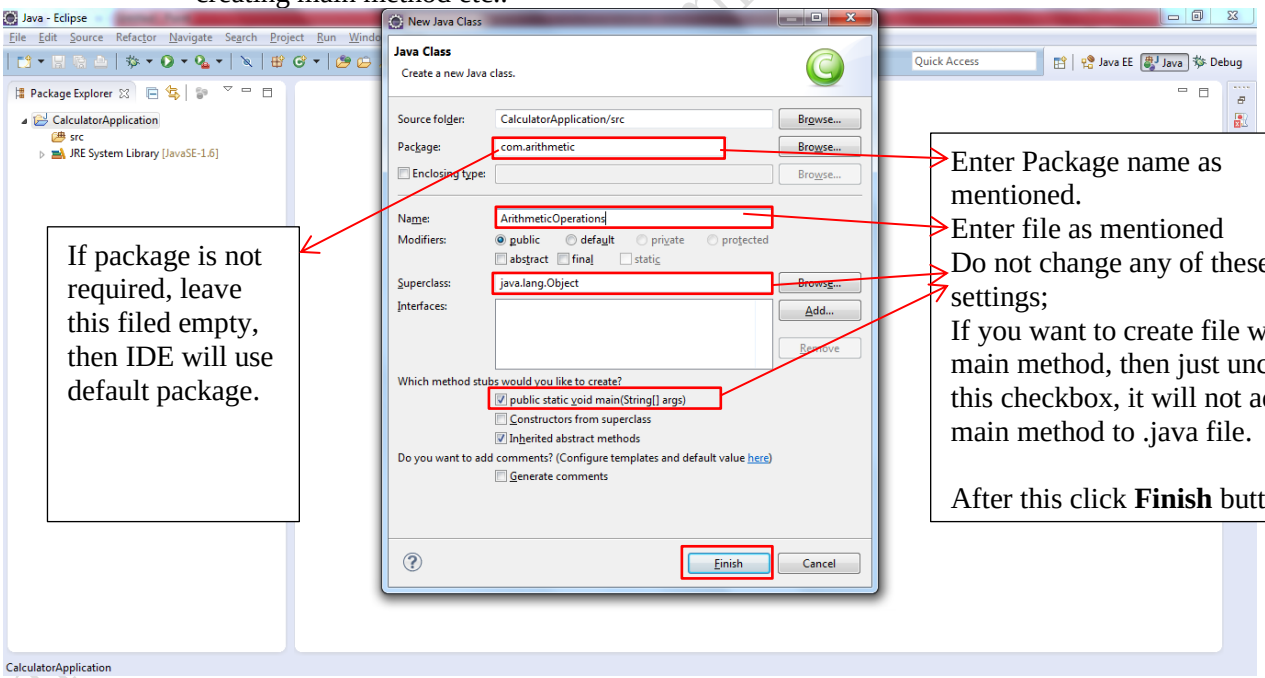


Fig: 21

- File with main class and method.

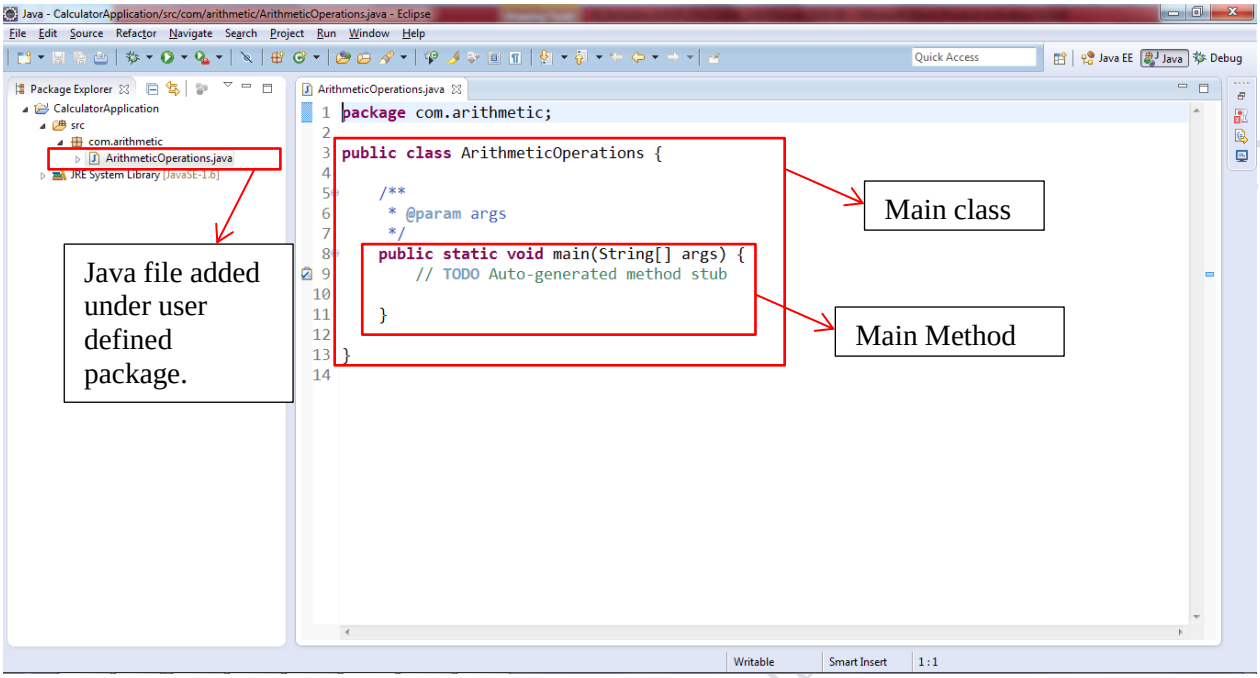


Fig: 22

- Edit the auto generated main methods i.e. add required logic into main method and save the file.

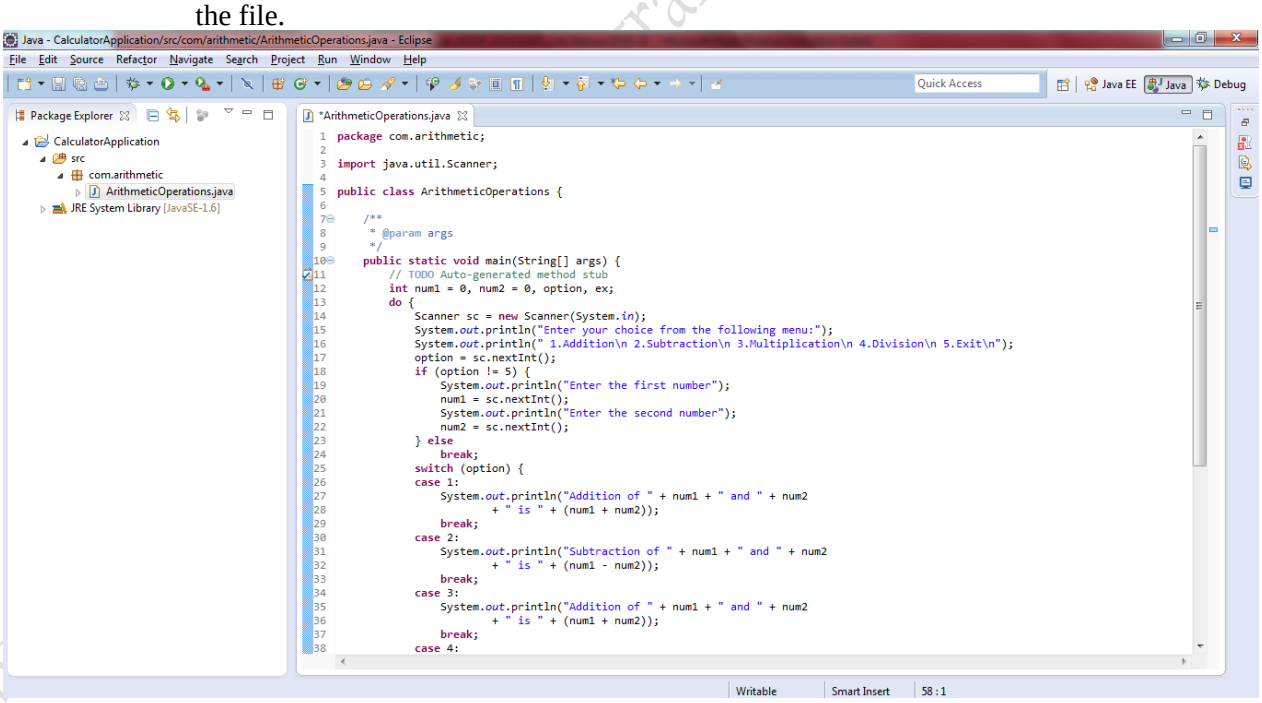


Fig: 23

- Now it's time to compile and execute the program to test the result:

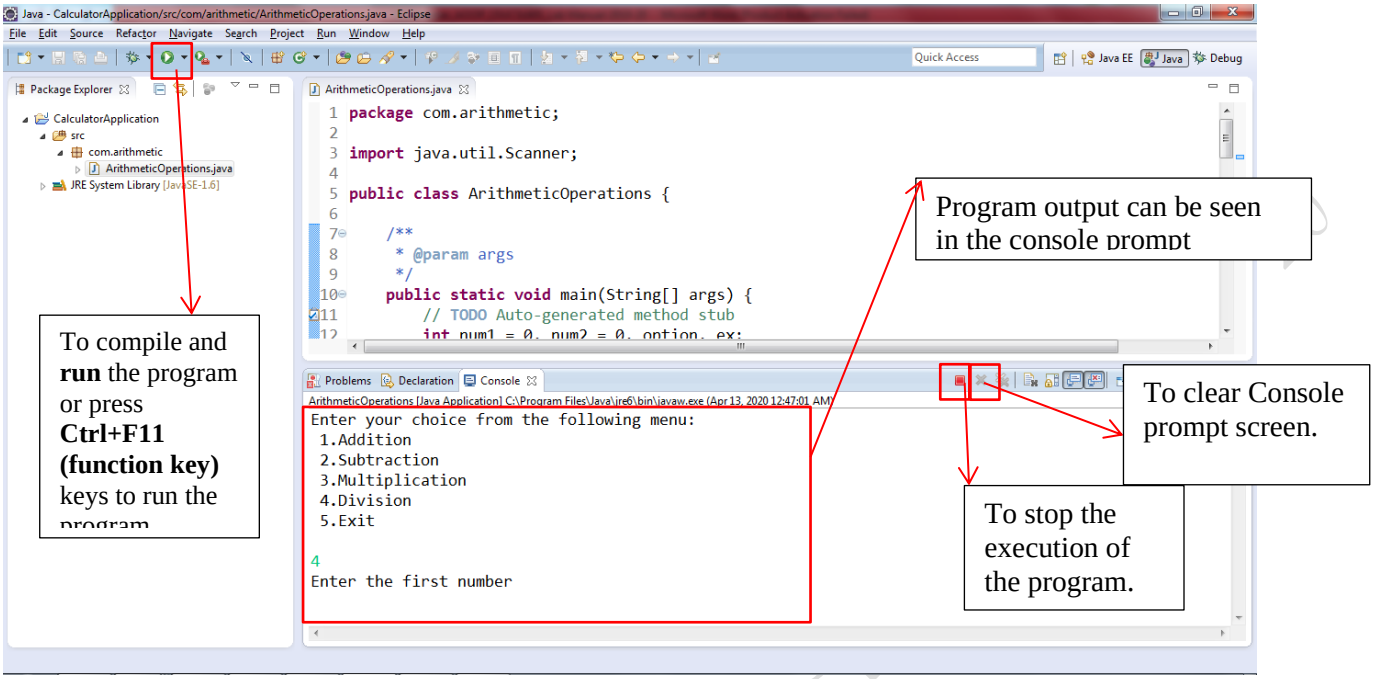
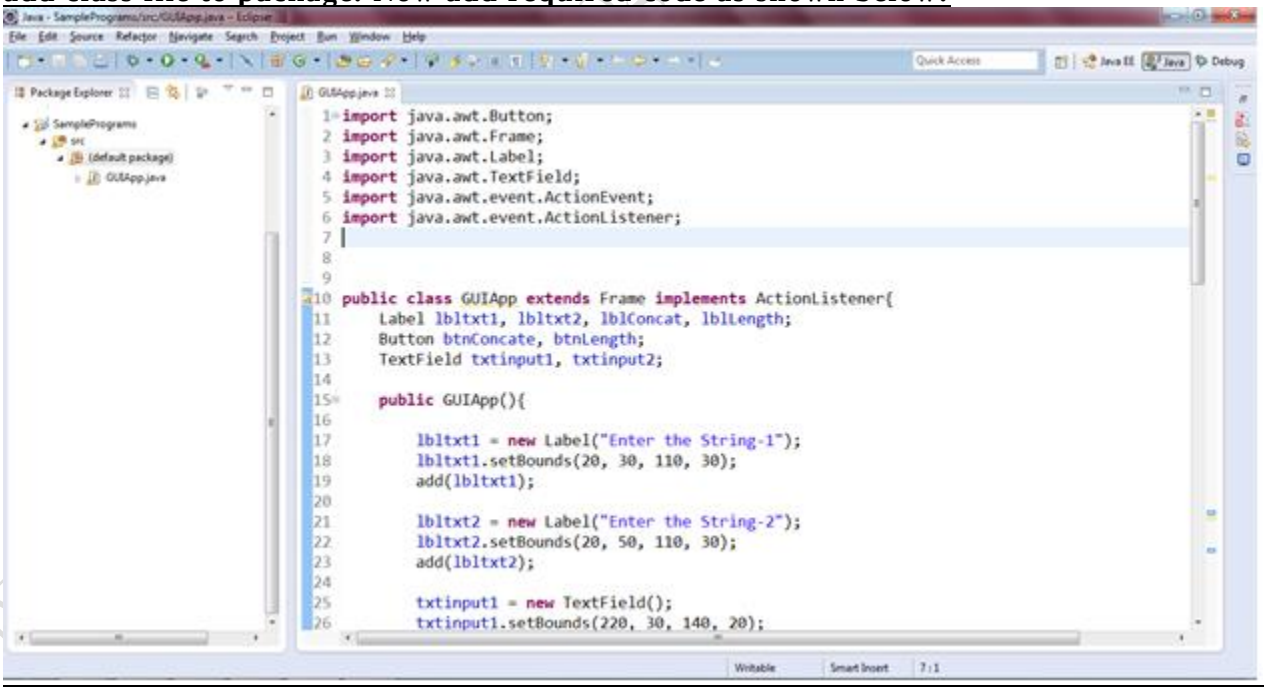


Fig: 24

5.2 Creating GUI application using Eclipse IDE: **To create simple AWT based GUI application follow the approaches used in** Fig: 12 to Fig: 22 **to create Java project and add class file to package. Now add required code as shown below:**



```
import java.awt.Button;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;


public class GUIApp extends Frame implements ActionListener{
    Label lbltxt1, lbltxt2, lblConcat, lblLength;
    Button btnConcate, btnLength;
    TextField txtinput1, txtinput2;

    public GUIApp(){

        lbltxt1 = new Label("Enter the String-1");
        lbltxt1.setBounds(20, 30, 110, 30);
        add(lbltxt1);

        lbltxt2 = new Label("Enter the String-2");
        lbltxt2.setBounds(20, 50, 110, 30);
        add(lbltxt2);

        txtinput1 = new TextField();
        txtinput1.setBounds(220, 30, 140, 20);
```

Fig: 25

- **Now run the GUI application as show below:**

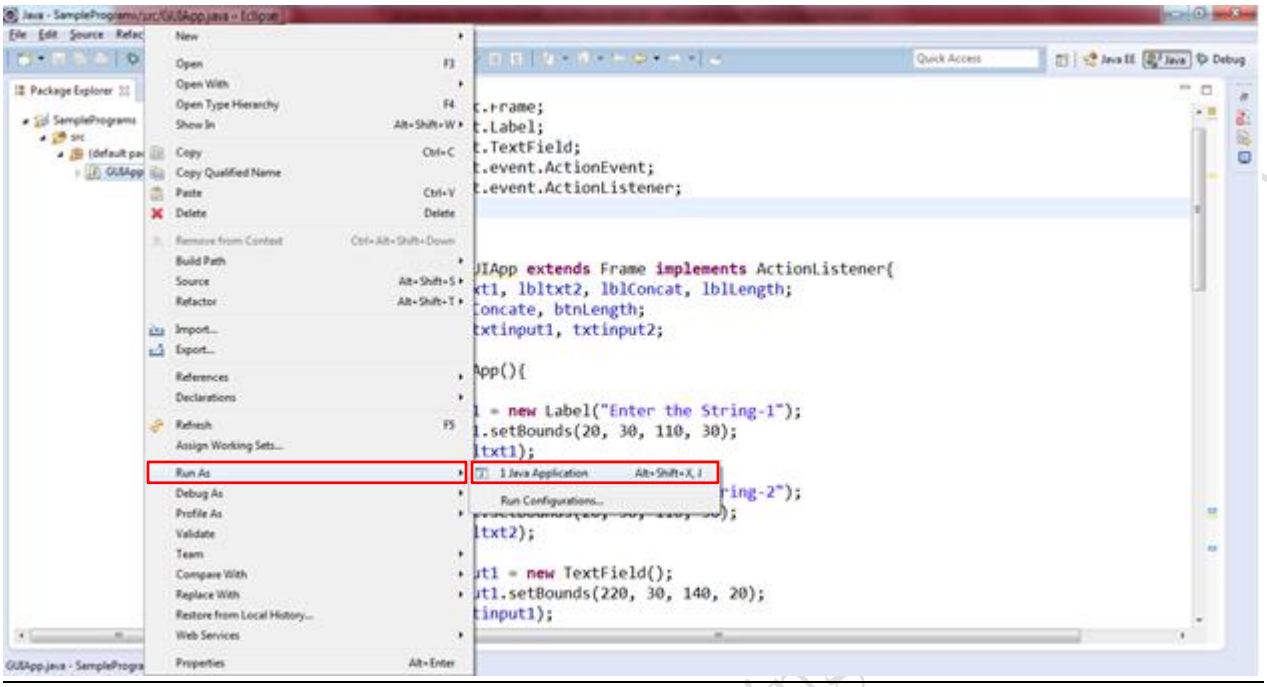


Fig: 26

Output of the above program:

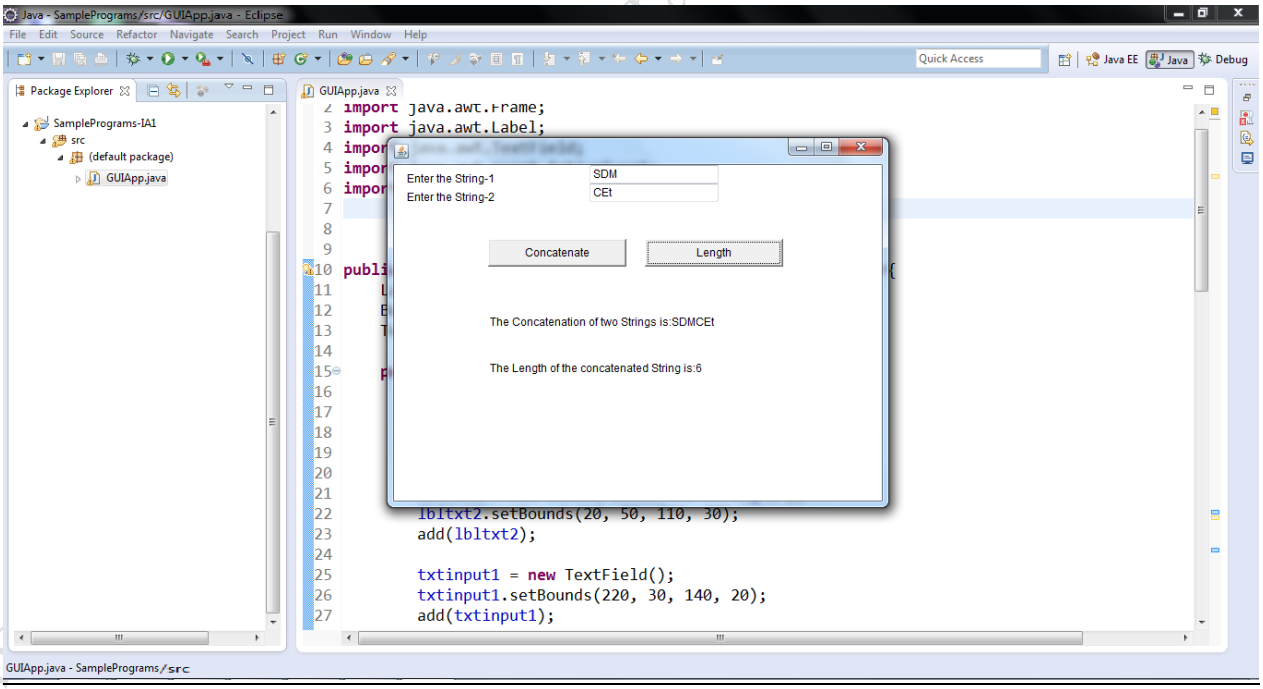
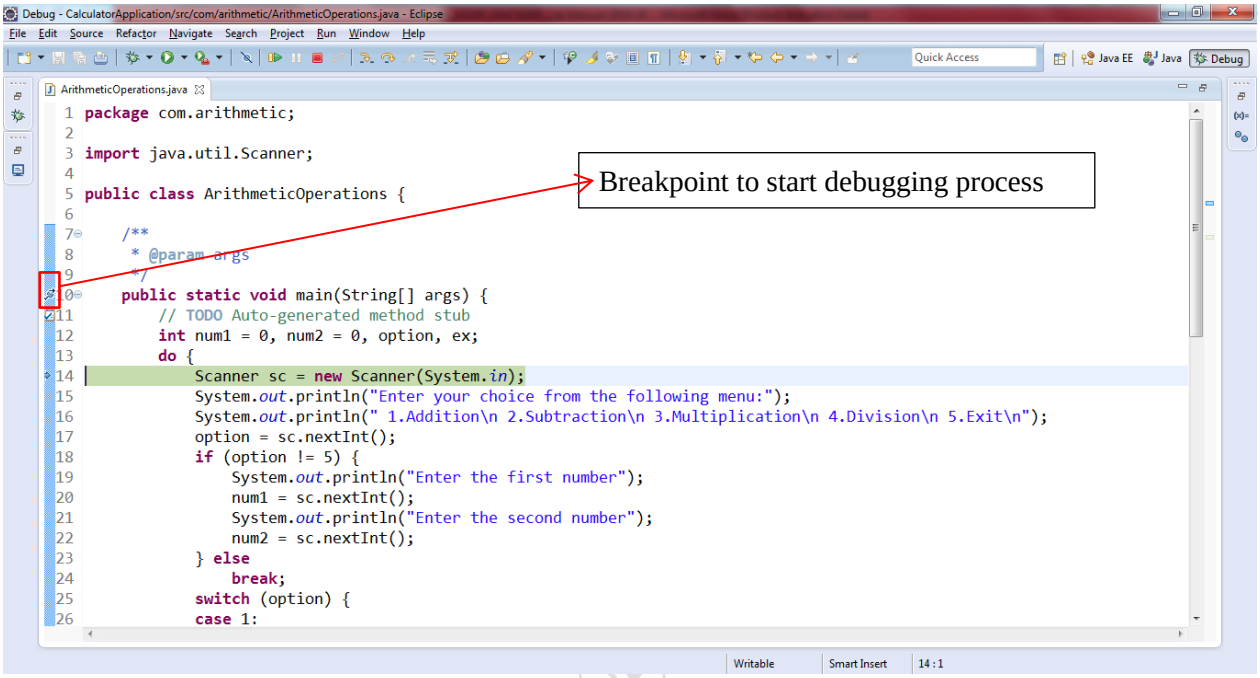


Fig: 27

5.3 How to debug the program in Eclipse IDE:

- Program can be debugged line by line by **putting a breakpoint against the method** (i.e. from which method or line you want to debug the code) by **double clicking on the line number** as highlighted below. Then press **F11 key** to **start debugging**. To continue debugging **line by line** press **F6 key.** Check in the Run menu of eclipse IDE.




```
1  package com.arithmetic;
2
3  import java.util.Scanner;
4
5  public class ArithmeticOperations {
6
7      /**
8       * @param args
9       */
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12         int num1 = 0, num2 = 0, option, ex;
13         do {
14             Scanner sc = new Scanner(System.in);
15             System.out.println("Enter your choice from the following menu:");
16             System.out.println(" 1.Addition\n 2.Subtraction\n 3.Multiplication\n 4.Division\n 5.Exit\n");
17             option = sc.nextInt();
18             if (option != 5) {
19                 System.out.println("Enter the first number");
20                 num1 = sc.nextInt();
21                 System.out.println("Enter the second number");
22                 num2 = sc.nextInt();
23             } else
24                 break;
25             switch (option) {
26             case 1:
```

Fig: 25

6. **Developing Java applications (Java SE) using NetBeans IDE**

6.1 Java program to build Simple calculator using NetBeans IDE

Problem Statement: Write a Java program to implement simple calculator using NetBeans IDE.

- **Open the NetBeans IDE: Go to start→all programs→NetBeans folder→NetBeans application** or **double click** 

shortcut icon placed on the desktop.

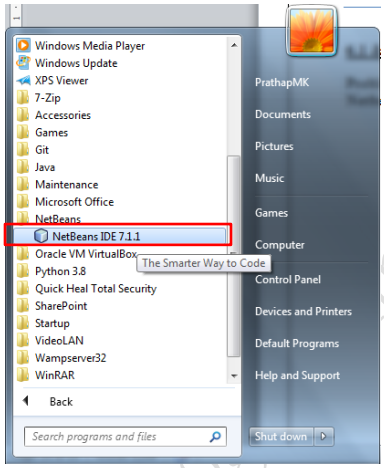


Fig: 26

- It will launch the NetBeans IDE as shown below:

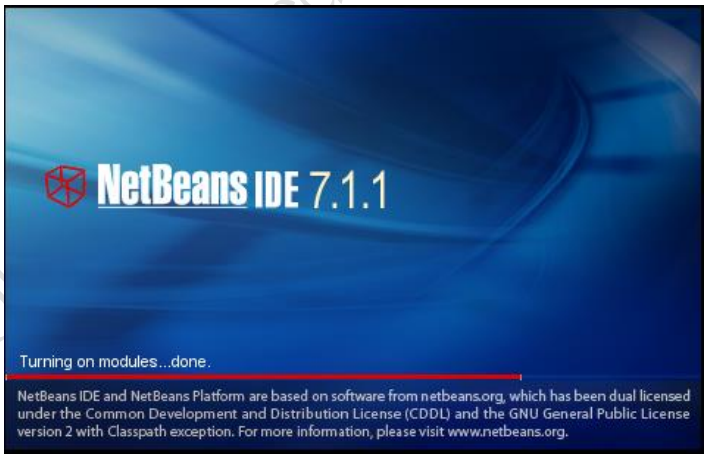


Fig: 27

- Next It will load the IDE framework all required windows as shown below:

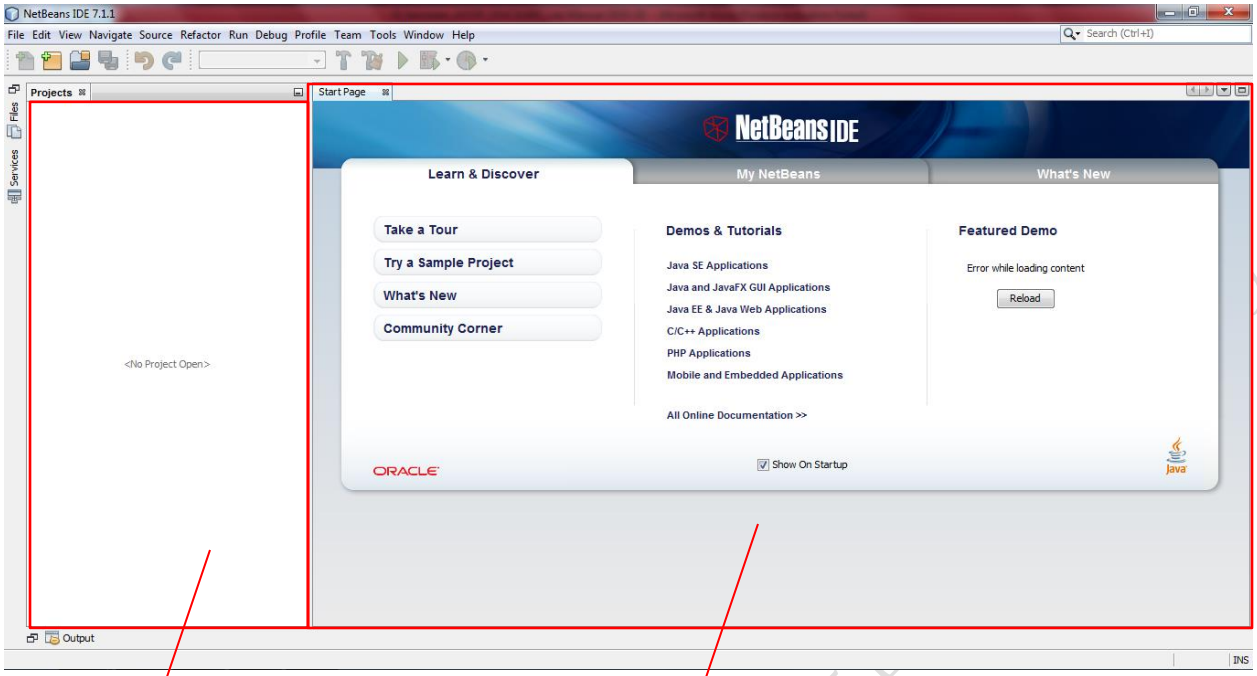


Fig: 28

Project List window

Start page of NetBeansIDE

- Create a new project as shown below:

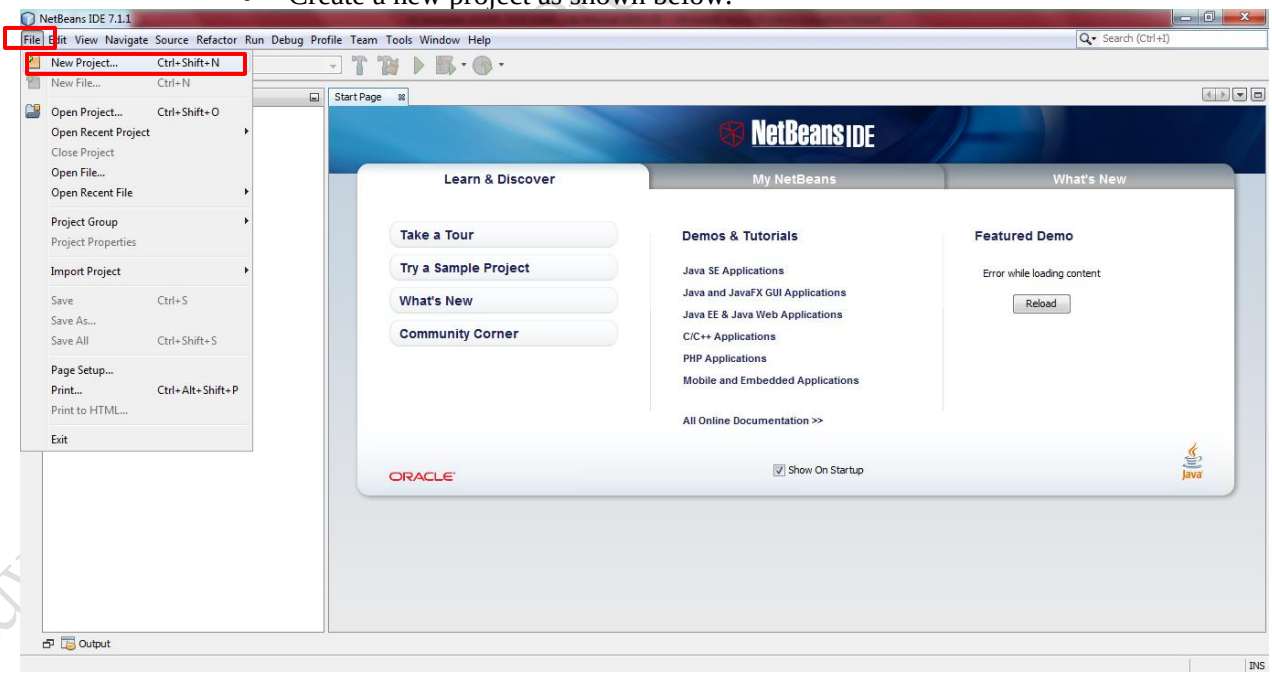


Fig: 29

- Select the type of the project as shown below:

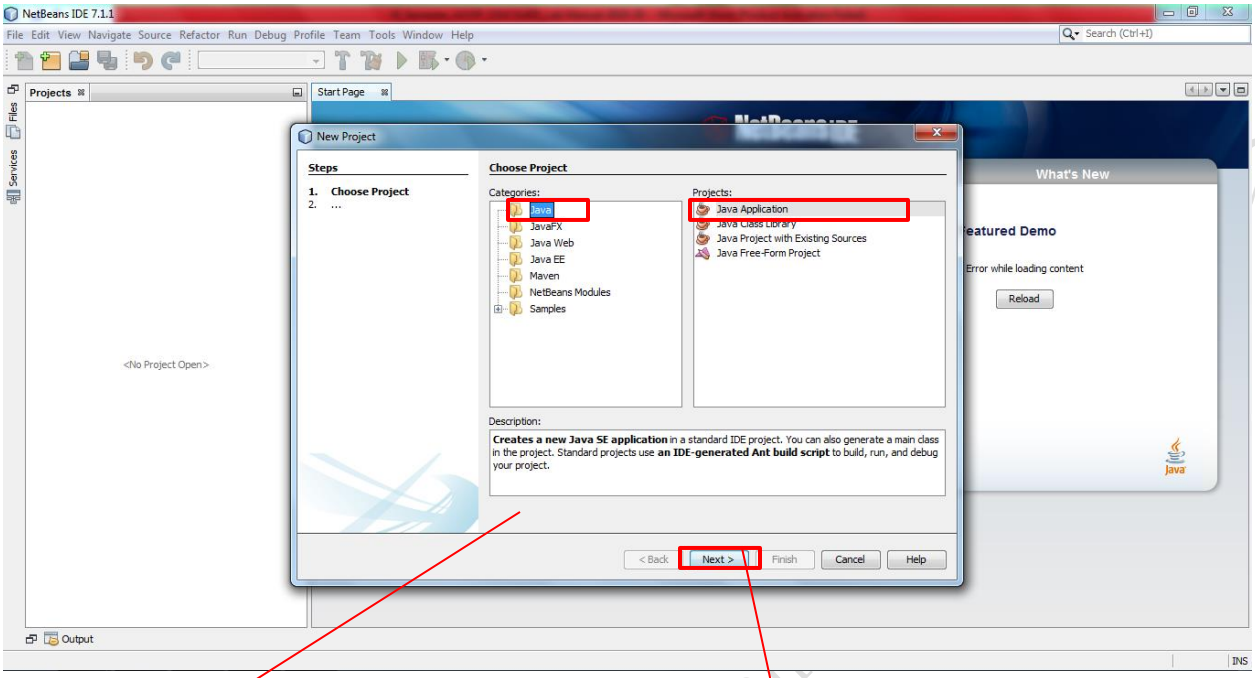


Fig: 30

This New Project window list out different types supported by Java and IDE

Select type of project you want to develop from Categories panel and related file type from Projects panel window and click Next button to continue.

- Now enter the project name and

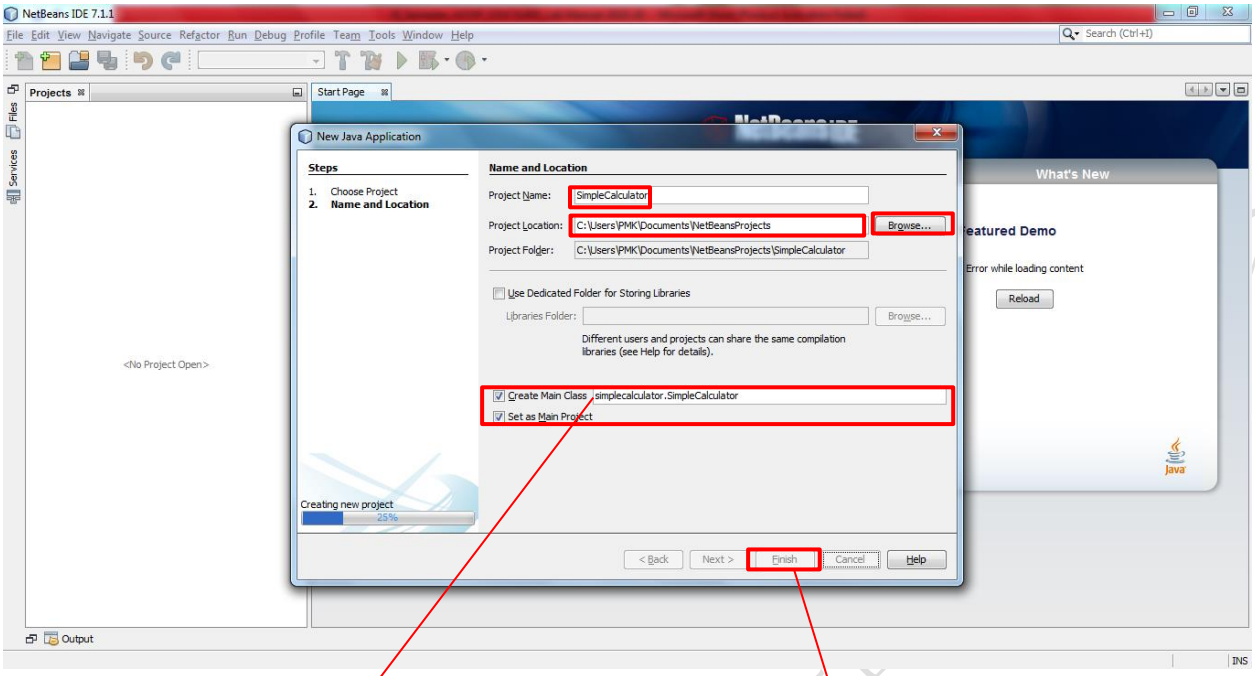


Fig: 31

Do not change these settings.

Enter the project name and select the absolute path of project by clicking the browse button. Don't change other settings. Click **Finish** button.

- IDE is loaded with Java SE project folder structure and main class as shown below.

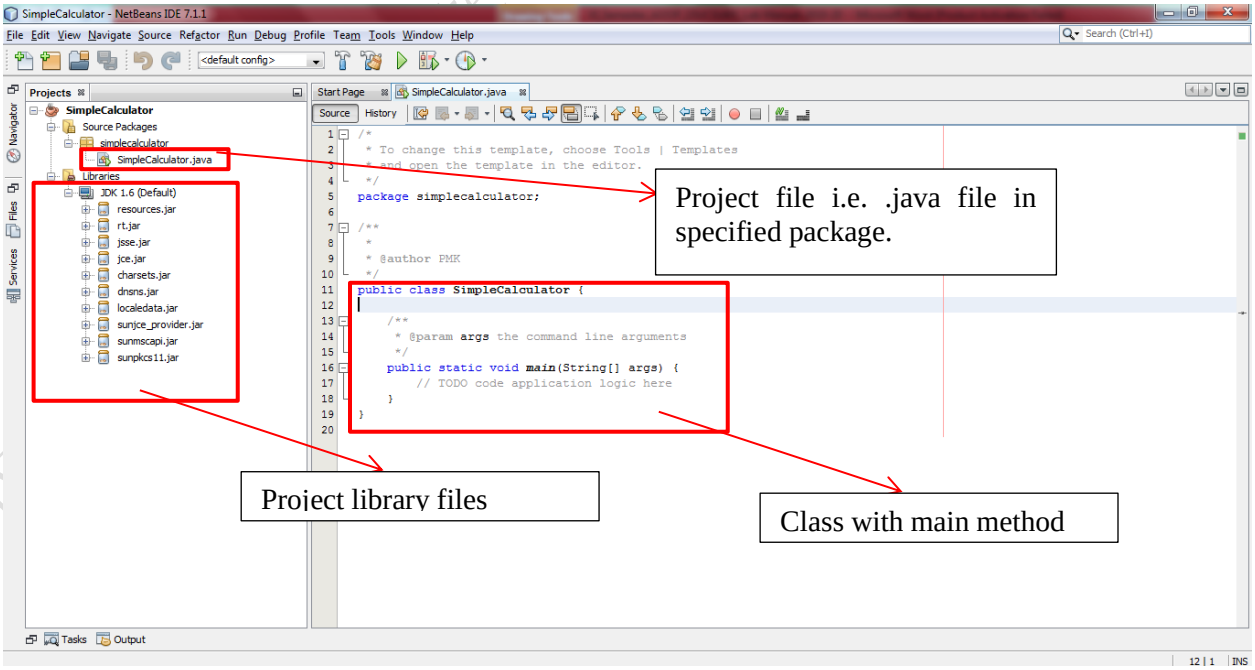


Fig: 32

- Write a required Java program to create a simple calculator as shown below:

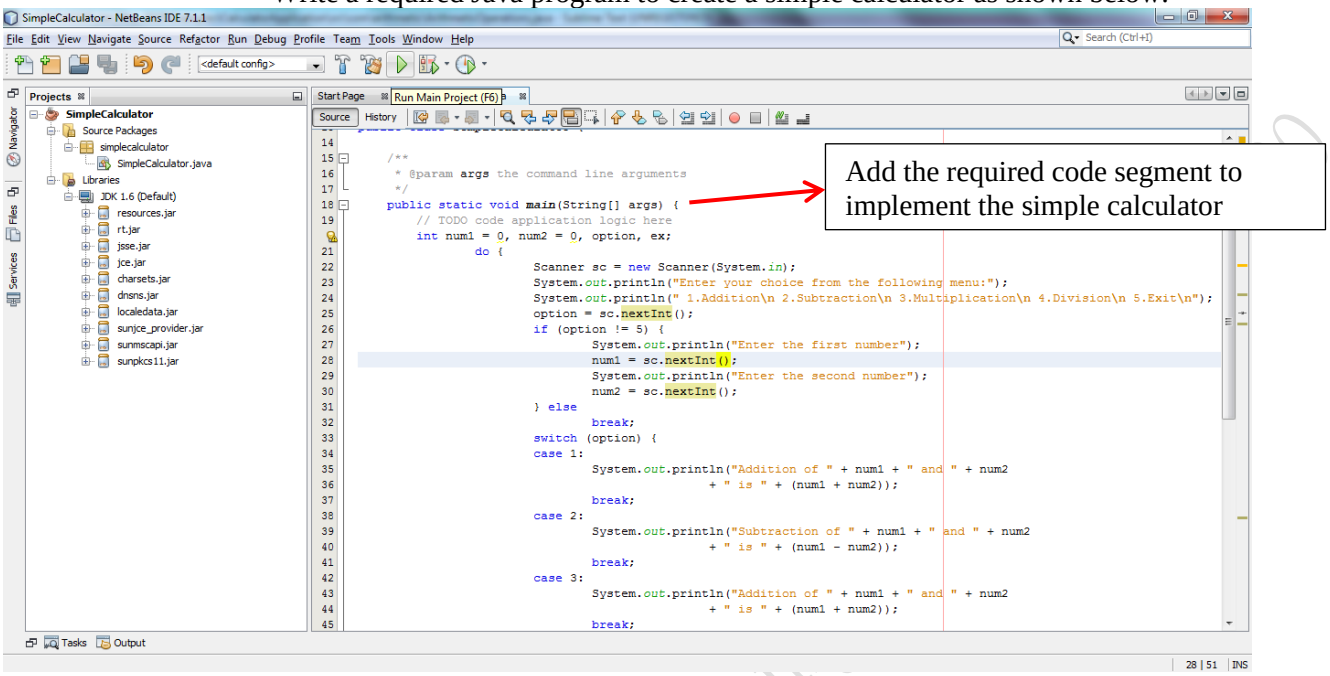


Fig: 33

- Now it's time to run the main project to test the program output as shown below:

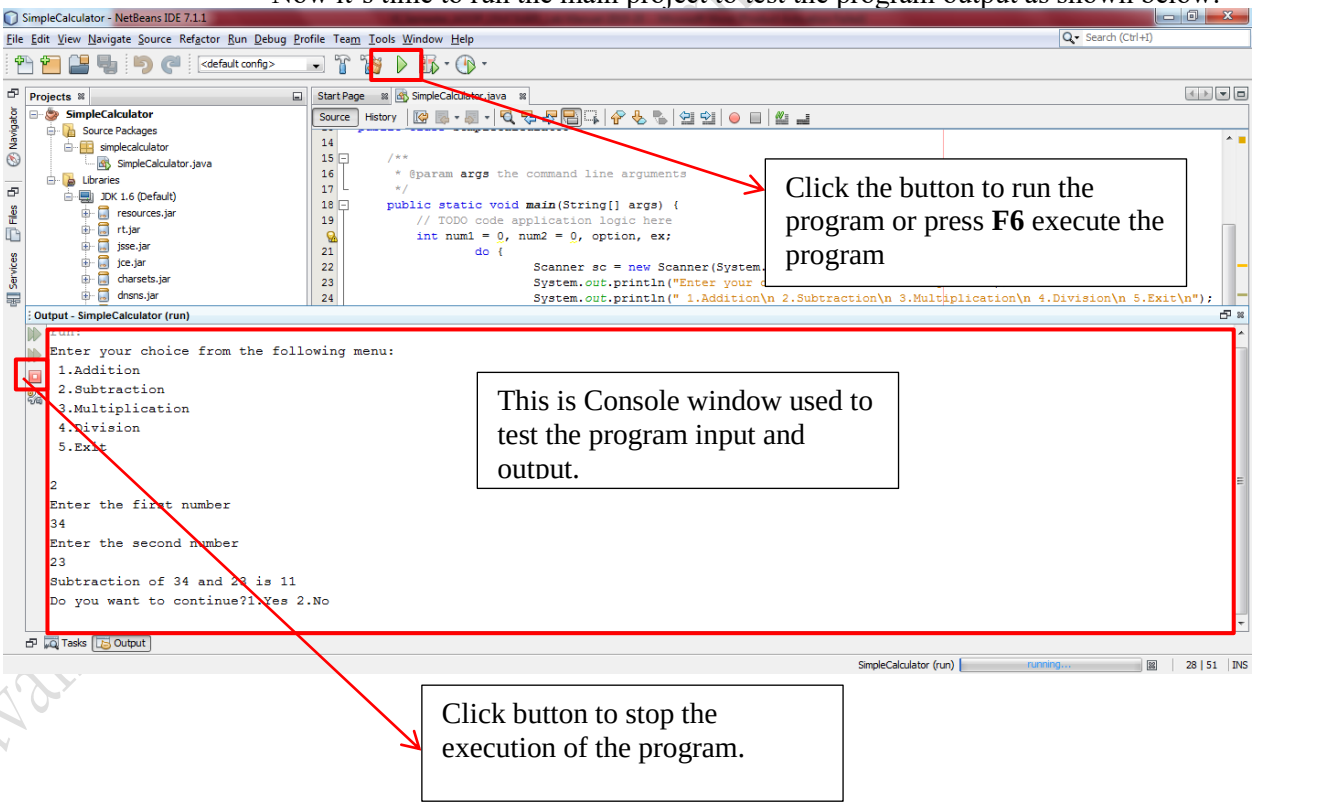


Fig: 34

6.2 Develop GUI application using NetBeans IDE.

Problem statement-**Java program to design GUI application to find the following operations:**

i. Concatenate two strings.
ii. Find the Length of the concatenated string.

Above example is implemented using Swings:

- **Follow the Fig(s) from:** Fig 26 to 31 **to create new project.**

- **Designing GUI with all required components:**

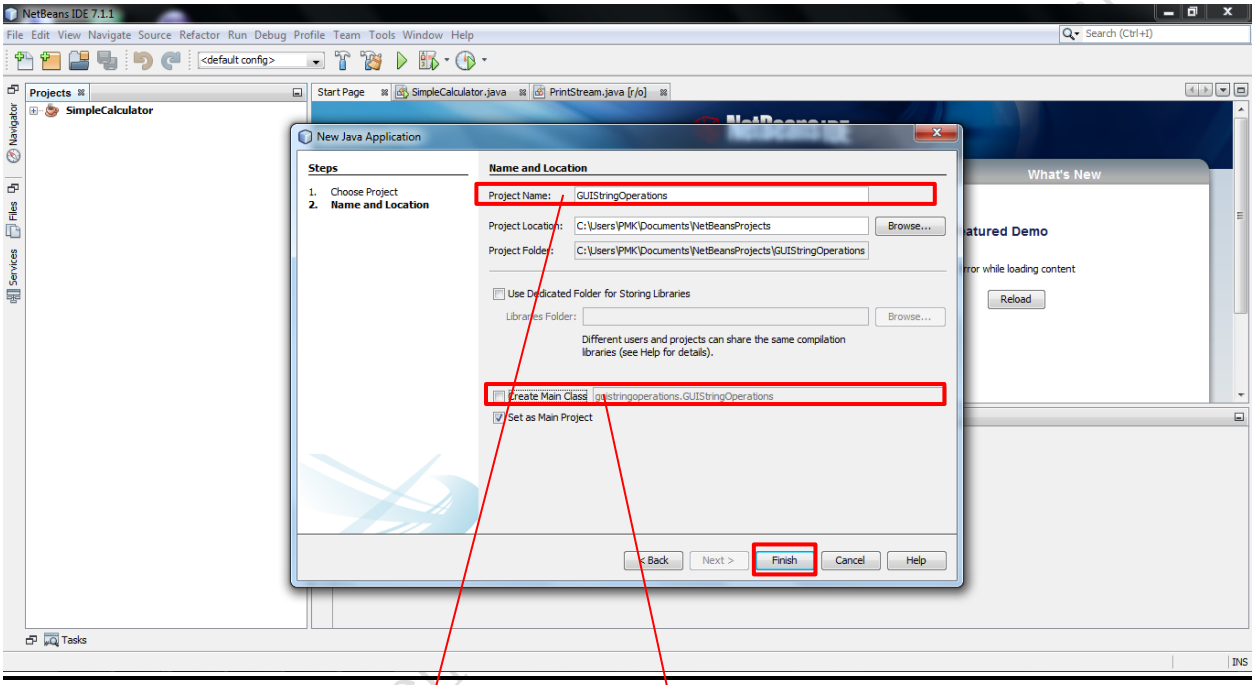
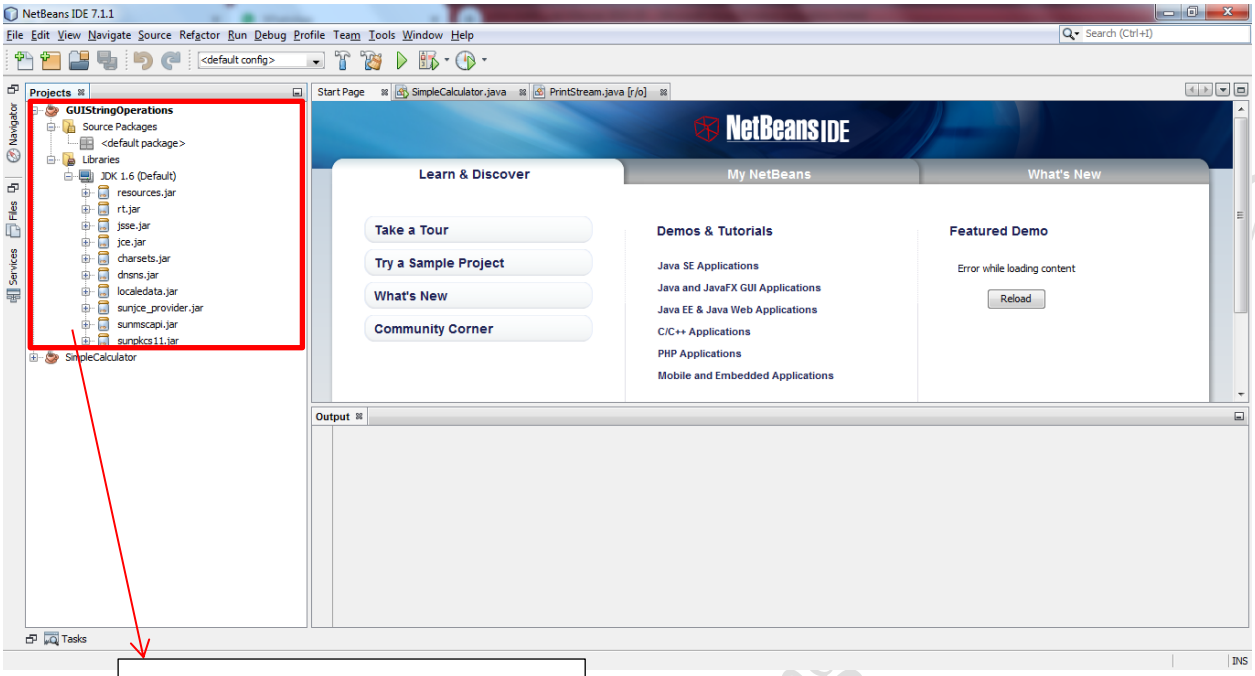


Fig: 36

Enter the project name, which will create…project folder with all required files…

For GUI based applications…we need to **uncheck** this option

- Now IDE is loaded with main project folder structure with required libraries as shown below.



Project folder with default package and library files.

Fig: 37

- Now we will add JFrame form to the project to do so follow these steps: **Select your GUI project right-click** on it→**New**→select **JFrame Form** option as shown below:

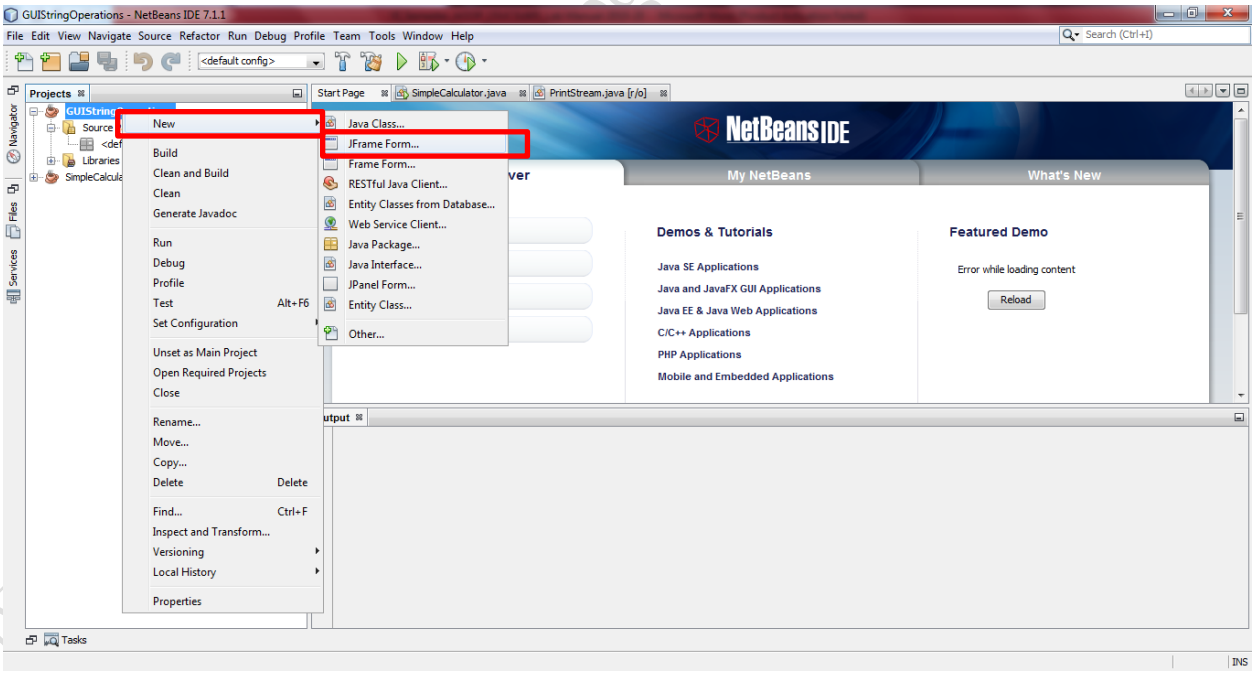


Fig: 38

- Add .java file to create GUI components to read user input(s) as shown below:

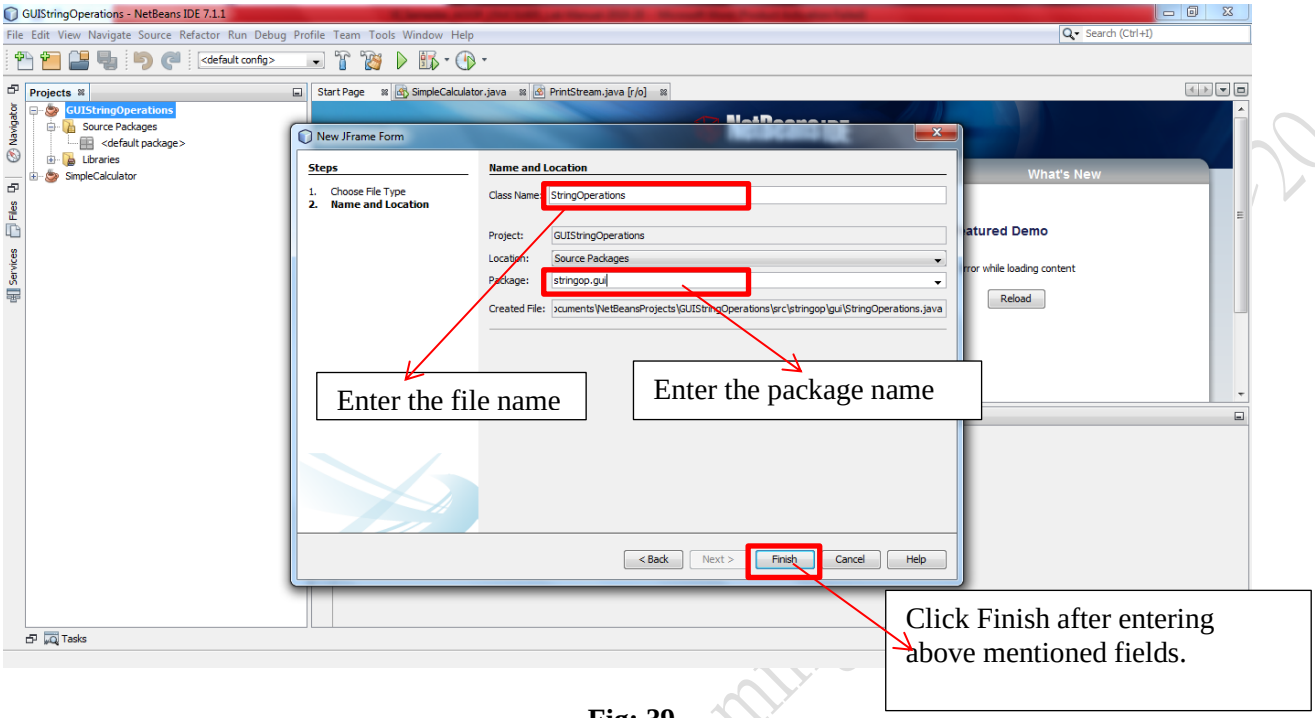


Fig: 39

- Now IDE has opened JFrame (which .java file) to create GUI as shown below:

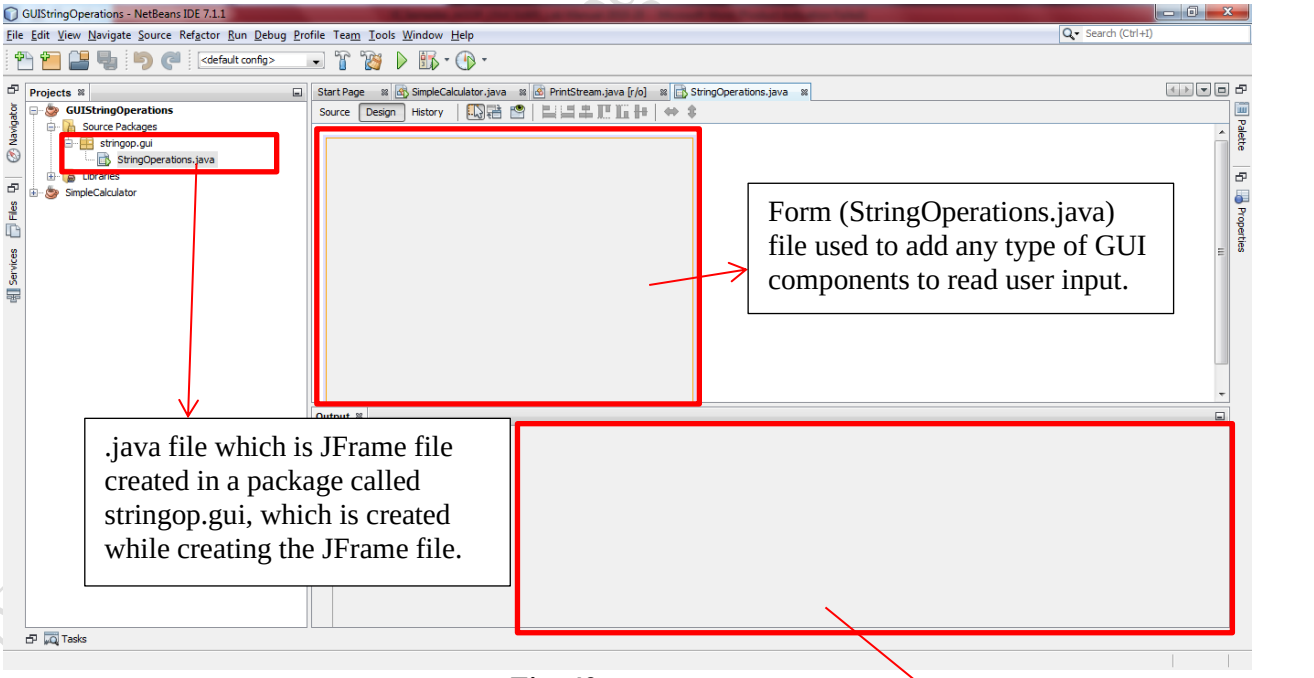


Fig: 40

- Size of the form (JFrame) can be increased by placing the cursor at the bottom corner of the form and drag it. Increase the size of the form depending on the number of components you want place. Then click on Palette and Properties tabs to place GUI components and assign name respectively:

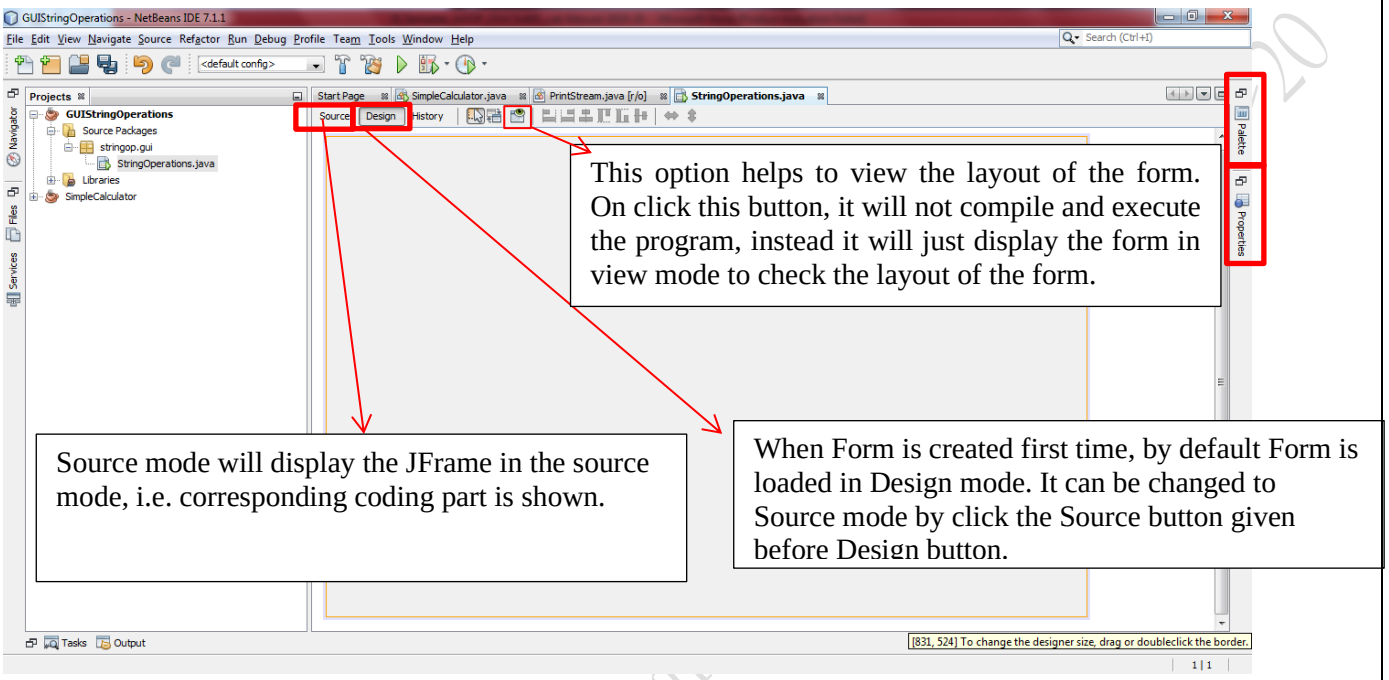


Fig: 41

- Now add all required GUI components as shown below:

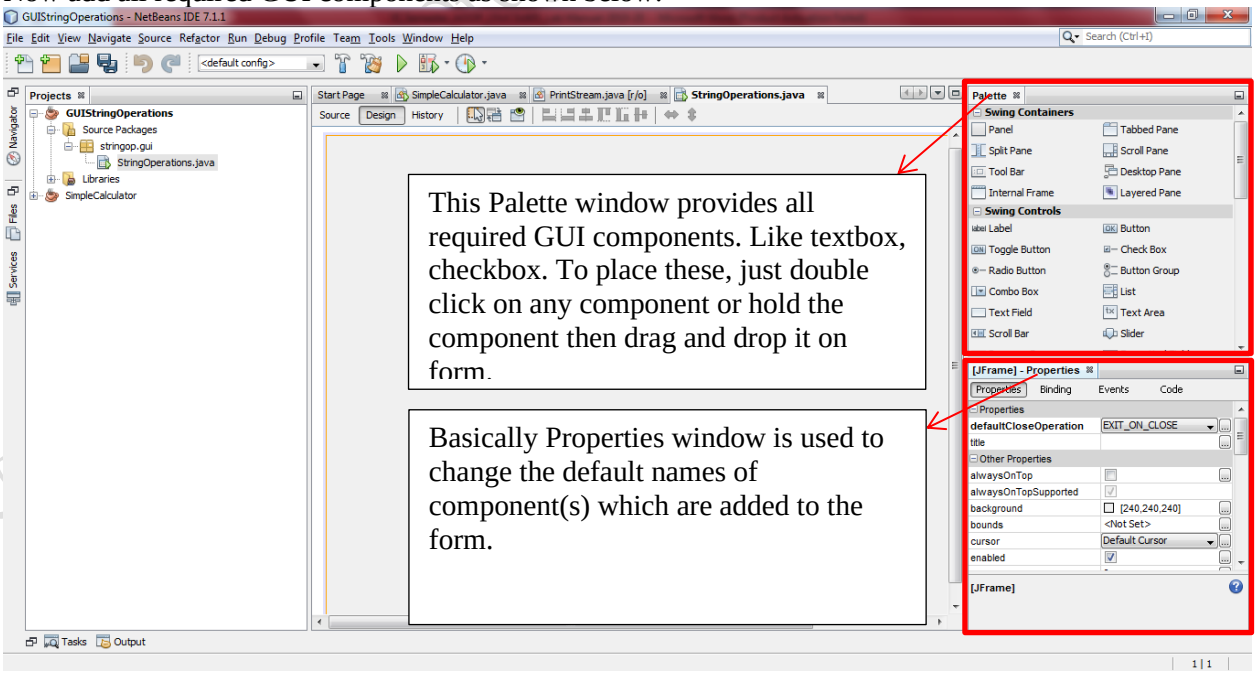


Fig: 42

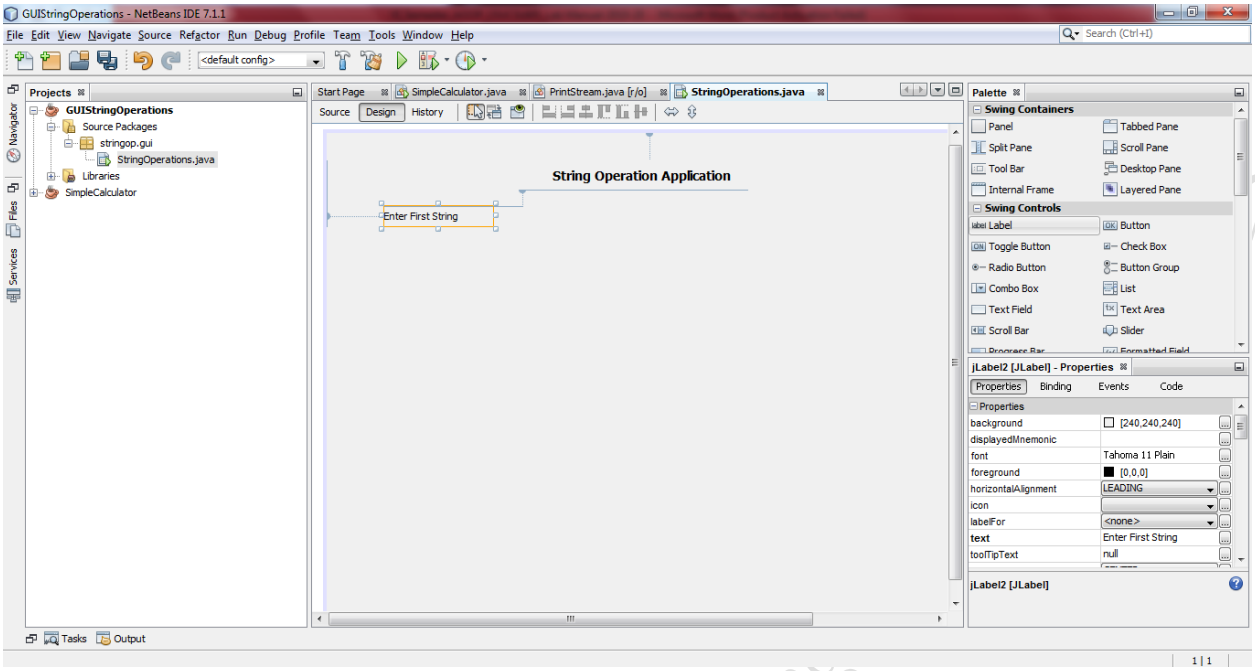

GUIStringOperations - NetBeans IDE 7.1.1
String Operation Application
Enter First String

Fig: 43

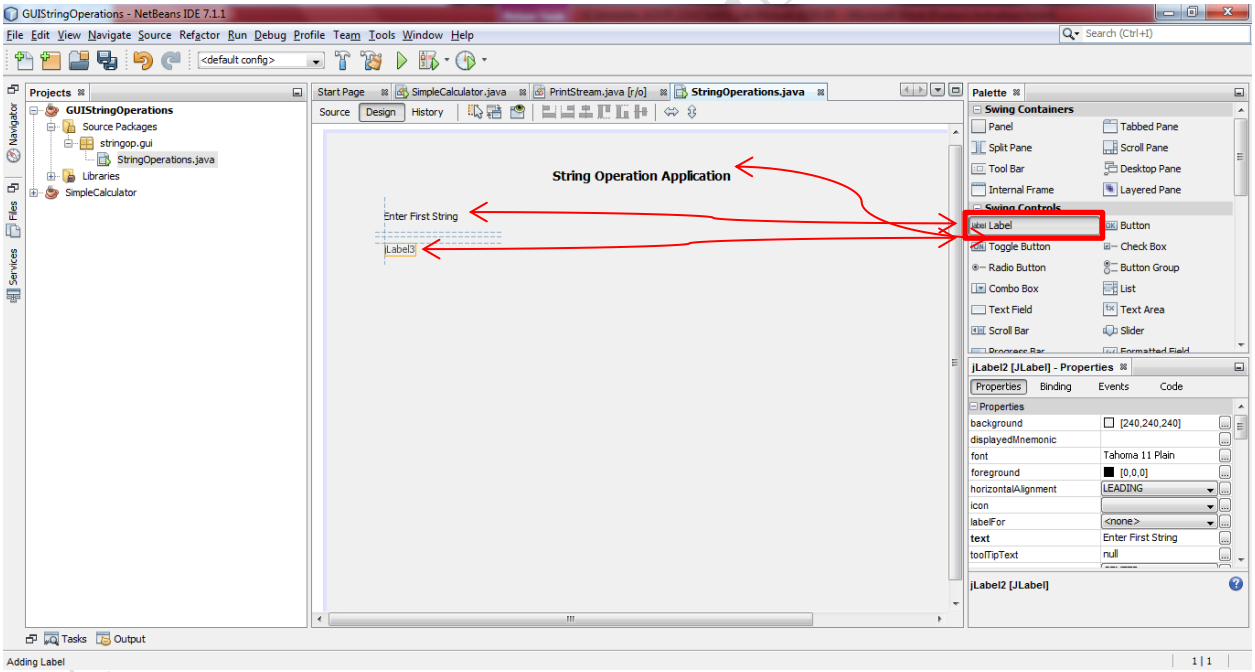

GUIStringOperations - NetBeans IDE 7.1.1
String Operation Application
Enter First String
jLabel3

Fig: 44

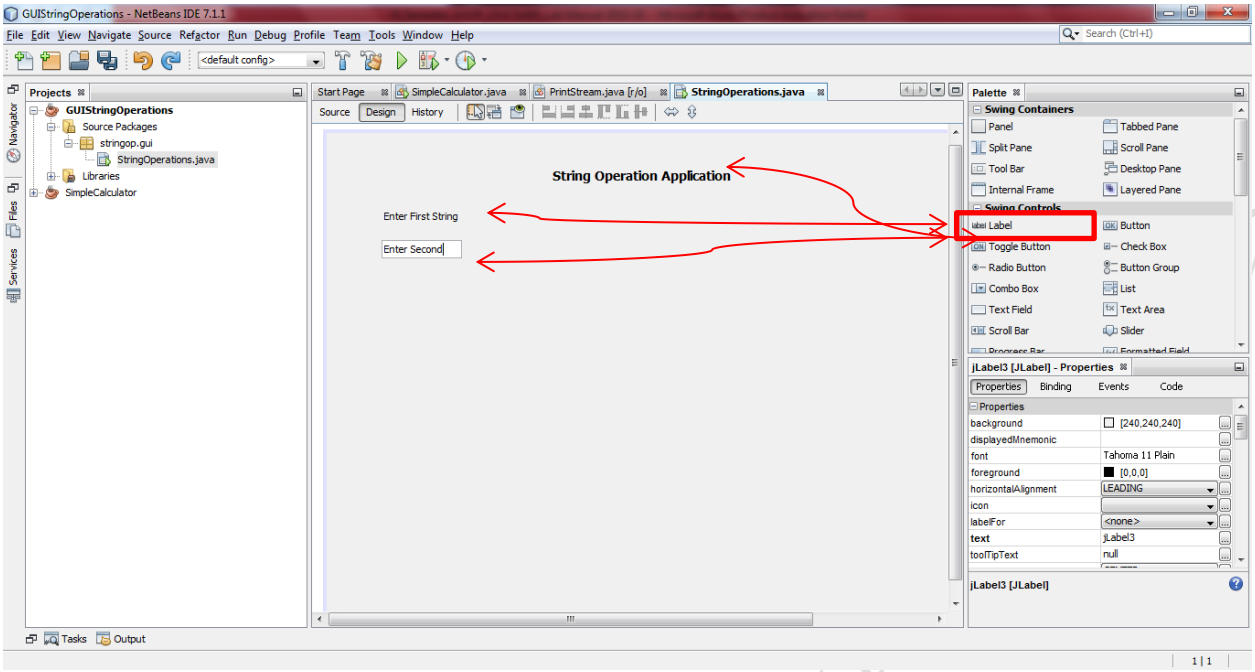

GUIStringOperations - NetBeans IDE 7.1.1
String Operation Application
Enter First String
Enter Second

Fig: 45

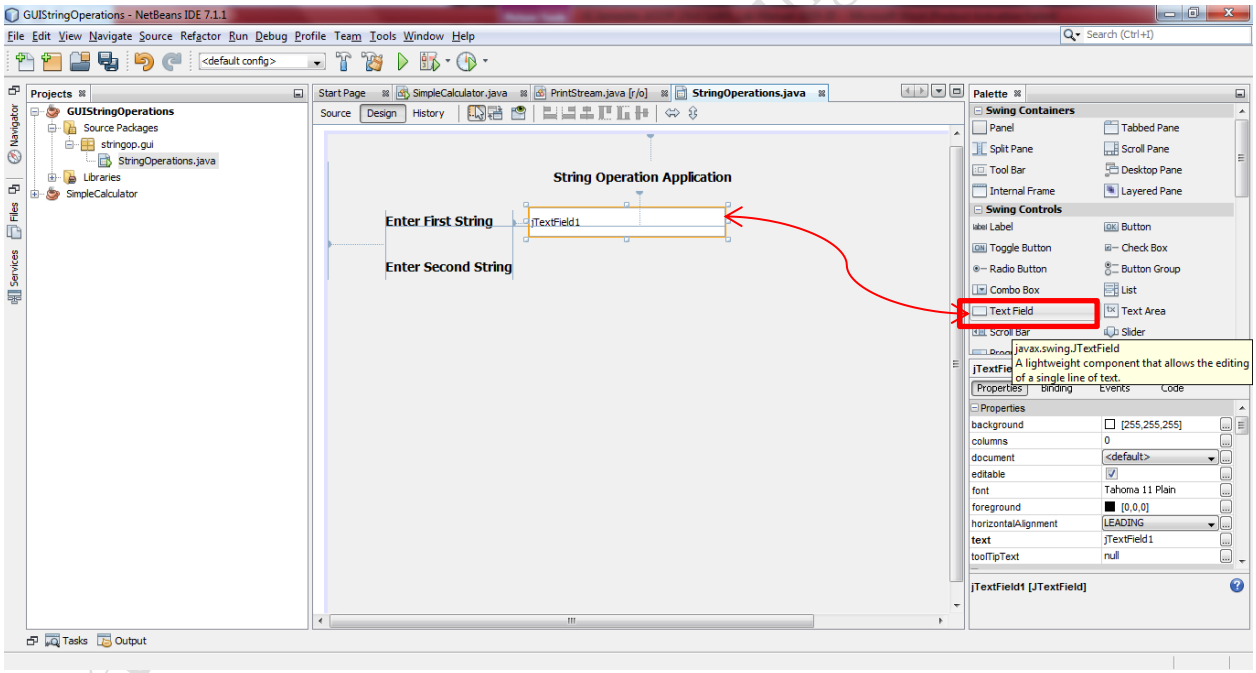

GUIStringOperations - NetBeans IDE 7.1.1
String Operation Application
Enter First String
jTextField1
Enter Second String
javax.swing.JTextField
A lightweight component that allows the editing of a single line of text.

Fig: 46

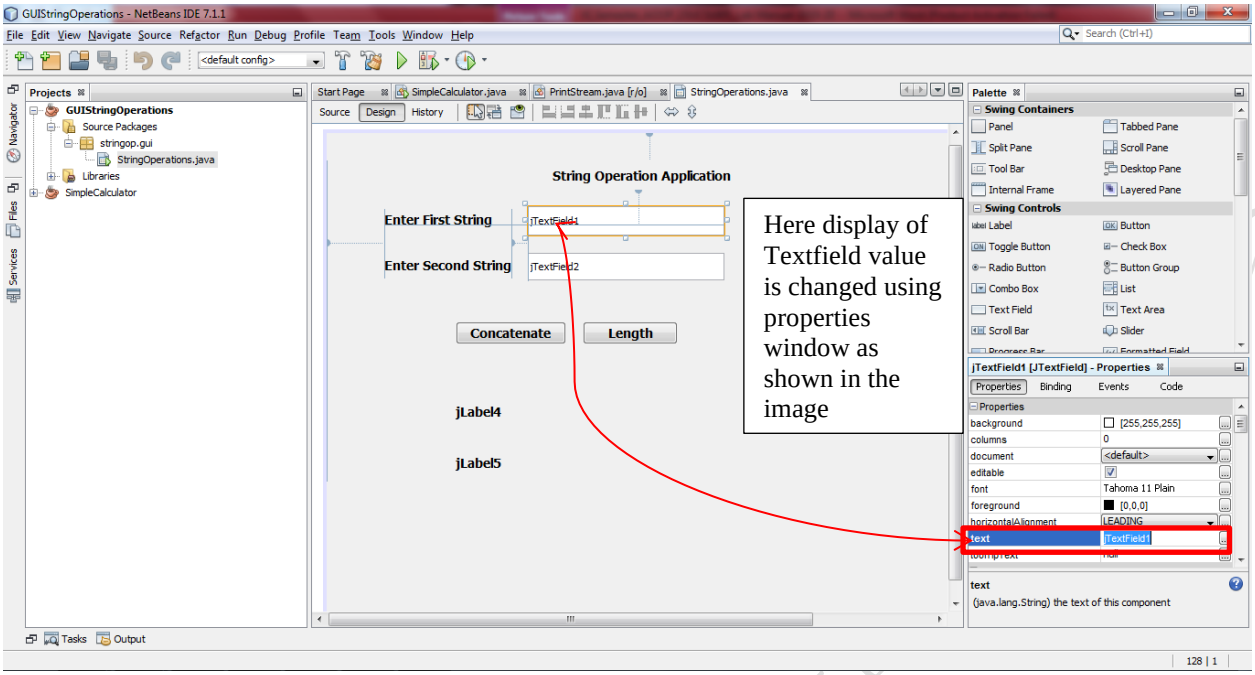


Fig: 47

- Final layout of the form.

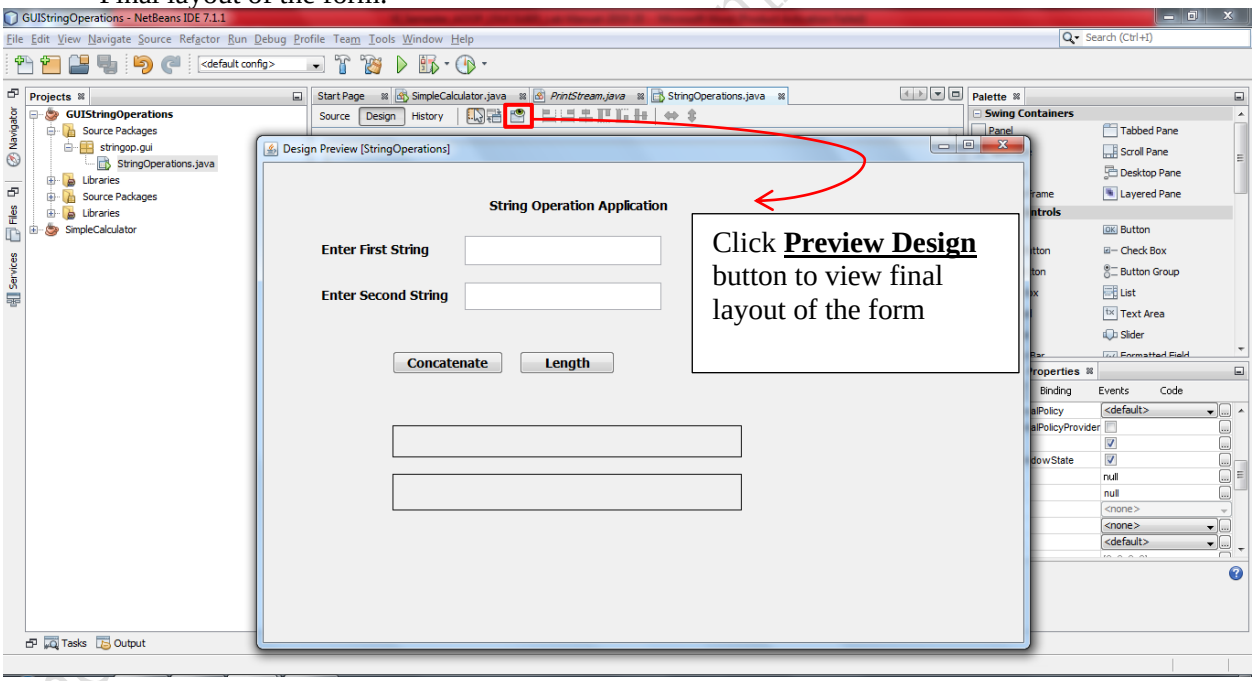


Fig: 48

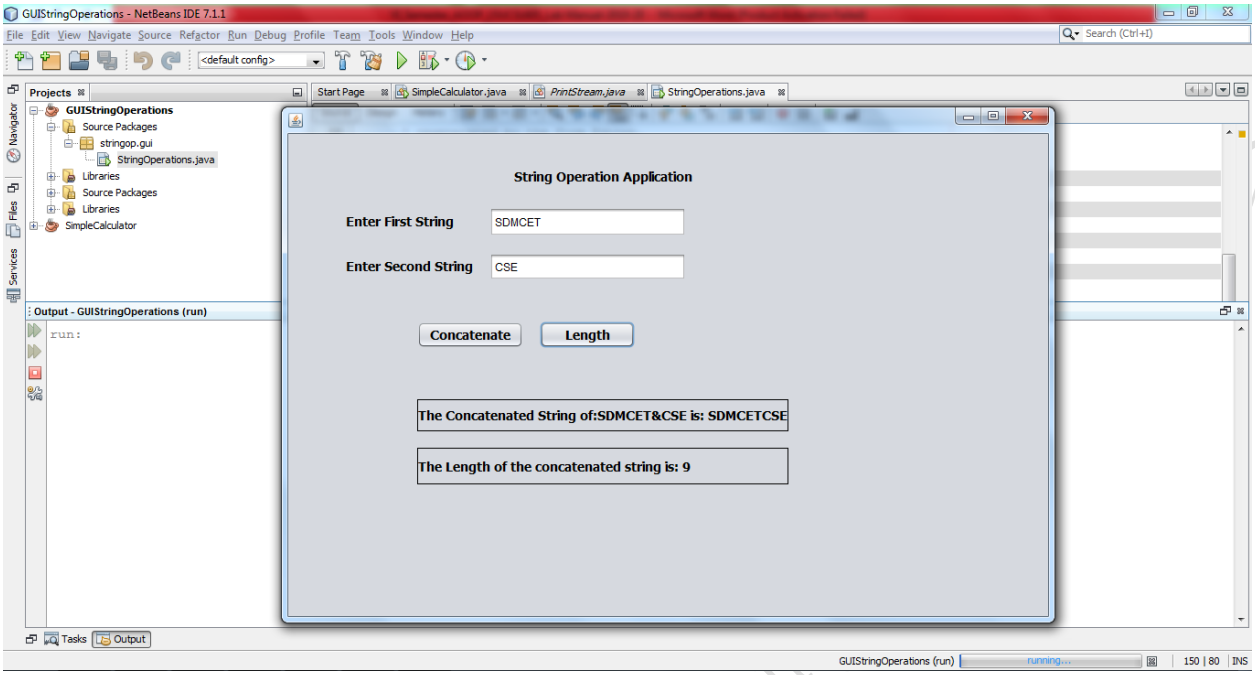


Fig: 49

- Click **Source** tab to view/change to source mode. On click of source tab, IDE will display the customized as well as auto generated code.

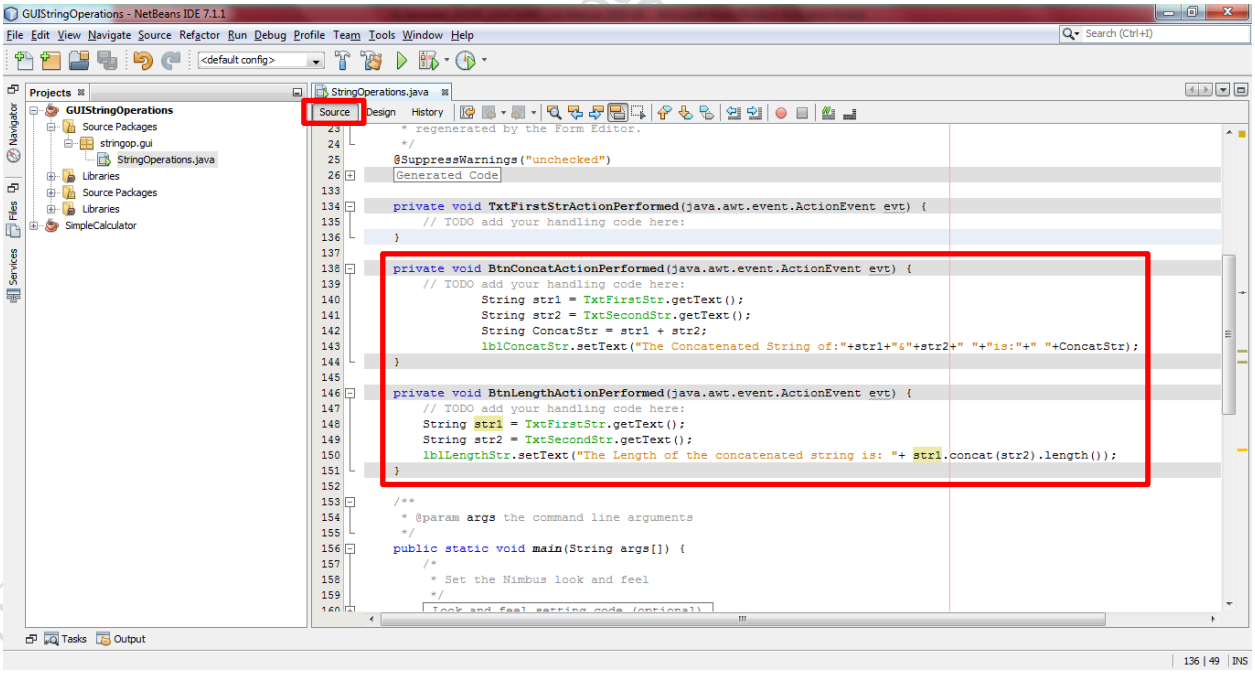


Fig: 50

7. Design and development of web application (Java EE) using NetBeans IDE

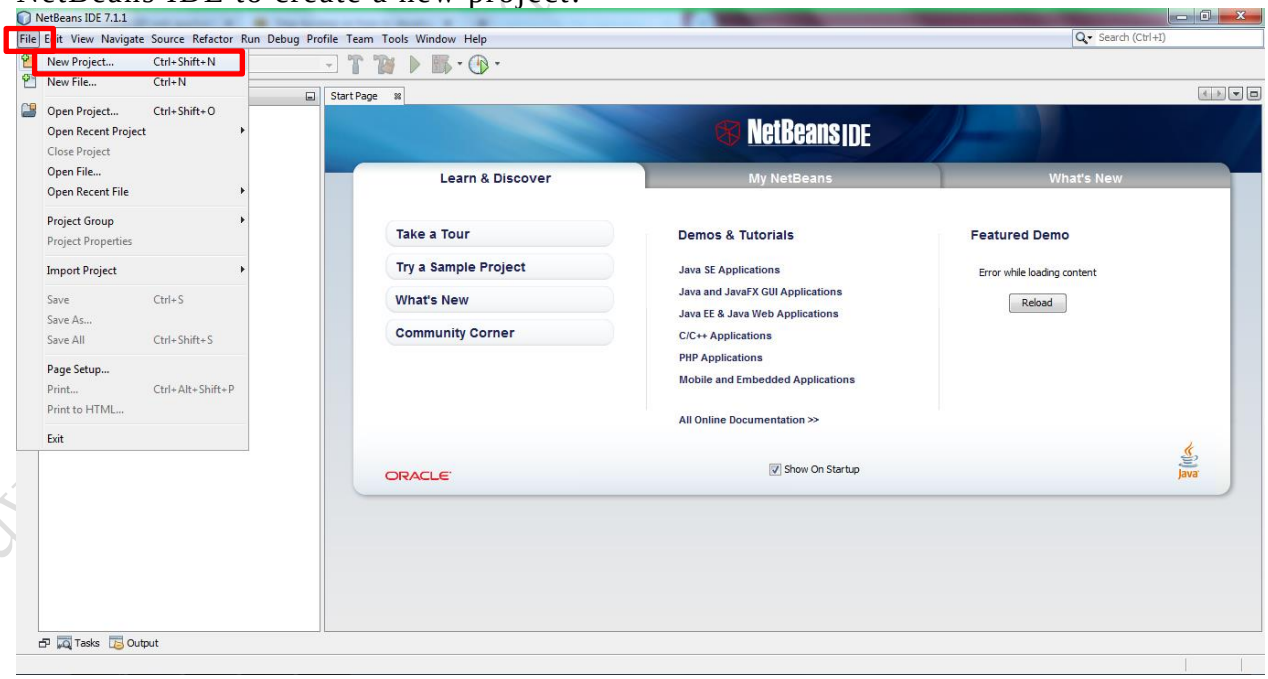
7.1 Problem Statement: Design a simple web application to collect and display the student information as a registration form with the following requirement:

1. Create a RegistrationForm.jsp page to collect input details using different web controls like Textbox, Checkbox, and DropdownList etc. The following data to be collected from the user:
   i. Student Name
   ii. Student USN
   iii. Email Address
   iv. Branch
   v. CGPA.
2. On click of submit button, page should be redirected to another page called DisplayDetails.jsp and data should be displayed in a tabular format. Along with the submit button, reset button to be added in RegistrationForm.jsp page to clear the user input data.

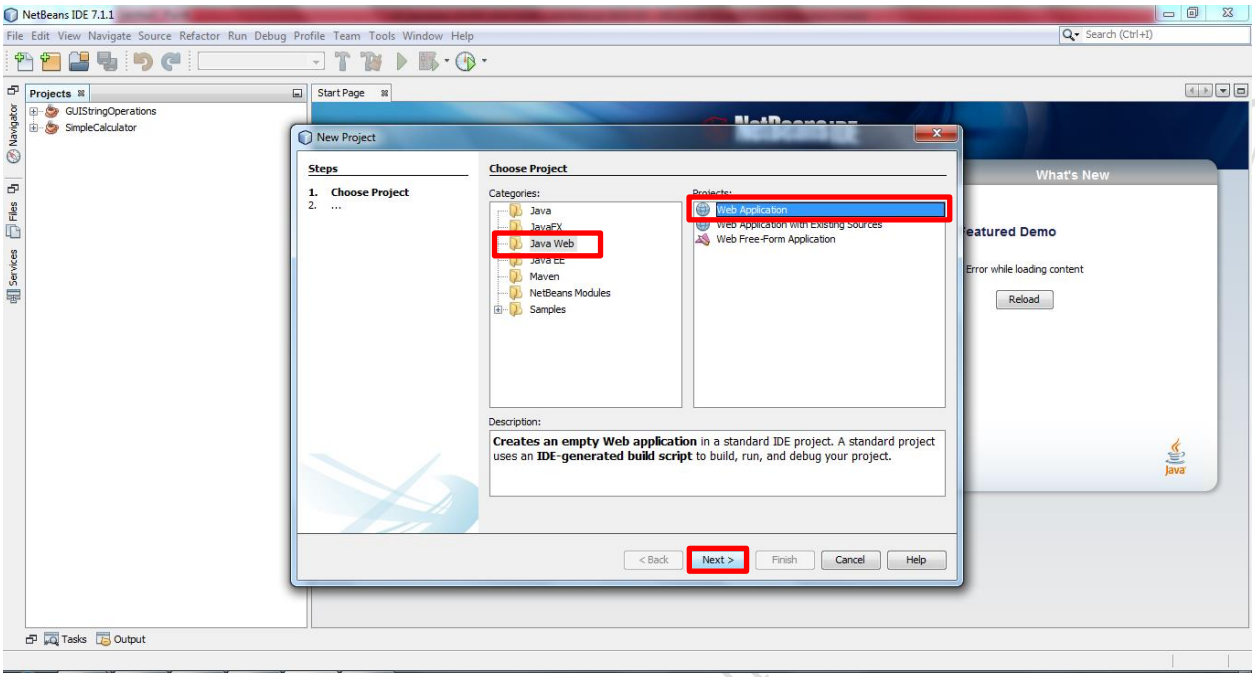
Note: Here GlassFish Server 3.1.2 is used to deploy the web application. Using Apache tomcat server we can deploy the application, for that we need to configure the IDE. Additional server is not required As NetBeans IDE comes with GlassFish server.

Step-1: Create a new project (Web application).

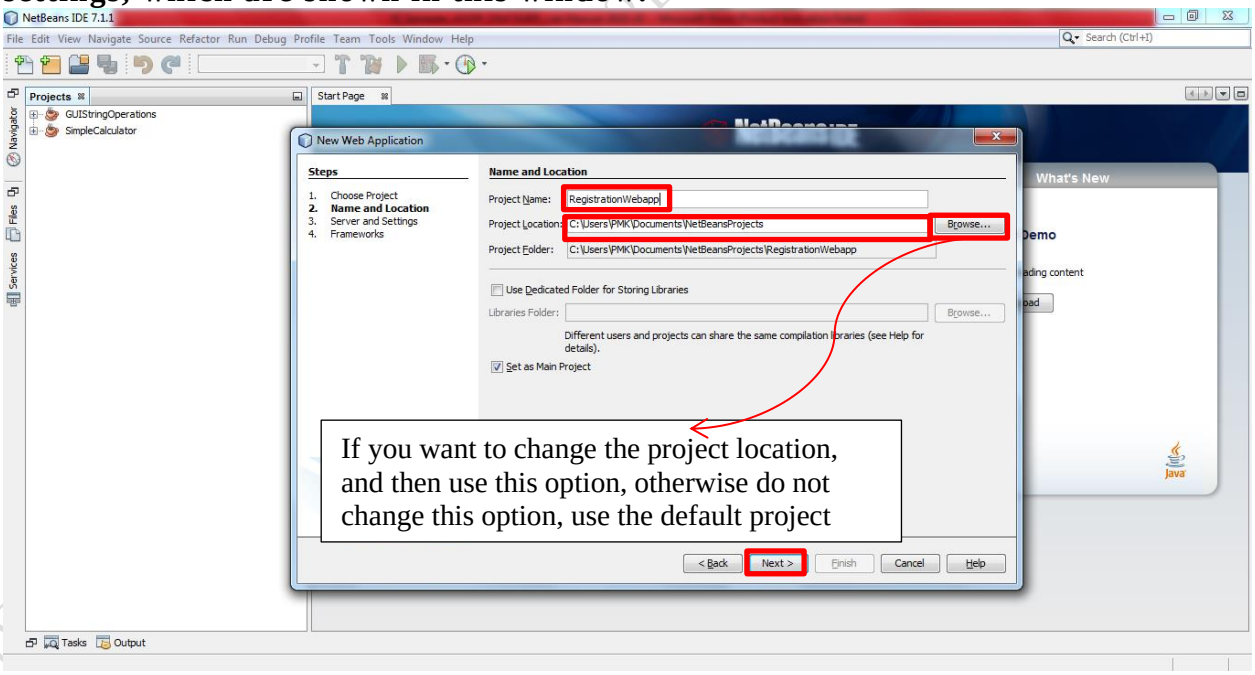
Follow the section 6.1 and the Fig(s): 26 to 29 to launch and open the NetBeans IDE to create a new project.



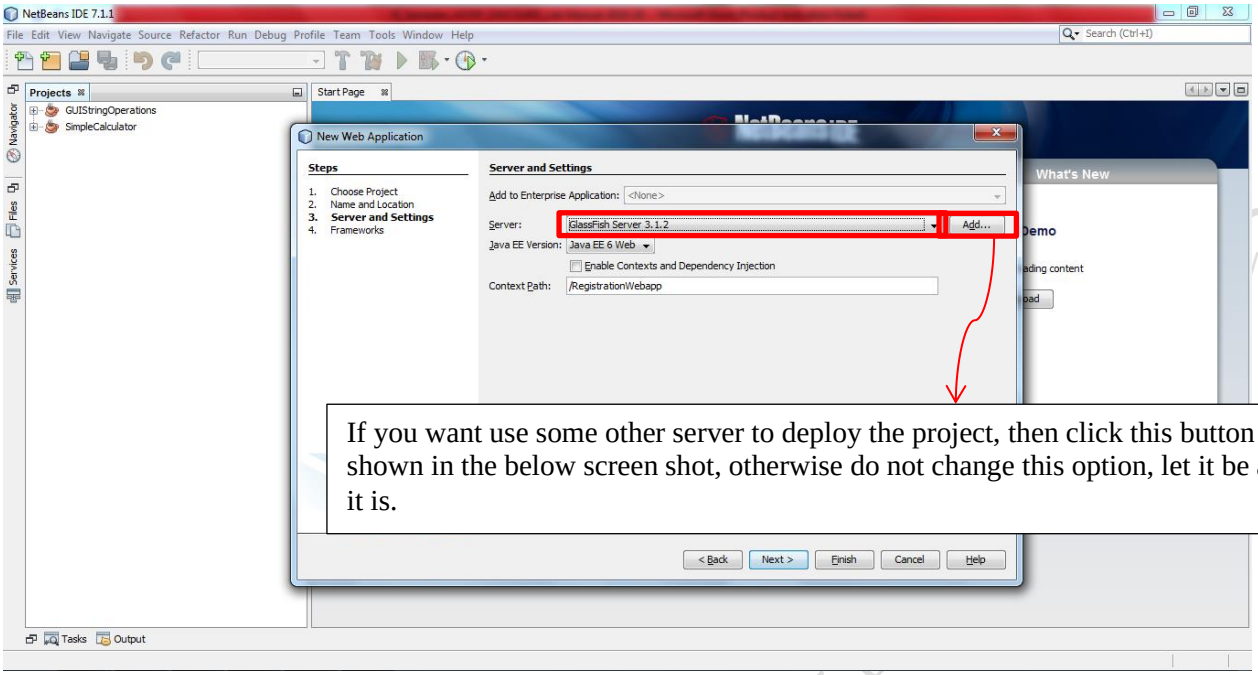
Step-2: Enter the project name and click next button to create a new project and its folder structure.

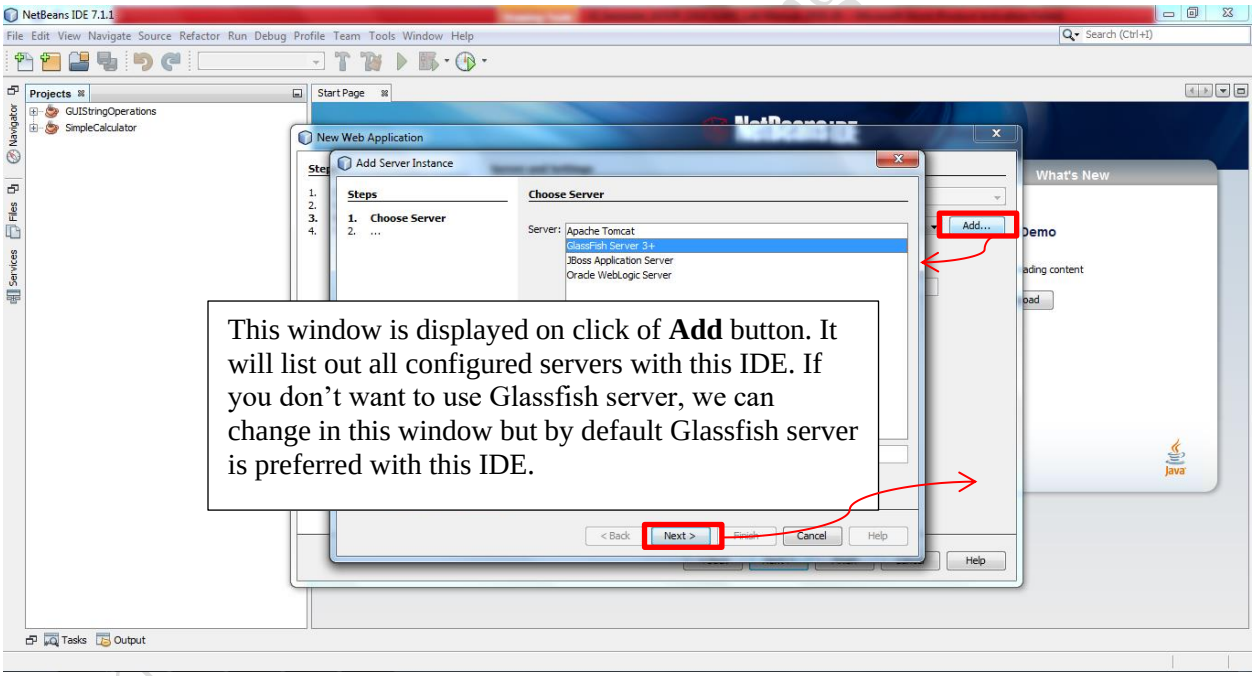


Step-3: Enter the project name and click next button. Do not change other settings, which are shown in this window.

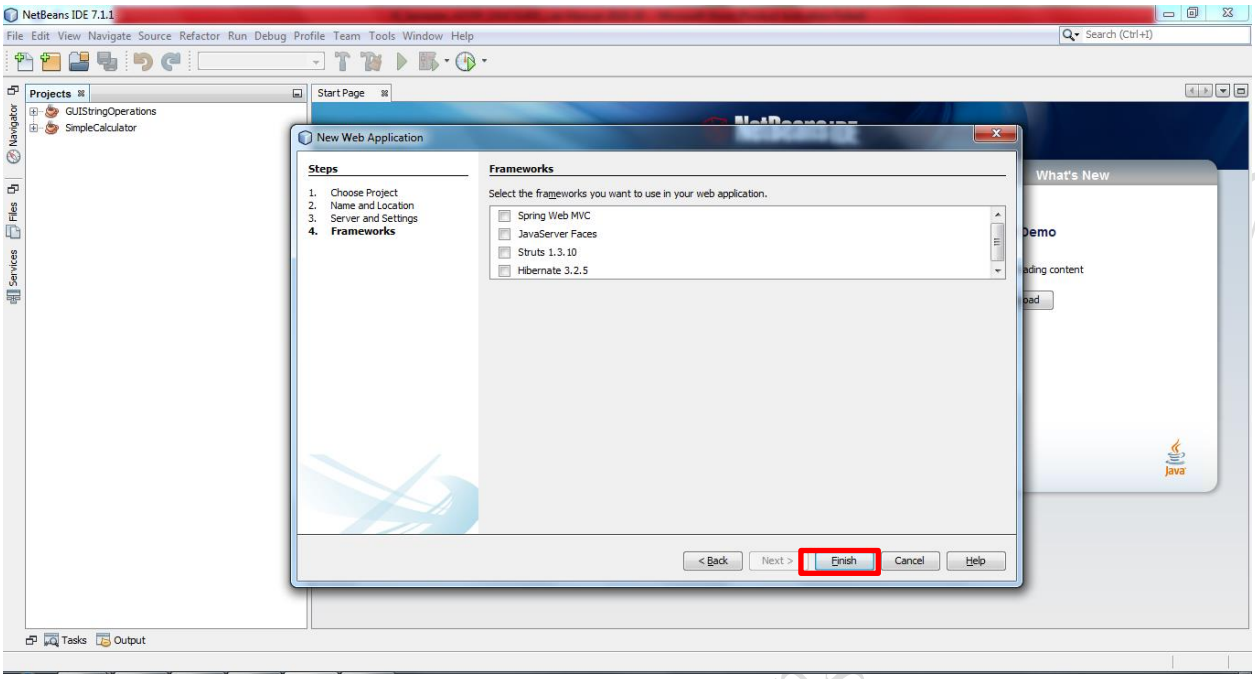


Step-4: Window to either change or use default server. By default Glassfish server is configured with this IDE and web application is deployed and hosted on this server.

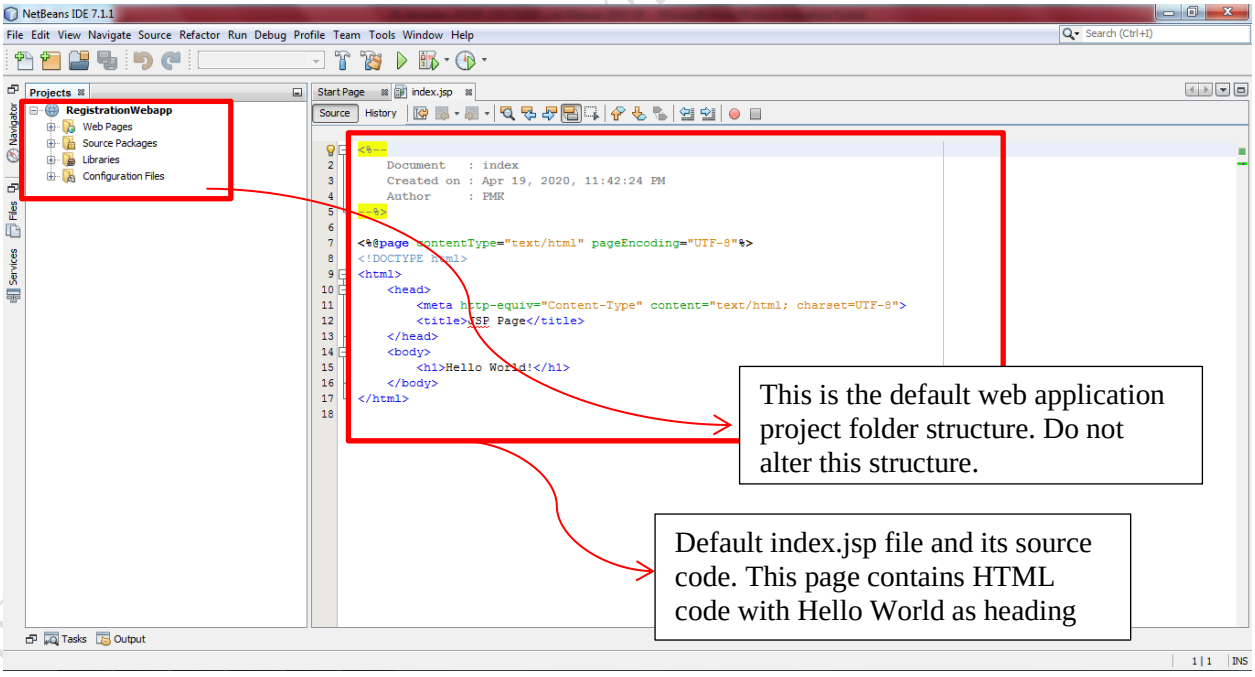




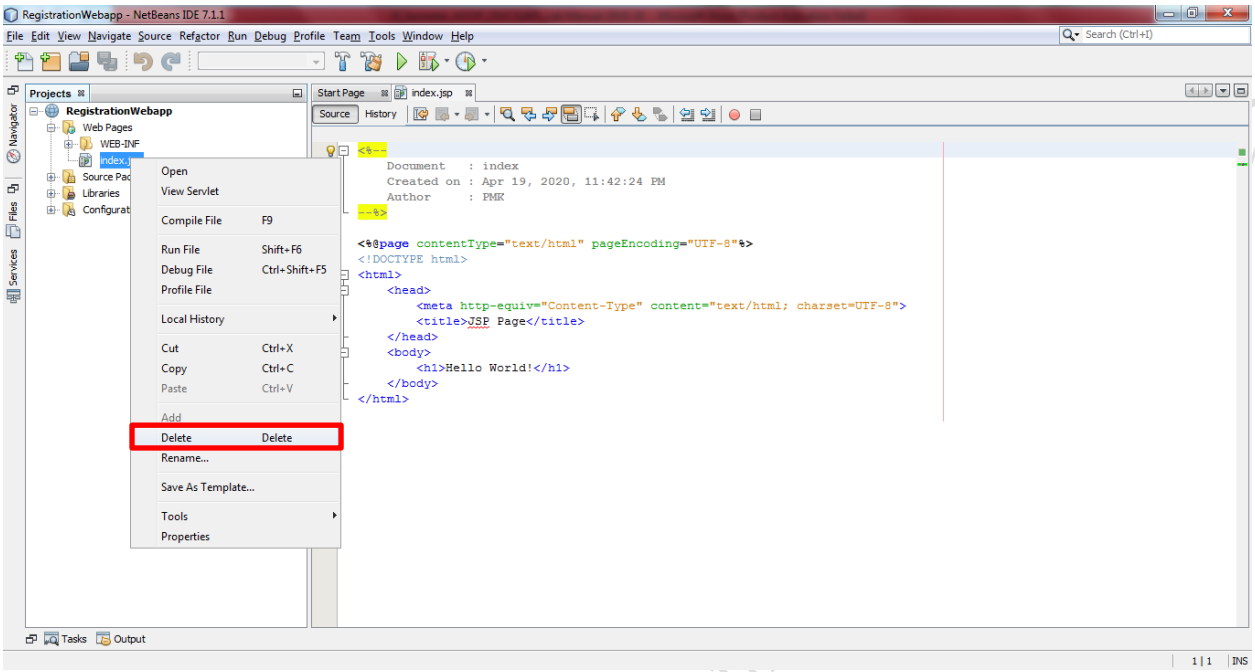
Step-5: Another window with default settings on click of next button, do not change any configurations click Finish **button to create web project folder structure.**



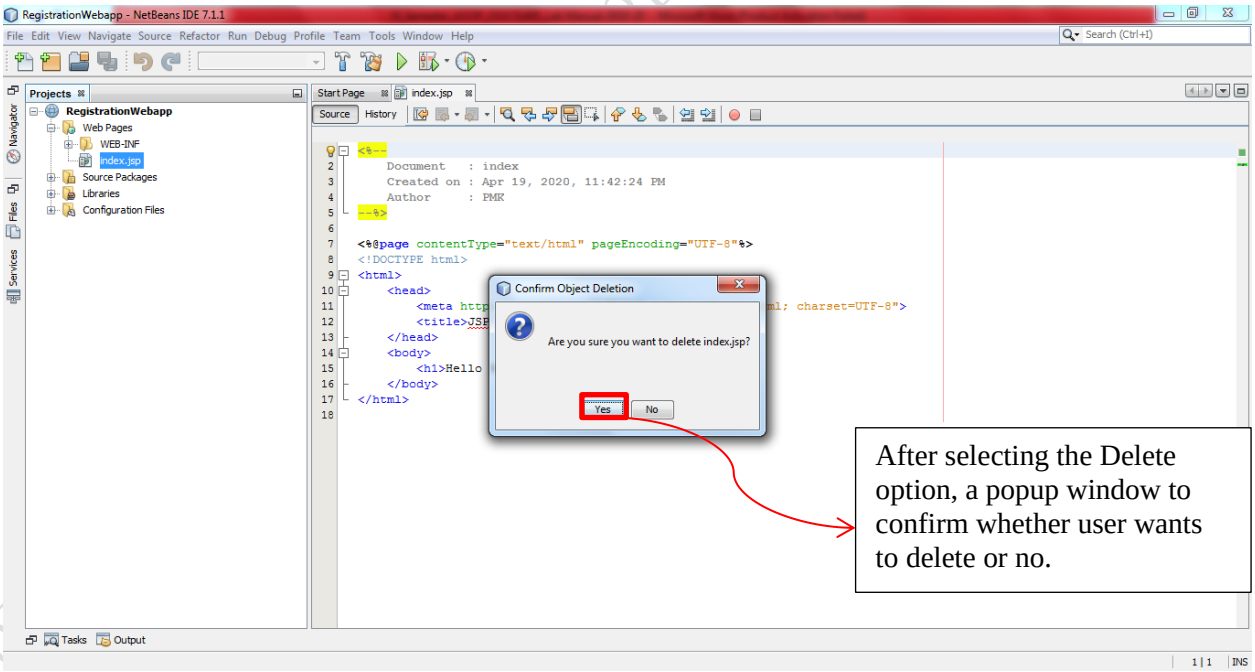
Step-6: Web application project folder structure with all required files. By default IDE will add index.jsp page as shown below.



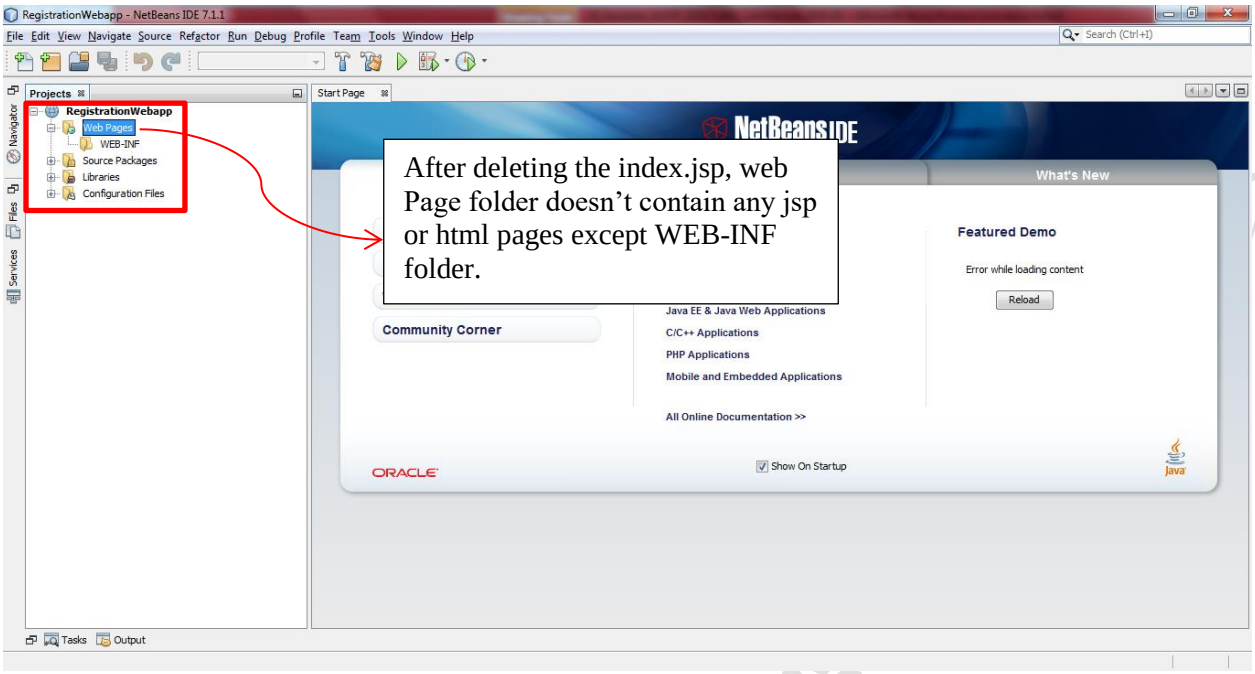
Step-7: Now remove the index.jsp page and create new jsp page as given in the problem statement. To remove any file from the project folder, go the project and then Web Pages→ expand the folder, select the file (.jsp) → right-click on it and select the Delete option as shown below.



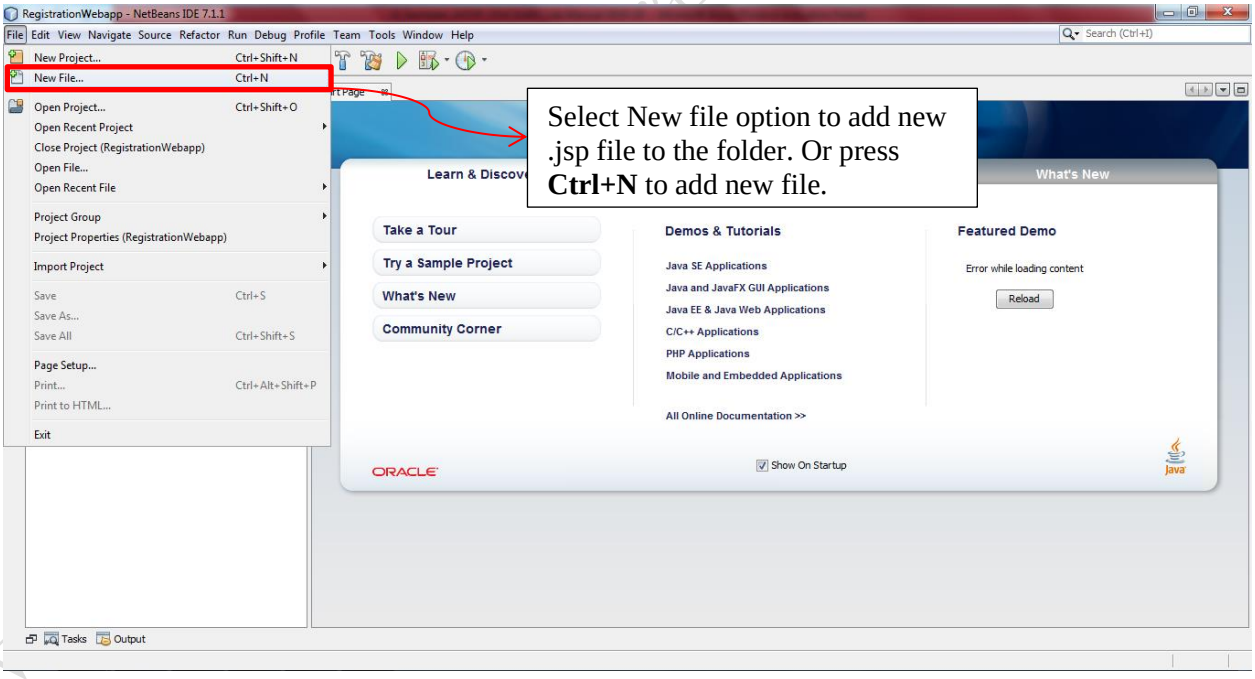
Step-8:



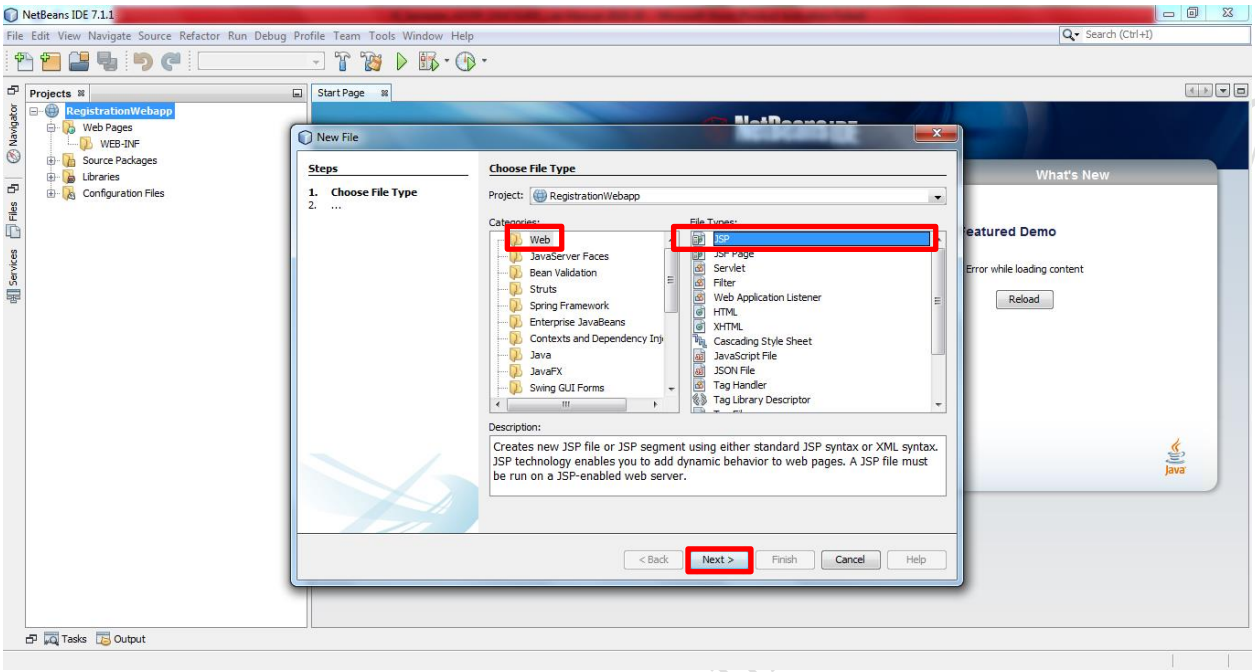
Step-9: After deleting the .jsp file from the project folder.



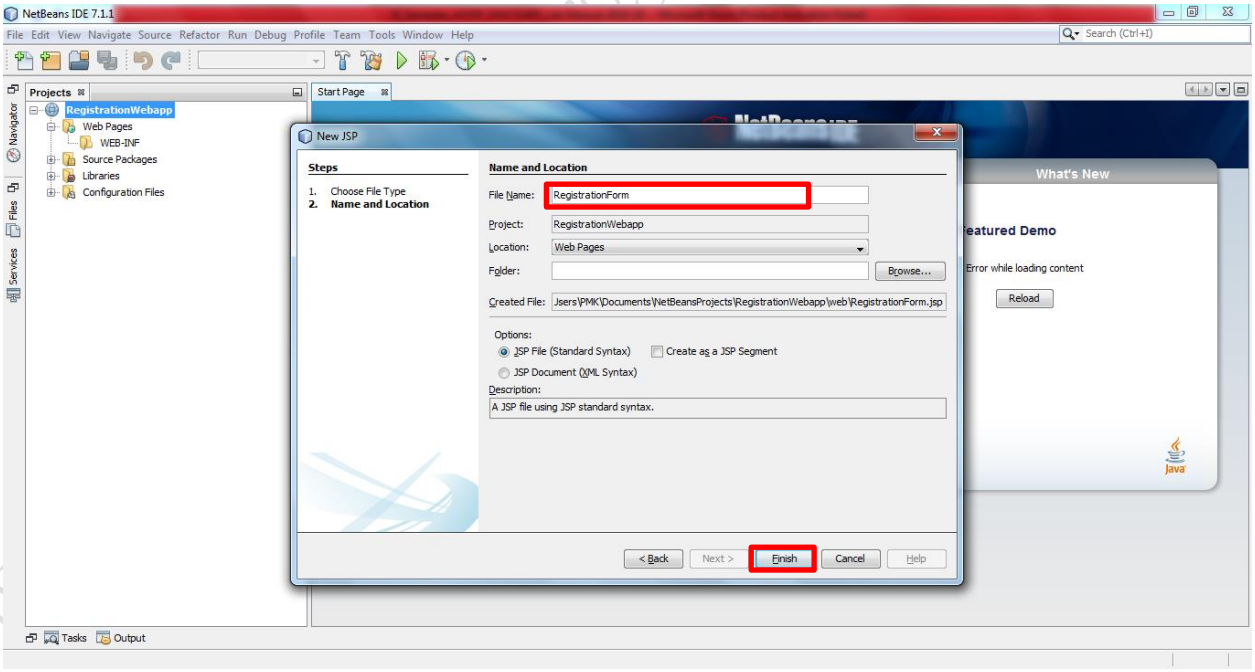
Step-10: Add a new .jsp file to the project folder. Select the project folder→ File menu→New File..



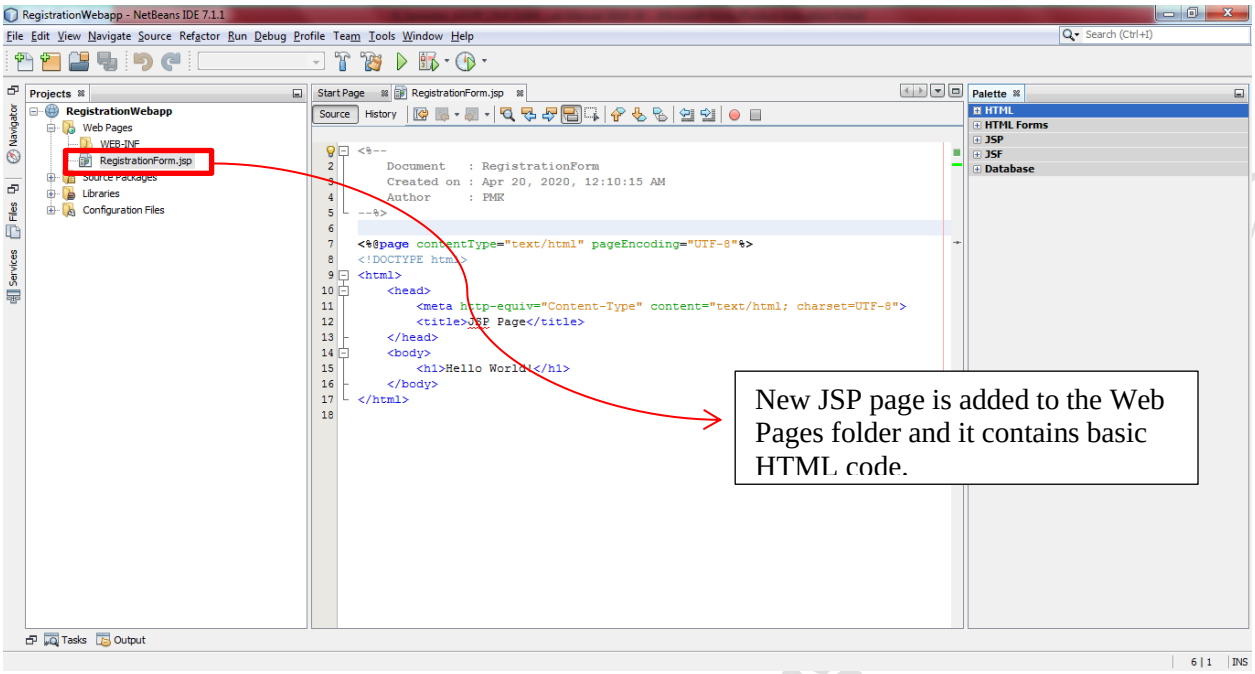
Step-11: Select Web from categories and JSP from File Type as shown below and click Next button to proceed.



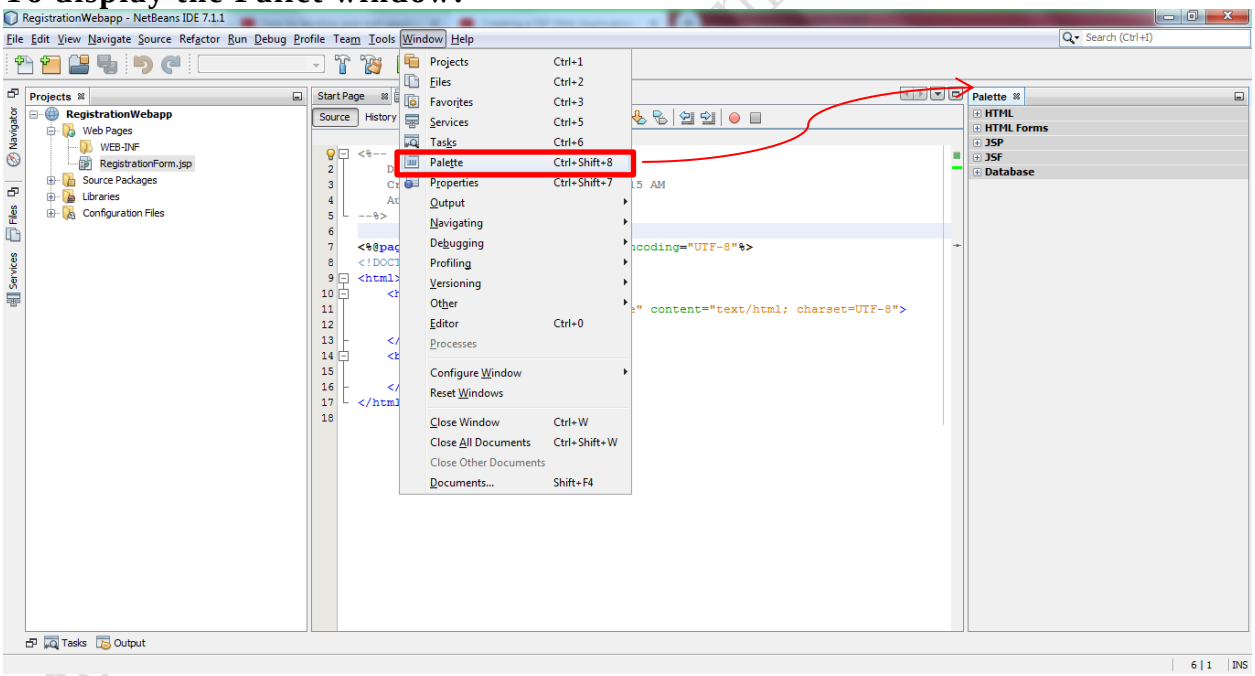
Step -12: Enter the JSP file name and click Finish button. Do not change any other settings.



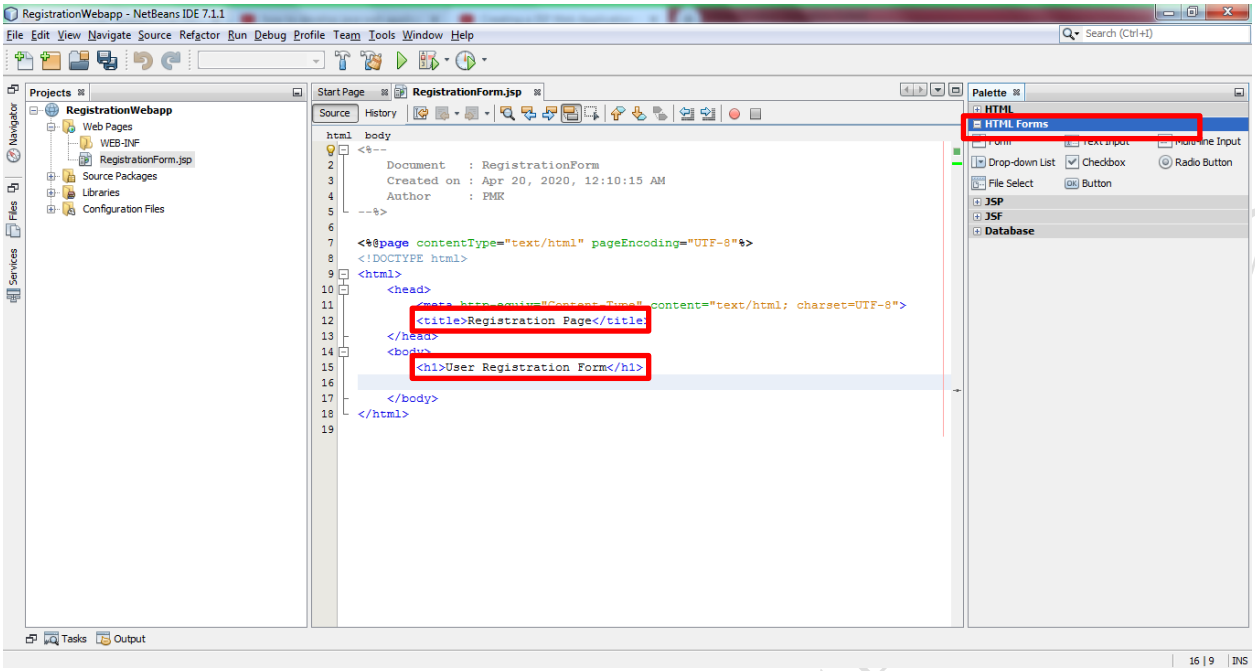
Step-13:



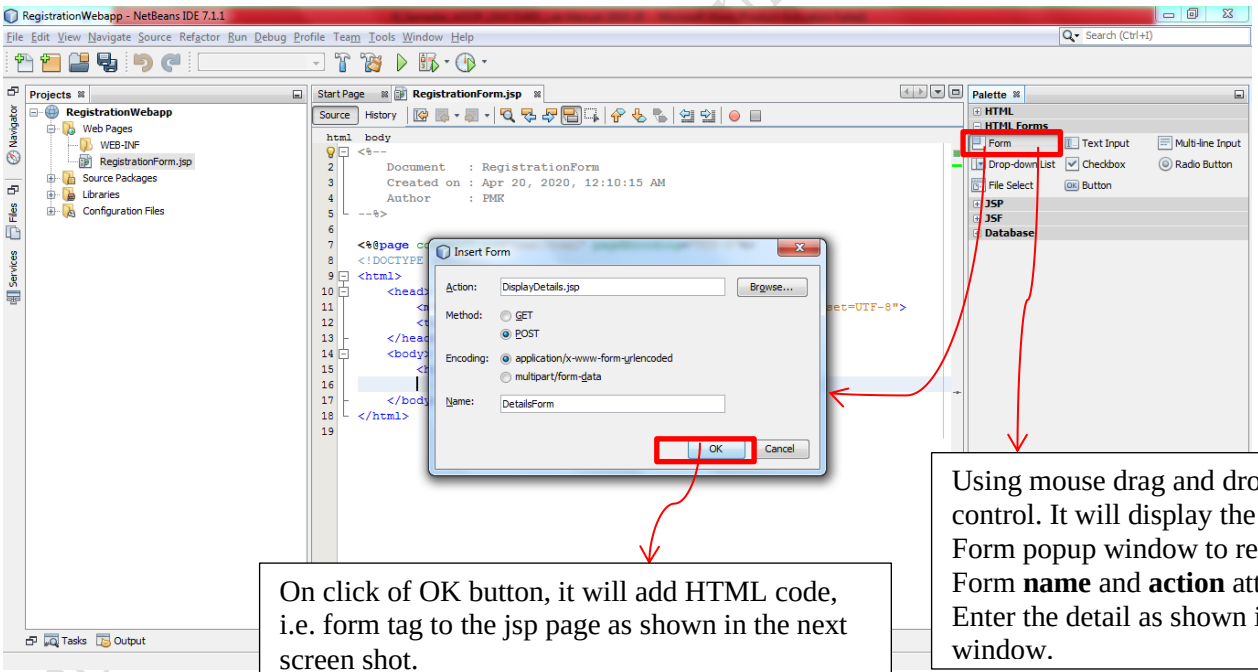
To display the Pallet window:



Step-14: Change the Title and header tags as shown in the screen shot and expand the HTML Form tab of Pallet window as highlighted:

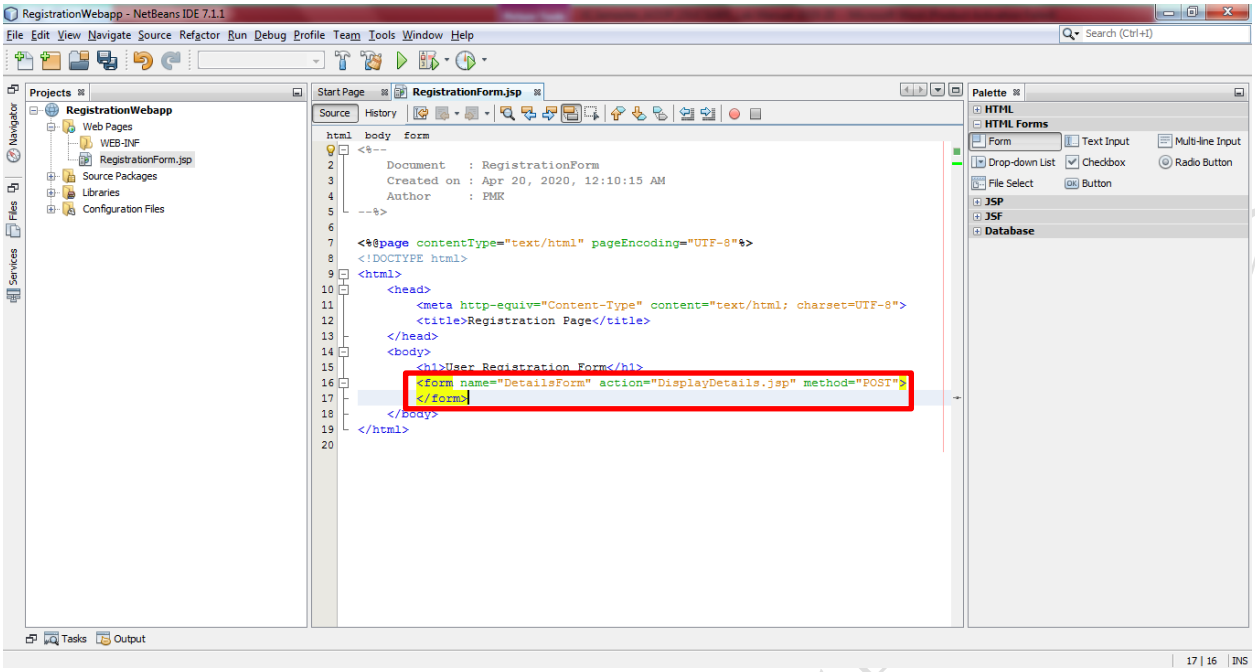


Step-15: Now create form and add all required input controls as shown below:

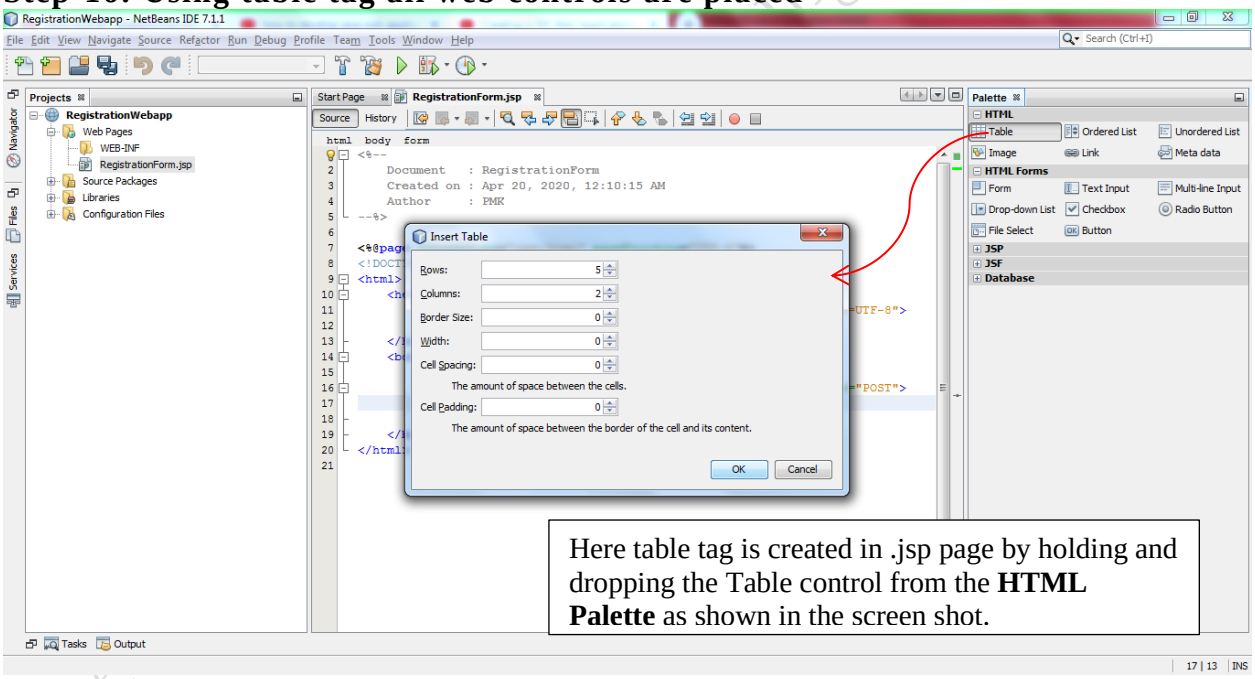


On click of OK button, it will add HTML code, i.e. form tag to the jsp page as shown in the next screen shot.

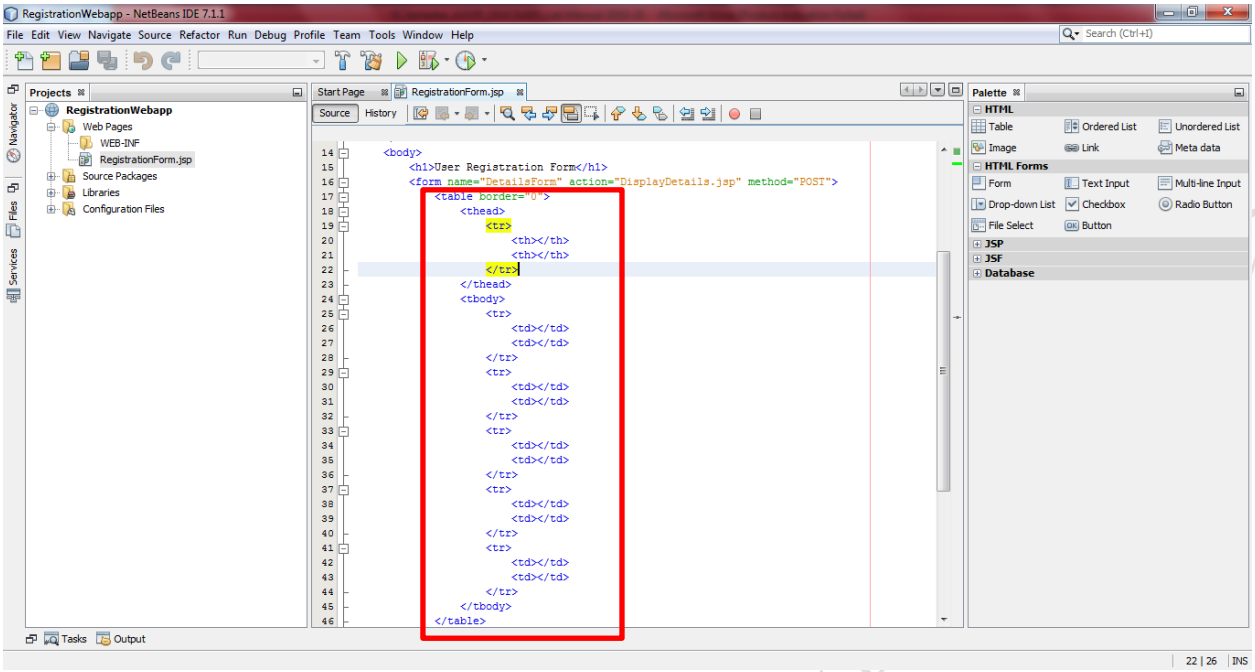
Using mouse drag and drop this control. It will display the Insert Form popup window to read Form **name** and **action** attribute. Enter the detail as shown in the window.

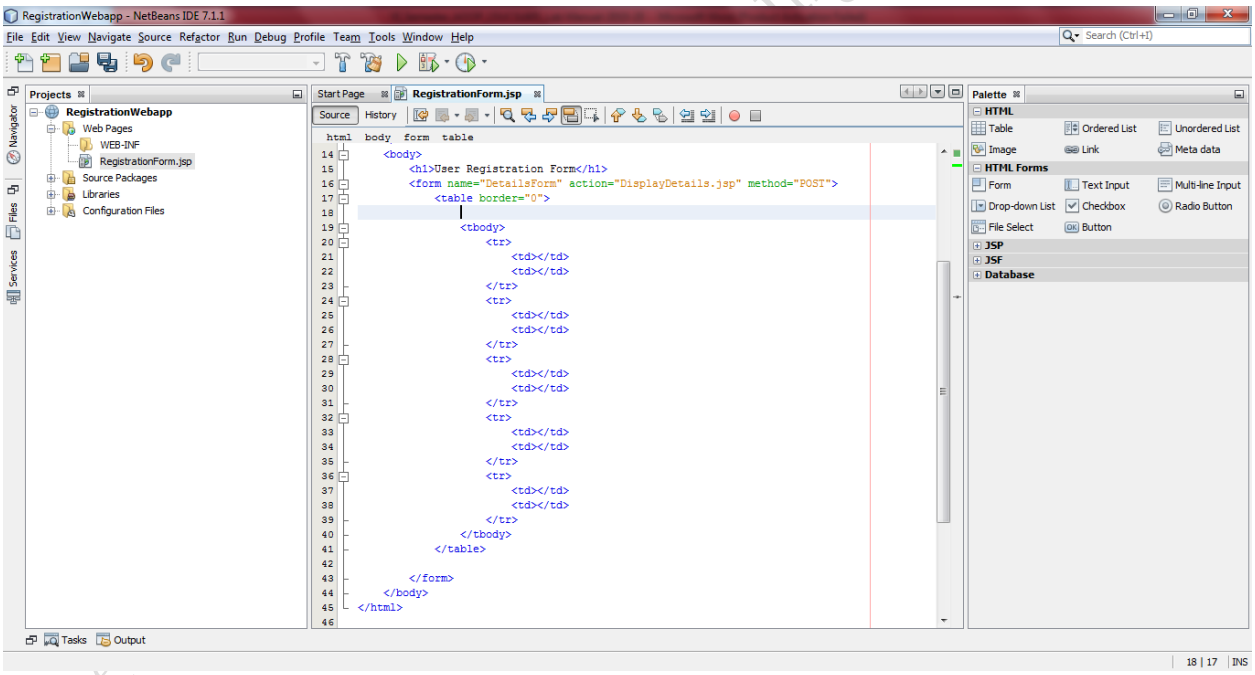


Step-16: Using table tag all web controls are placed



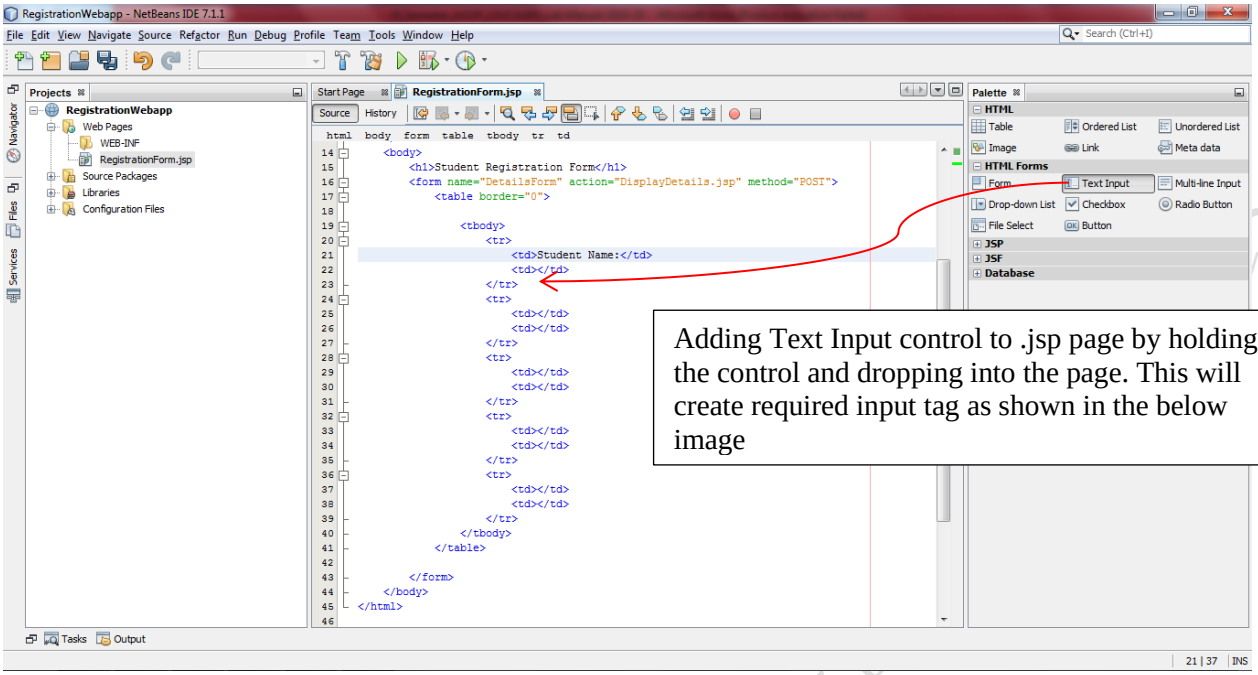
After dropping the html control from the palette to the .jsp page, IDE will generate the table tag with tr and td tags nested in it as shown in the below screen shots. In this example thead and its nested tags are removed. Similarly all required web controls are placed by dragging and dropping to the .jsp page



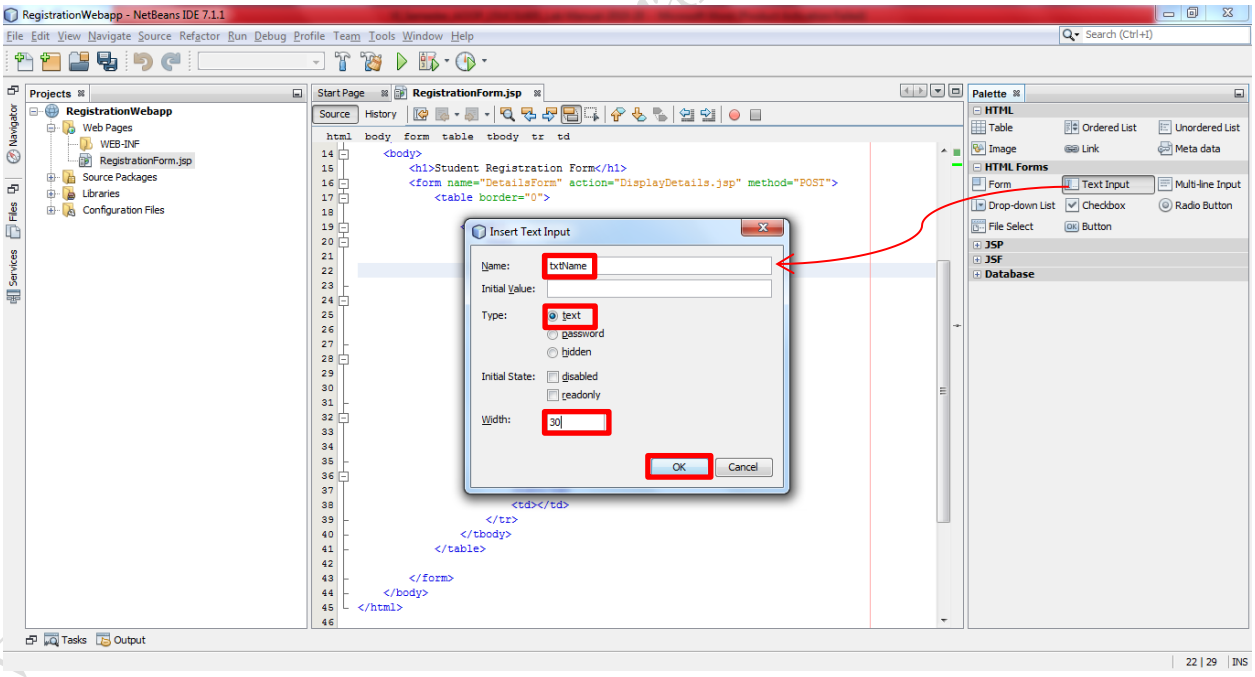


Step-17:

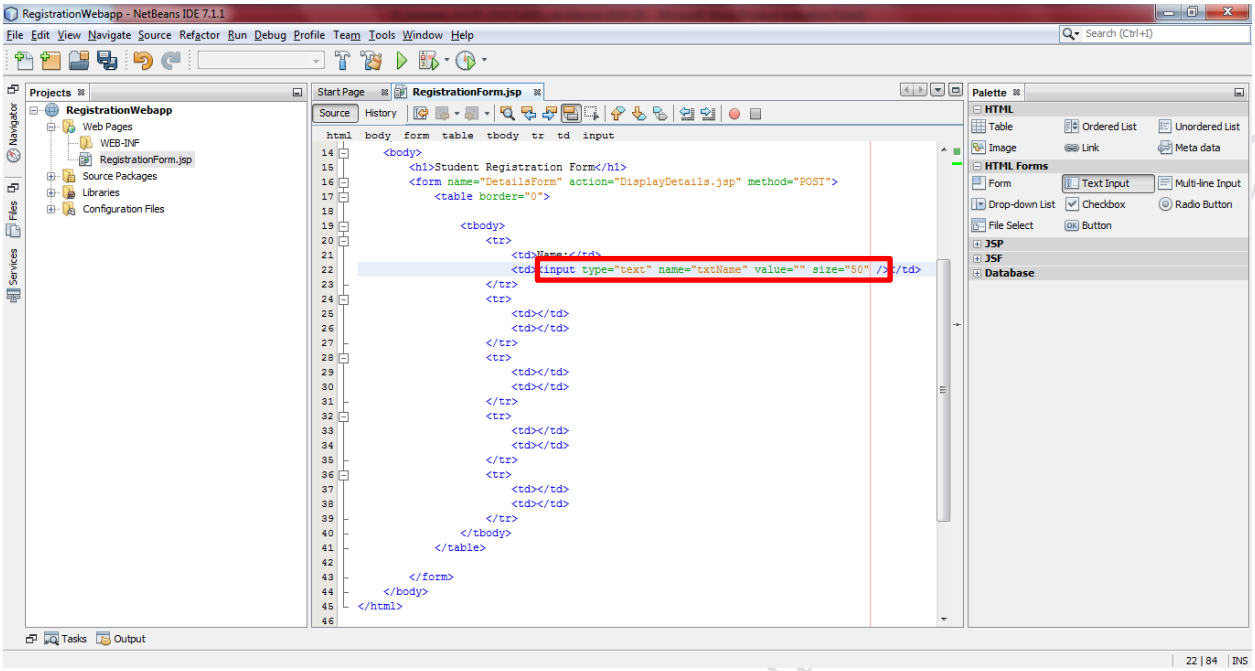
Now remaining web controls are added to the .jsp page as shown below:



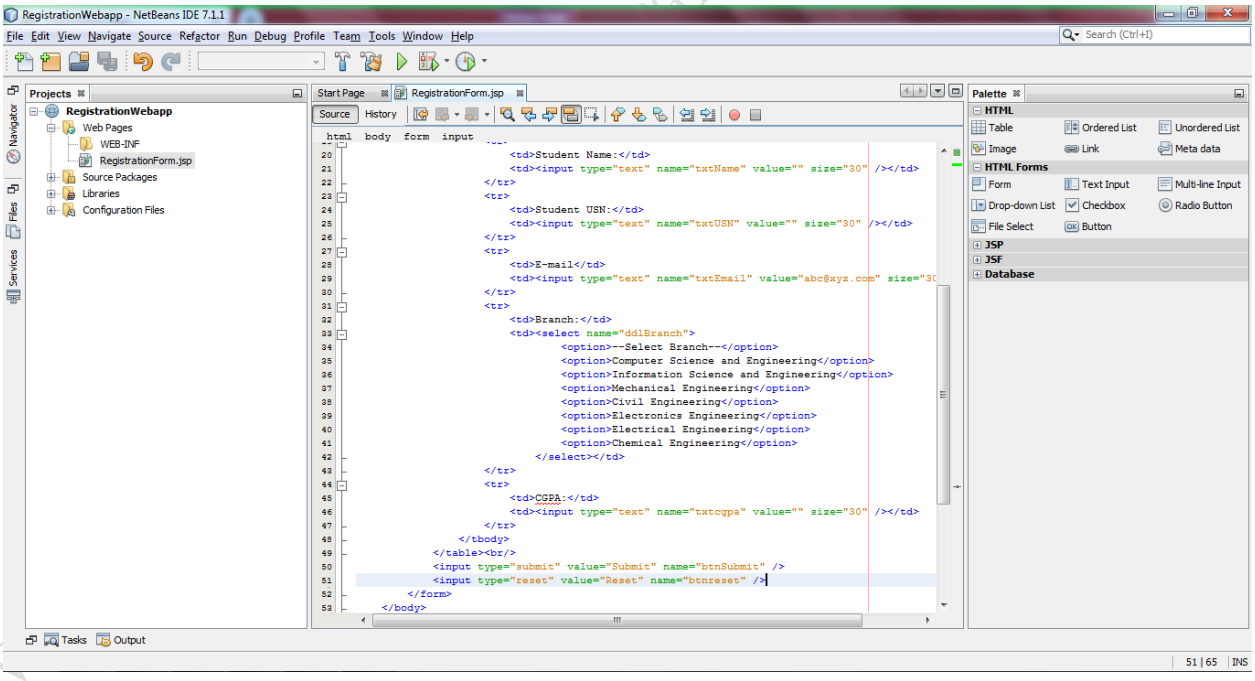
In the below screen shot, we are creating the textbox and assigning name to it. Specifying the type of control and its length. As we are creating textbox, type will be text only as highlighted in the box. After entering all the fields click OK button.



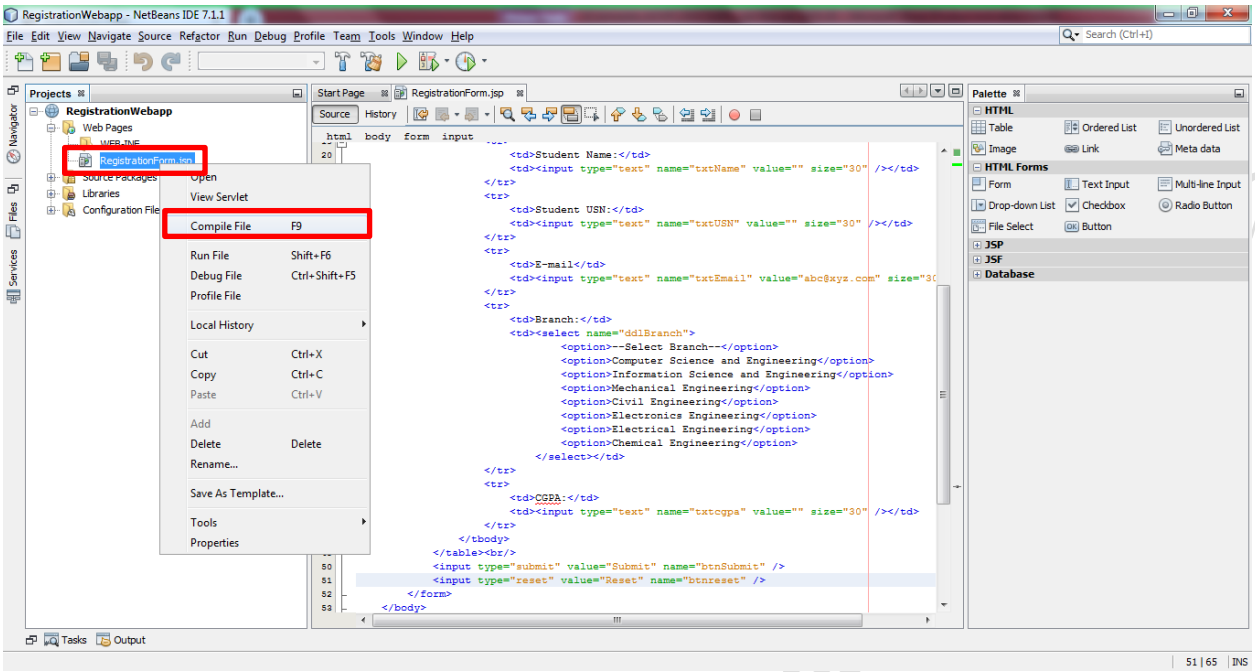
Source code version of the above screen:



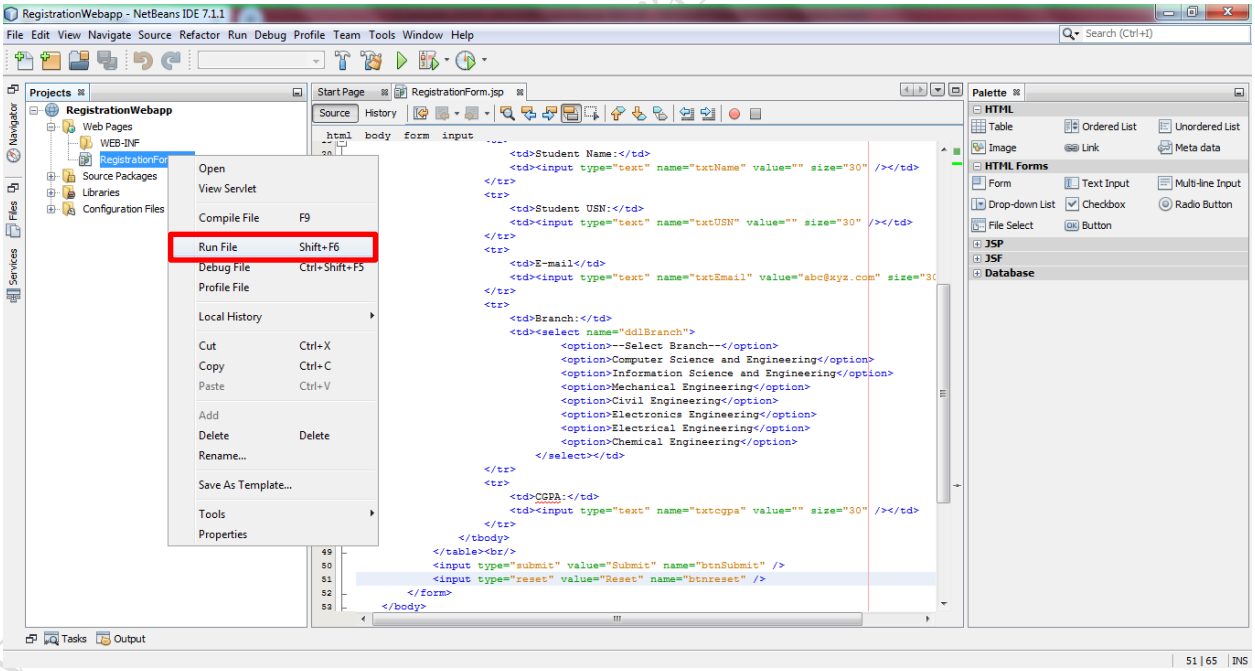
Step-18: After adding all the web controls:



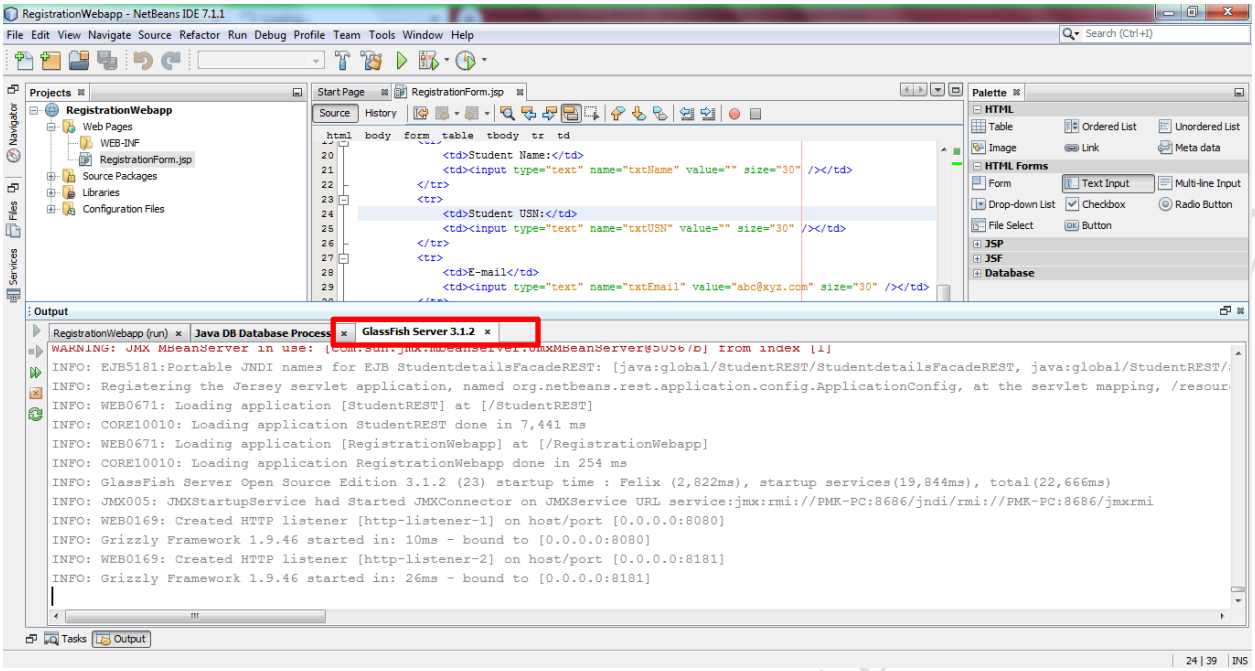
Step-19: Now, we will compile and deploy the RegistrationForm.jsp page to test for any errors.



In the above screen shot .jsp page is compiled, to compile the page go to project select the .jsp page right-click on it→Compile File as shown above.

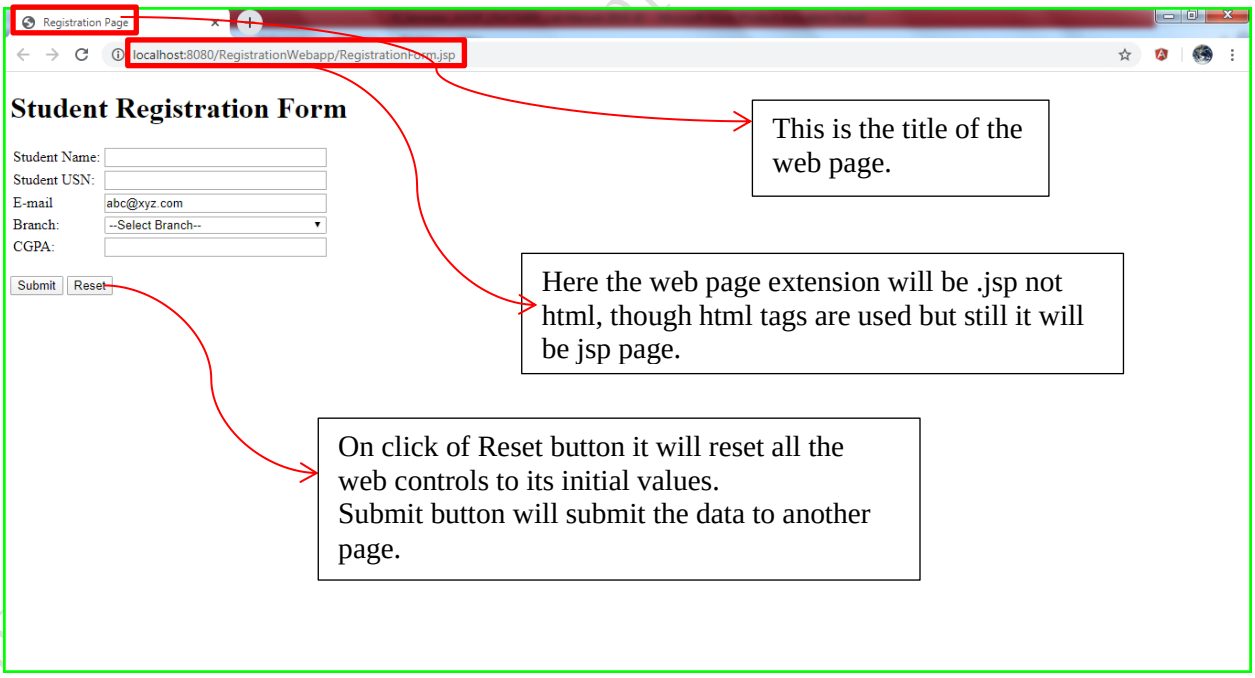


In this screen shot we are deploying the .jsp page on Glassfish server by selecting the Run File option.



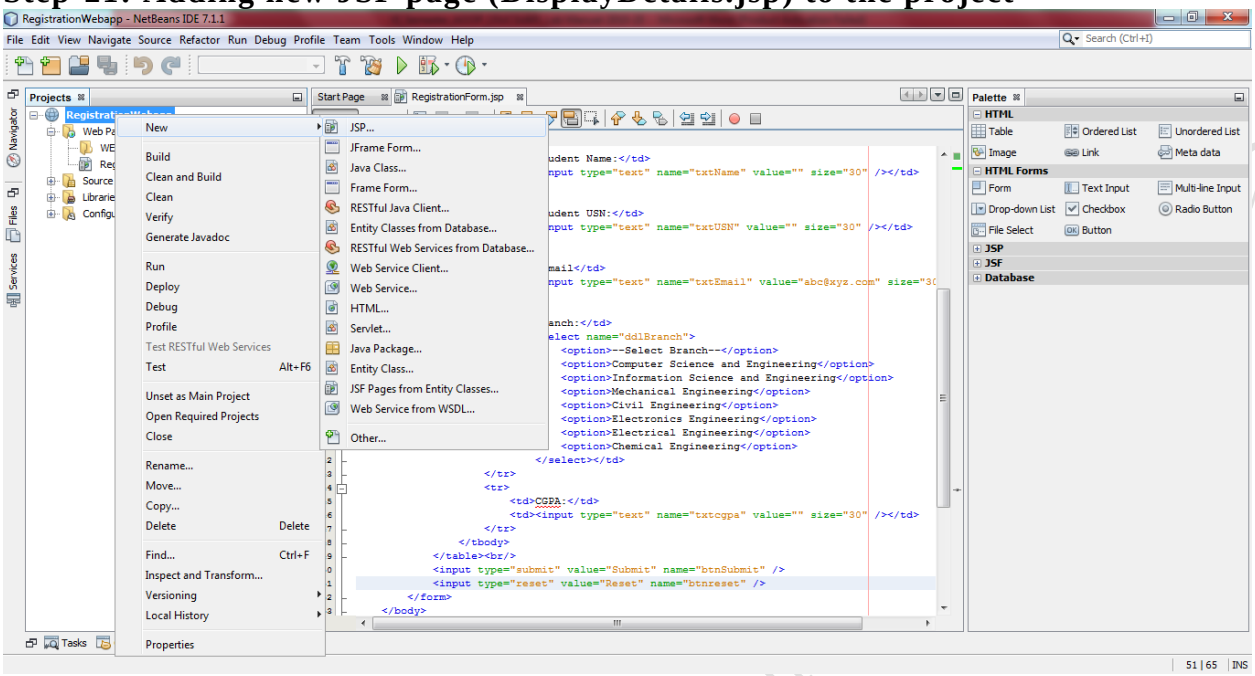
Once we click Run File option, IDE will starts deploying the .jsp page on to the Glassfish server as shown in the above screen shot and opens the web page in the browser as shown below.

To see the page the result, go to the browser where a web page is loaded.

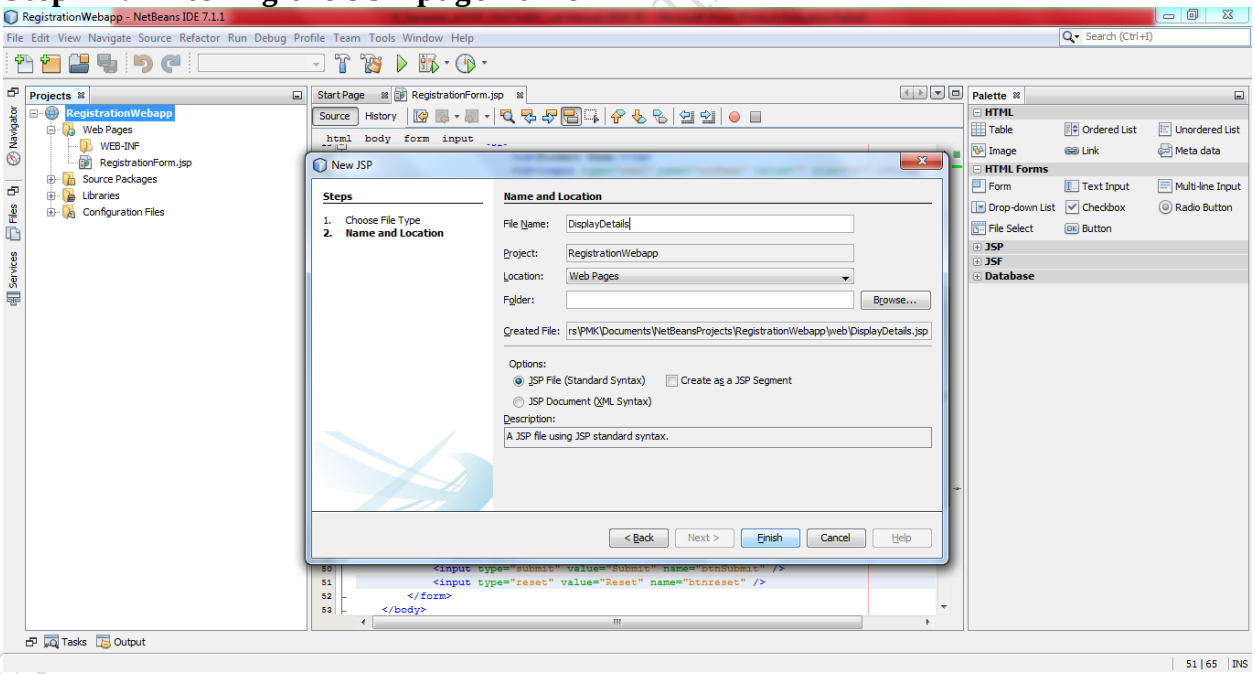


Step-20: Now we will design another JSP page to read and display the data in a tabular format. Here once again we are following the above steps to create new JSP page and add required web controls. This page doesn't require any web controls except table tag because it is given in the requirement that data should be displayed in a tabular format. Let us begin designing the page.

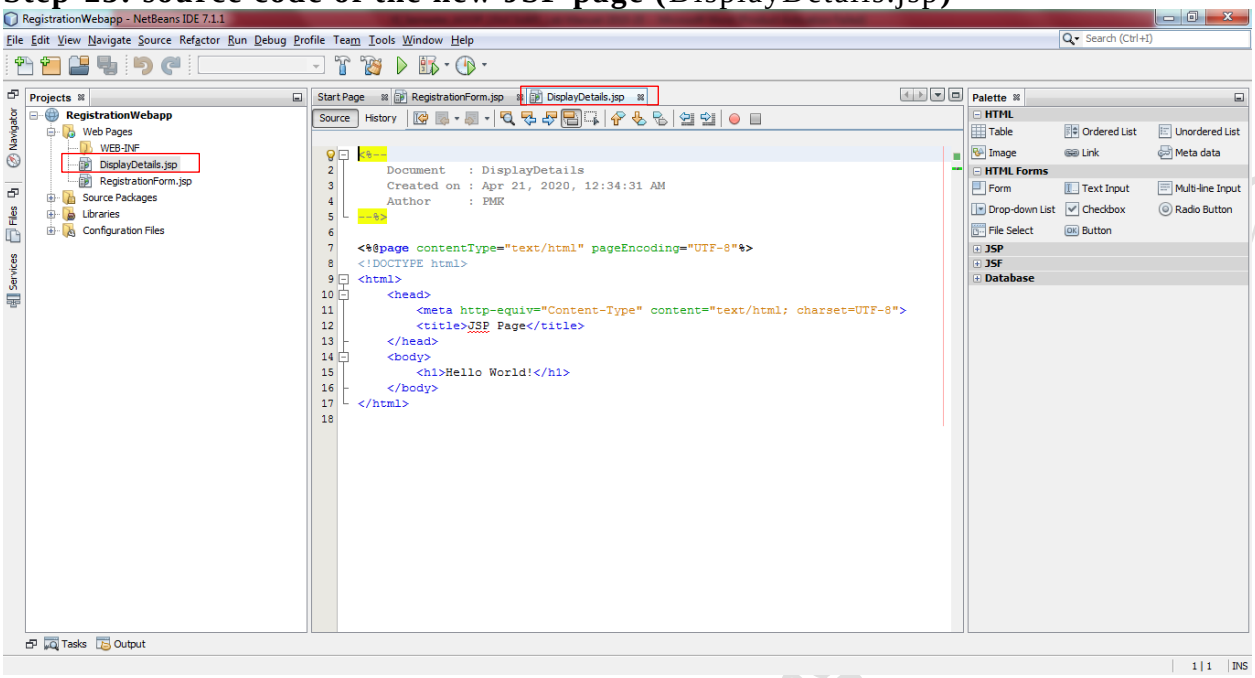
Step-21: Adding new JSP page (DisplayDetails.jsp) to the project



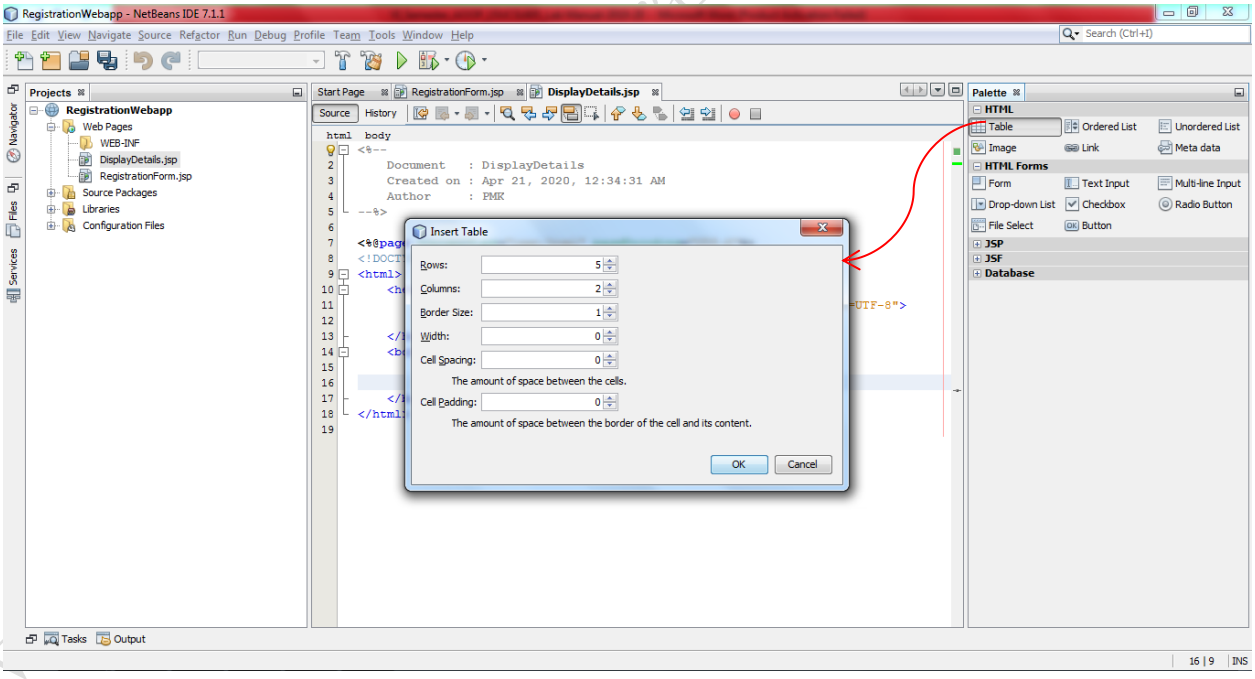
Step-22: Entering the JSP page name



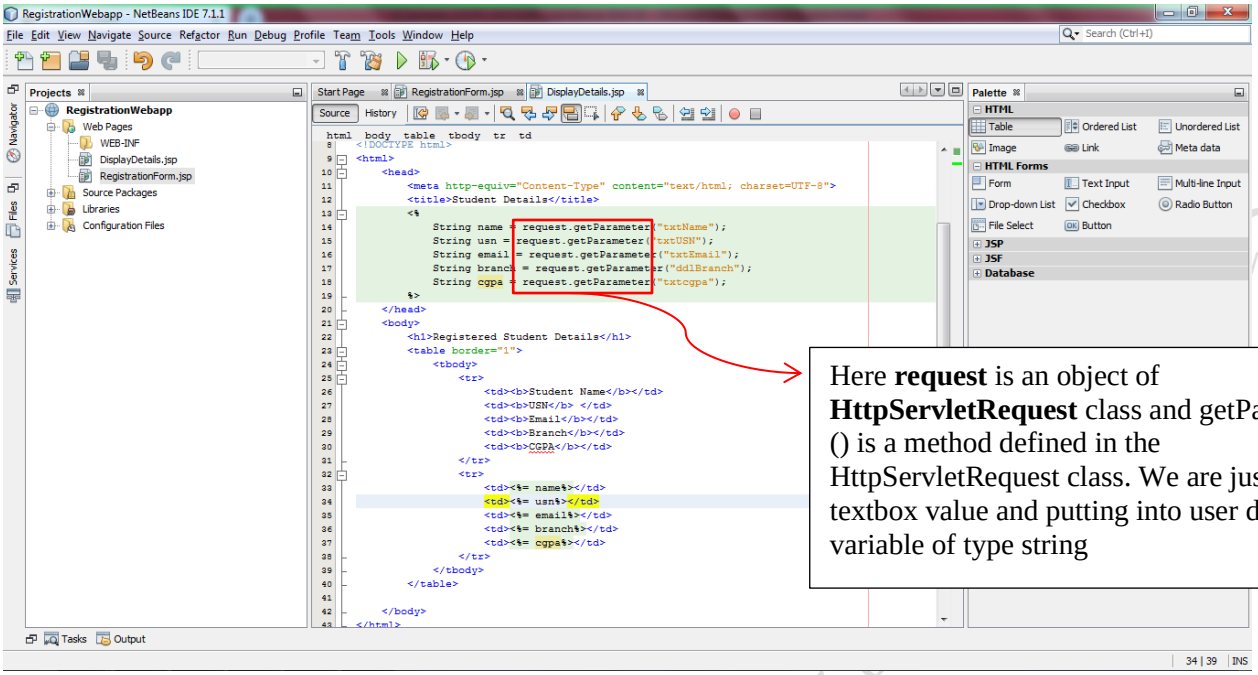
Step-23: source code of the new JSP page (DisplayDetails.jsp**)**



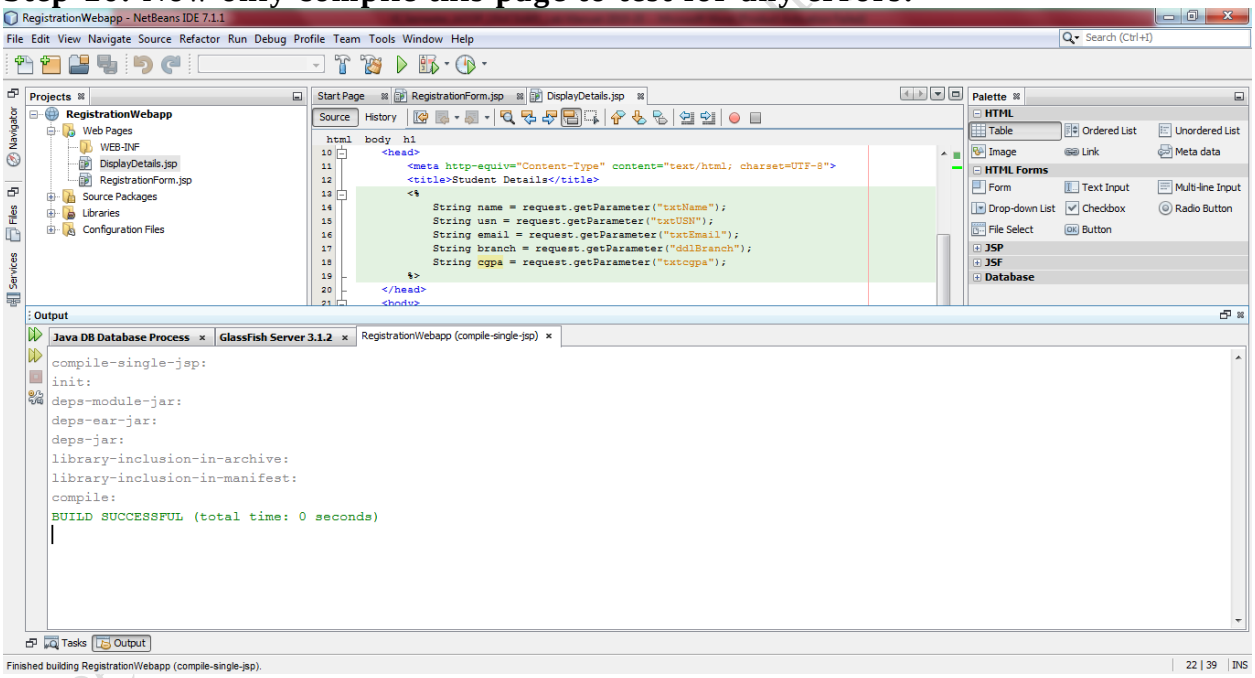
Step-24: Adding form & table tags to the jsp page to display the data in a tabular format:



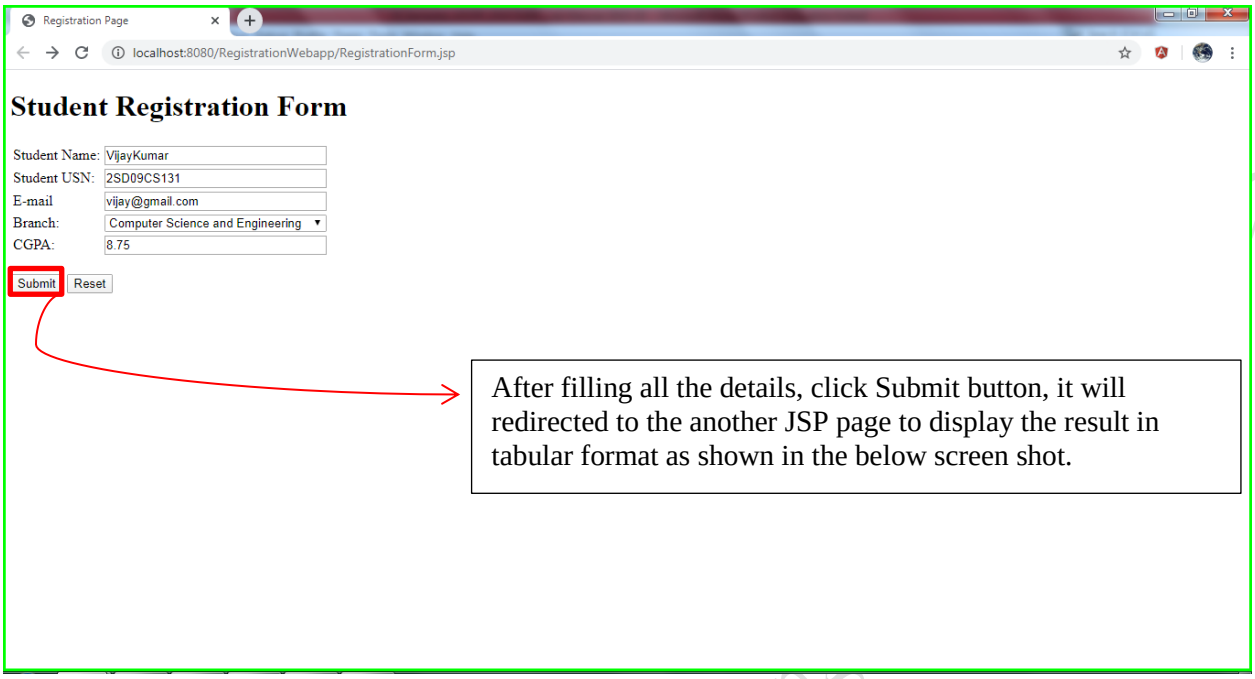
Step-25: Source code of how to read data and display the data in a tabular format:

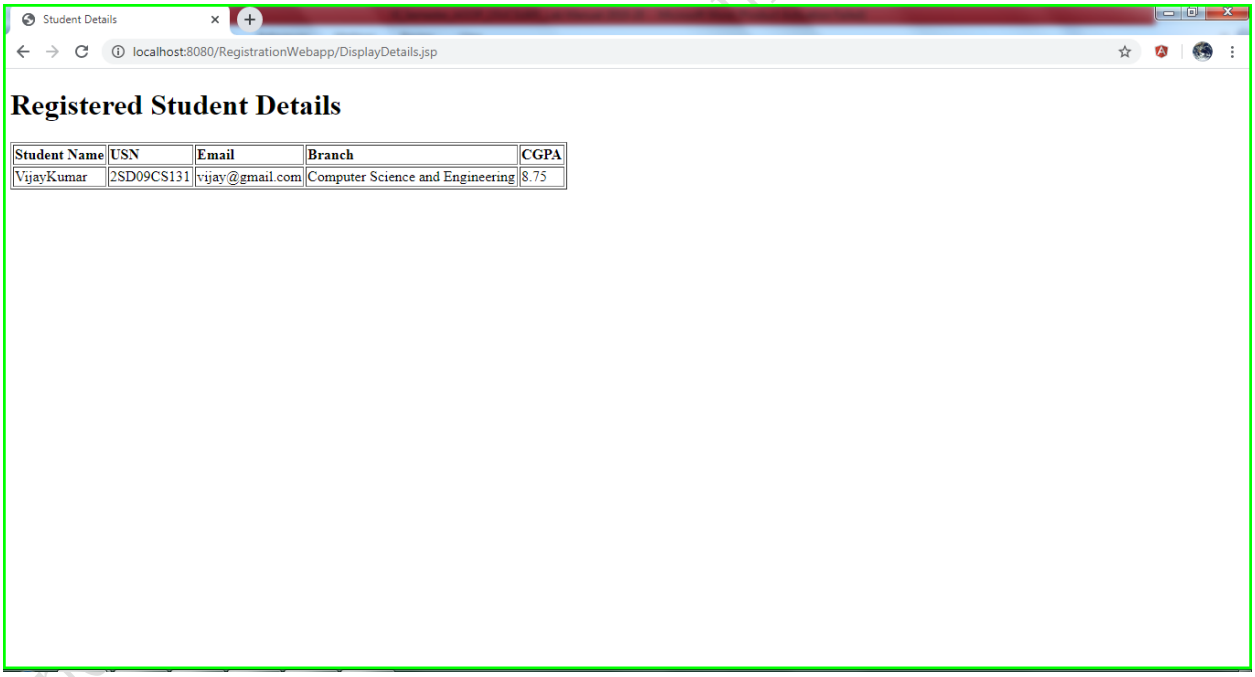


Step-26: Now only compile this page to test for any errors:



Step-27: Now once again we will run the previous page i.e. RegistrationDetails.jsp page, fill all the details and click on submit button, on click of this button it will navigate to another page i.e. DisplayDetails.jsp page which contains details of all entered data in a tabular format.


Registration Page
localhost:8080/RegistrationWebapp/RegistrationForm.jsp
Student Registration Form
Student Name: VijayKumar
Student USN: 2SD09CS131
E-mail vijay@gmail.com
Branch: Computer Science and Engineering
CGPA: 8.75
Submit
Reset
After filling all the details, click Submit button, it will redirected to the another JSP page to display the result in tabular format as shown in the below screen shot.


Student Details
localhost:8080/RegistrationWebapp/DisplayDetails.jsp
Registered Student Details
Student Name
USN
Email
Branch
CGPA
VijayKumar
2SD09CS131
vijay@gmail.com
Computer Science and Engineering
8.75

8. Design and development of web application with database (MySQL) using NetBeans IDE

8.1 Problem Statement: Design and develop dynamic web application for the following requirements:

i. Design the web page to collect the following student information using JSP page (it can be done using HTML also).

- Student name
- Student USN
- Program Type
- Awards
- Branch

ii. Use MySQL database to test the DDL and DML statements.

iii. Implement only the CRUD operations.

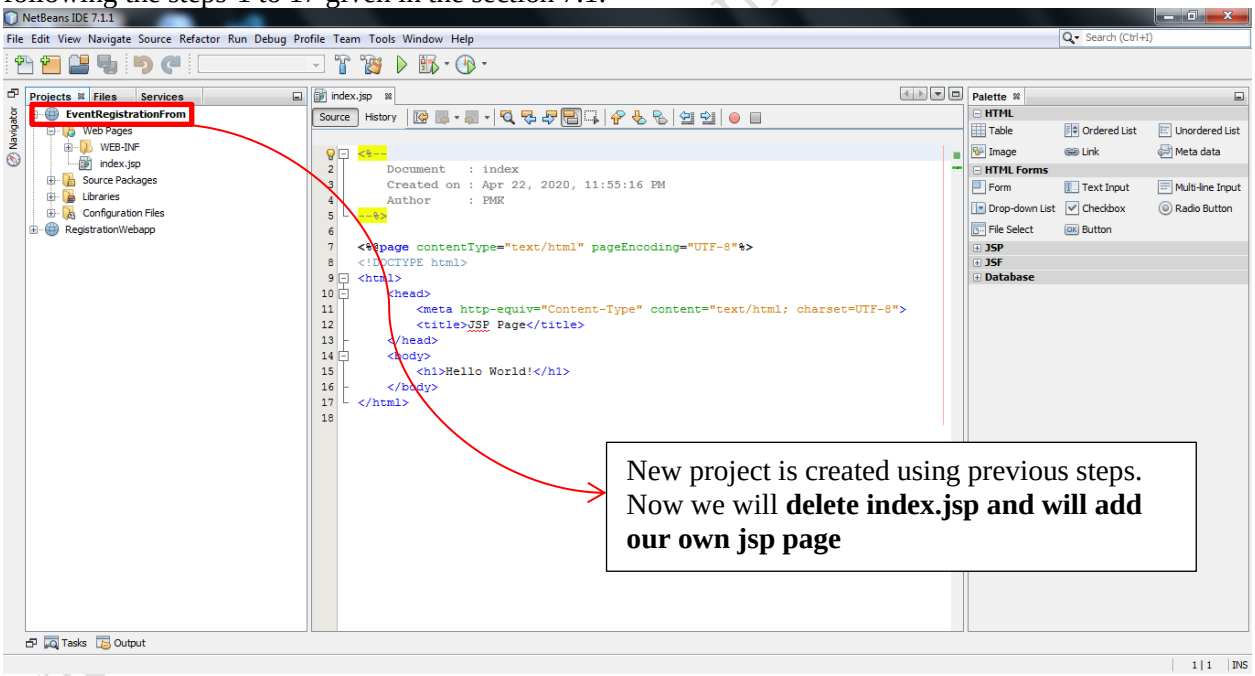
iv. Use different web page to display the table values. Data must be shown in a gird format with proper headings.

Step1: Launch the NetBeans IDE:

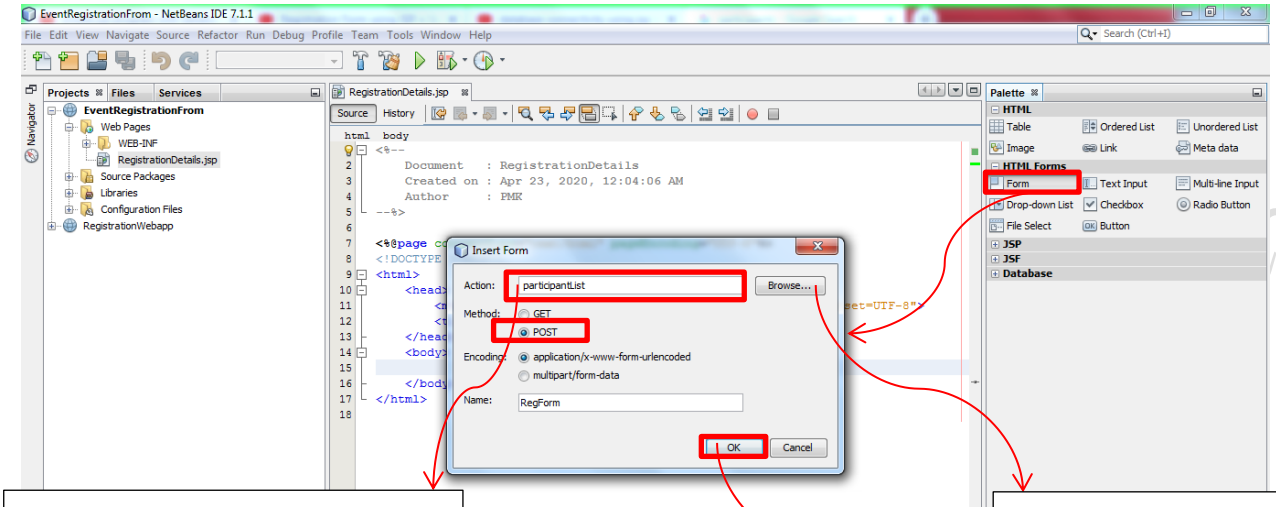
Refer the section 6.1 and the **Fig(s): 26 to 29** to launch and open the NetBeans IDE.

Step2: To create a new project and adding a web page refer steps-1 to 17 of section 7.1.

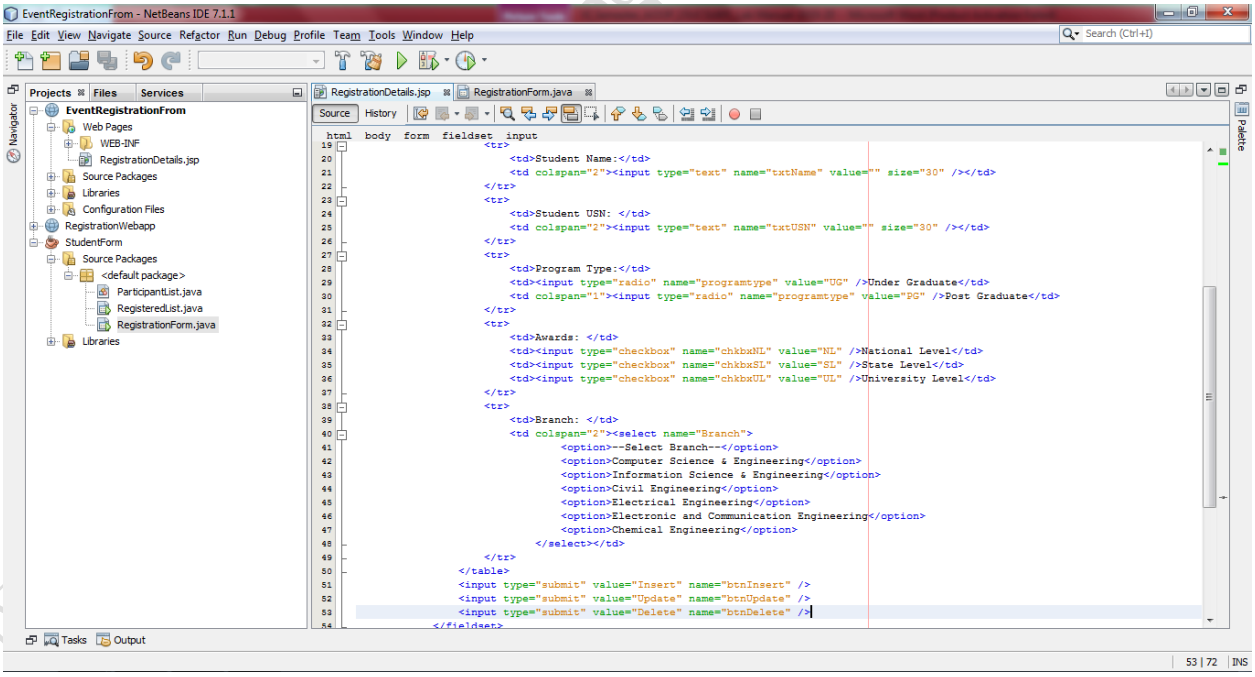
Step3: Here directly we will start with adding web controls to the jsp page (Designing user interface) by following the steps-1 to 17 given in the section 7.1:



Step4: In this step we are creating the form tag with the help of IDE Palette window. Here action attribute value is assigned and mode of submission i.e. using GET or POST methods. Similarly rest of the web controls are added to the JSP page.

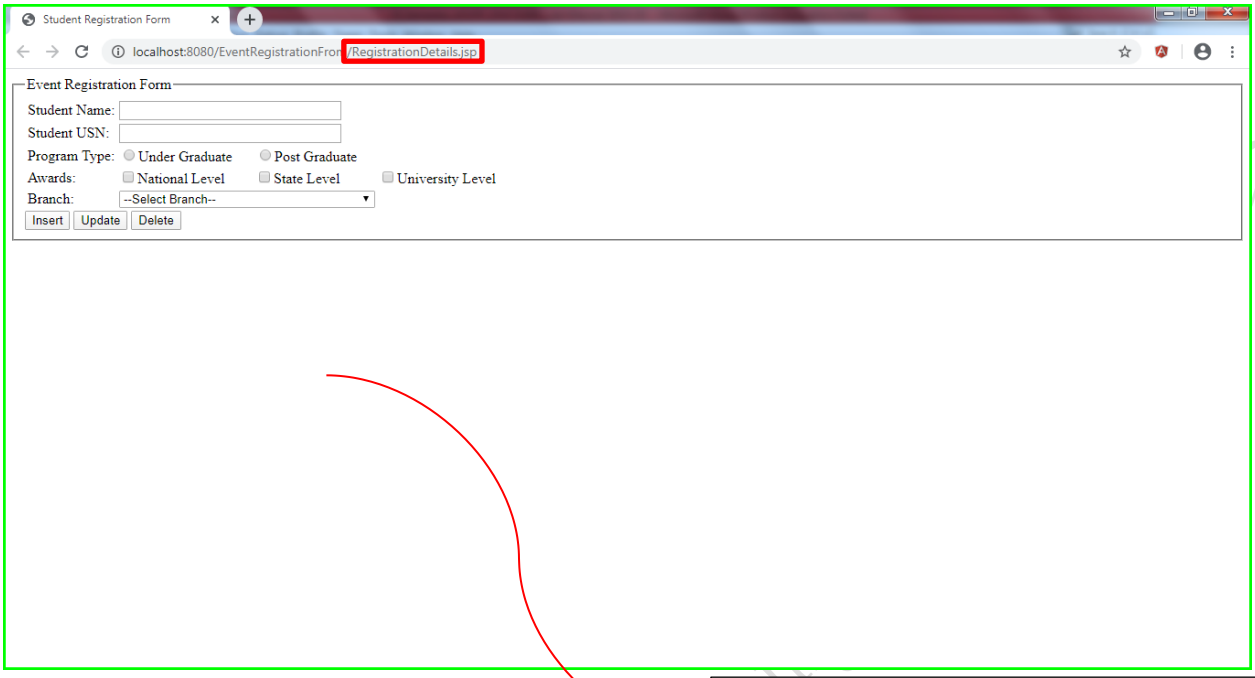


Step5: Source code User interface



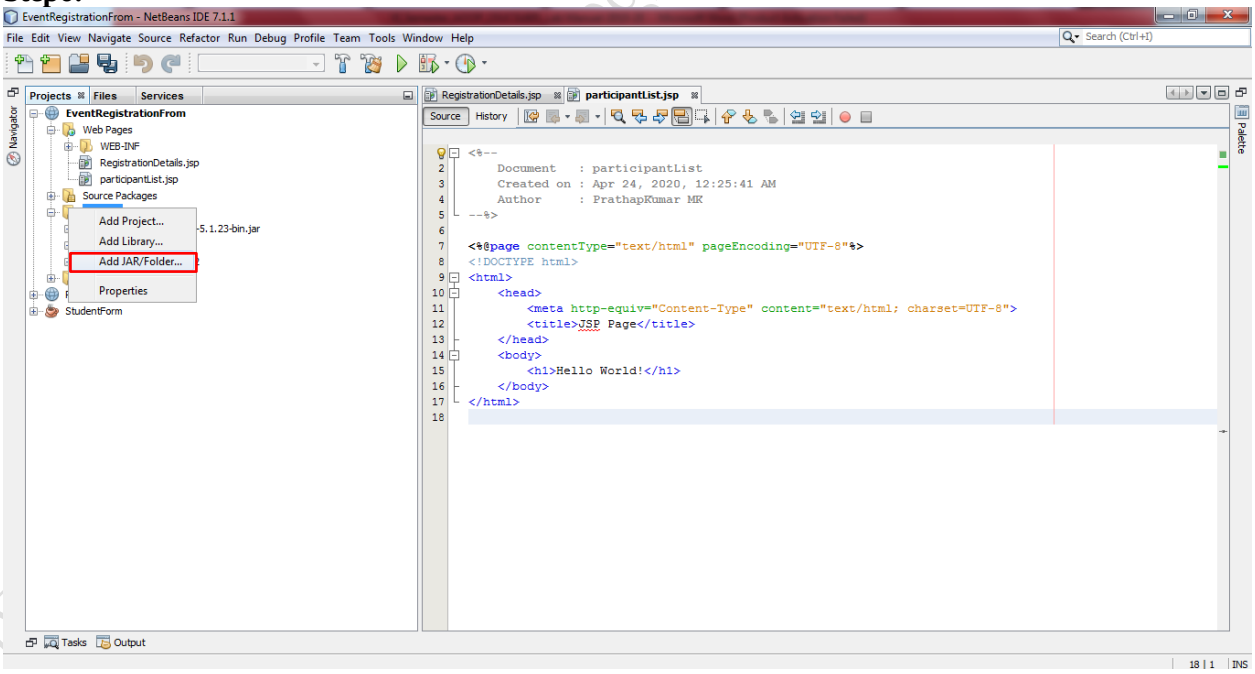
Now to view the user interface, we will just compile and deploy it onto the server. Refer the section 7.1 of step-19 to perform above step.

We not yet completed the implementation of given requirement. It will be continued after checking UI alignment.

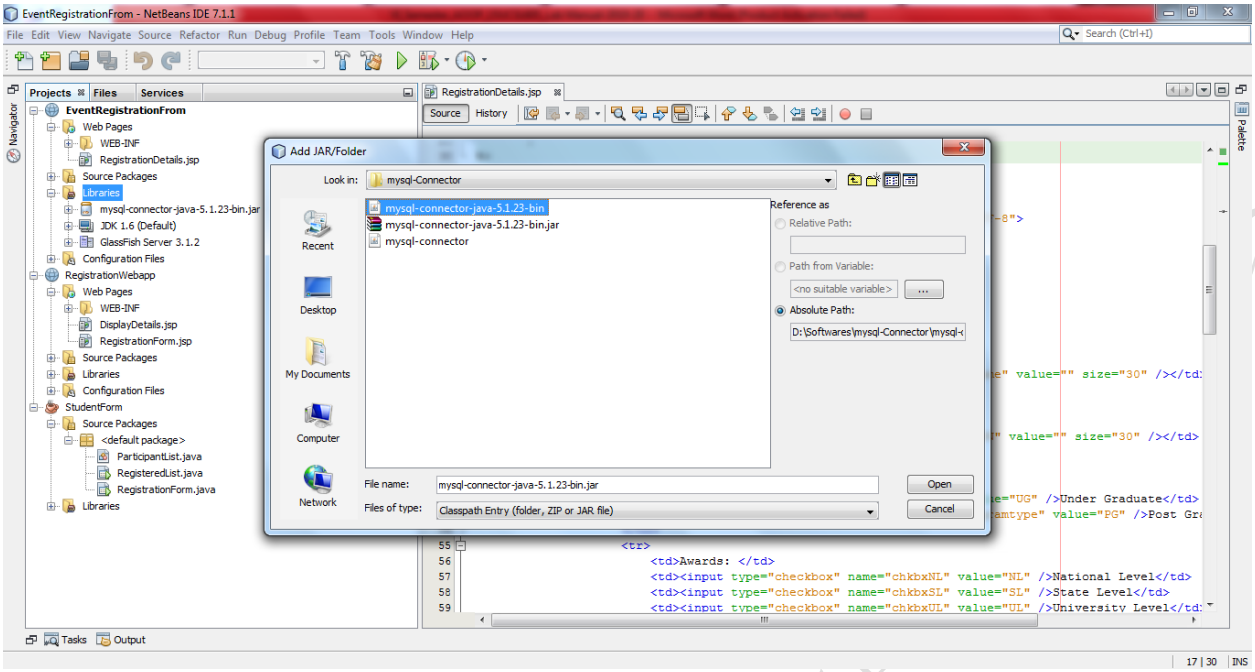


Web page with all required input controls.

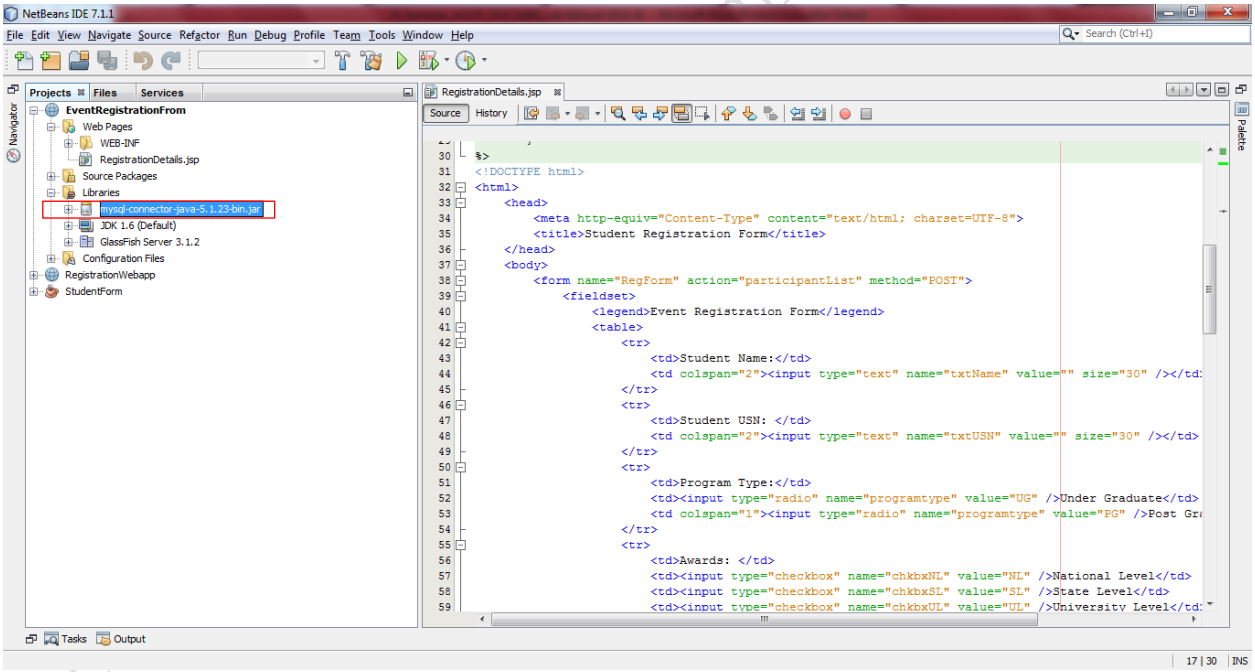
Step6:



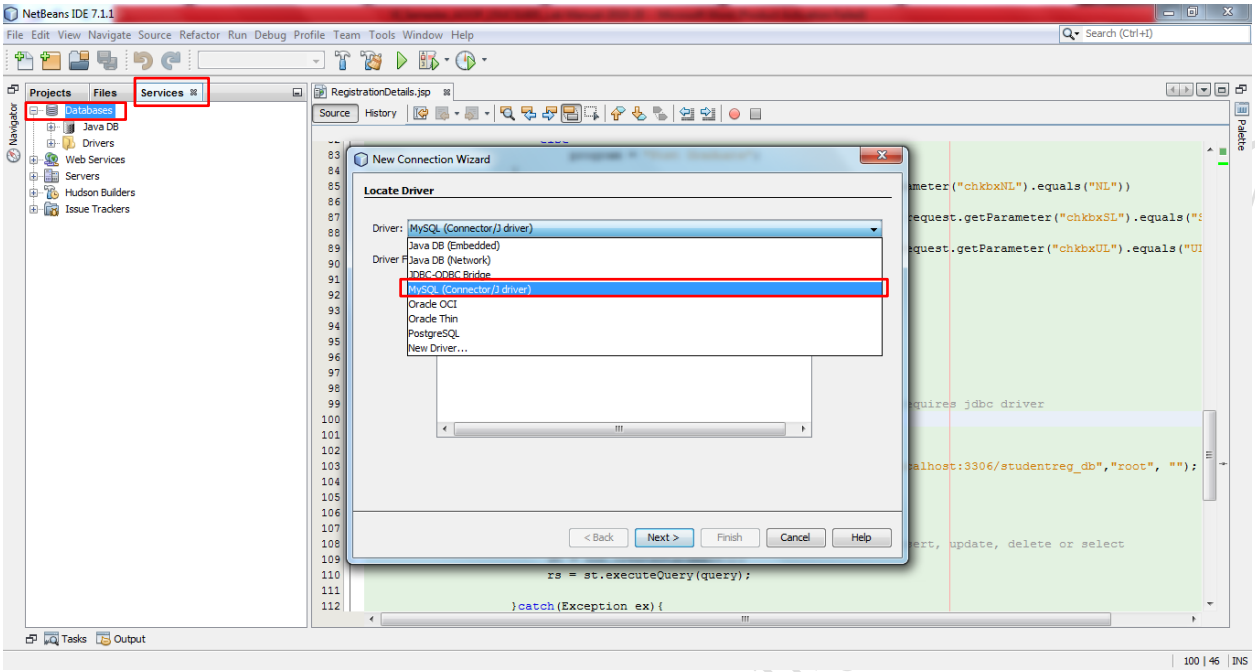
Step7:


EventRegistrationFrom - NetBeans IDE 7.1.1
Add JAR/Folder
Look in: mysql-Connector
mysql-connector-java-5.1.23-bin
mysql-connector-java-5.1.23-bin.jar
mysql-connector
Recent
Desktop
My Documents
Computer
Network
Reference as
Relative Path:
Path from Variable:
Absolute Path:
File name: mysql-connector-java-5.1.23-bin.jar
Files of type: Classpath Entry (folder, ZIP or JAR file)
Open
Cancel

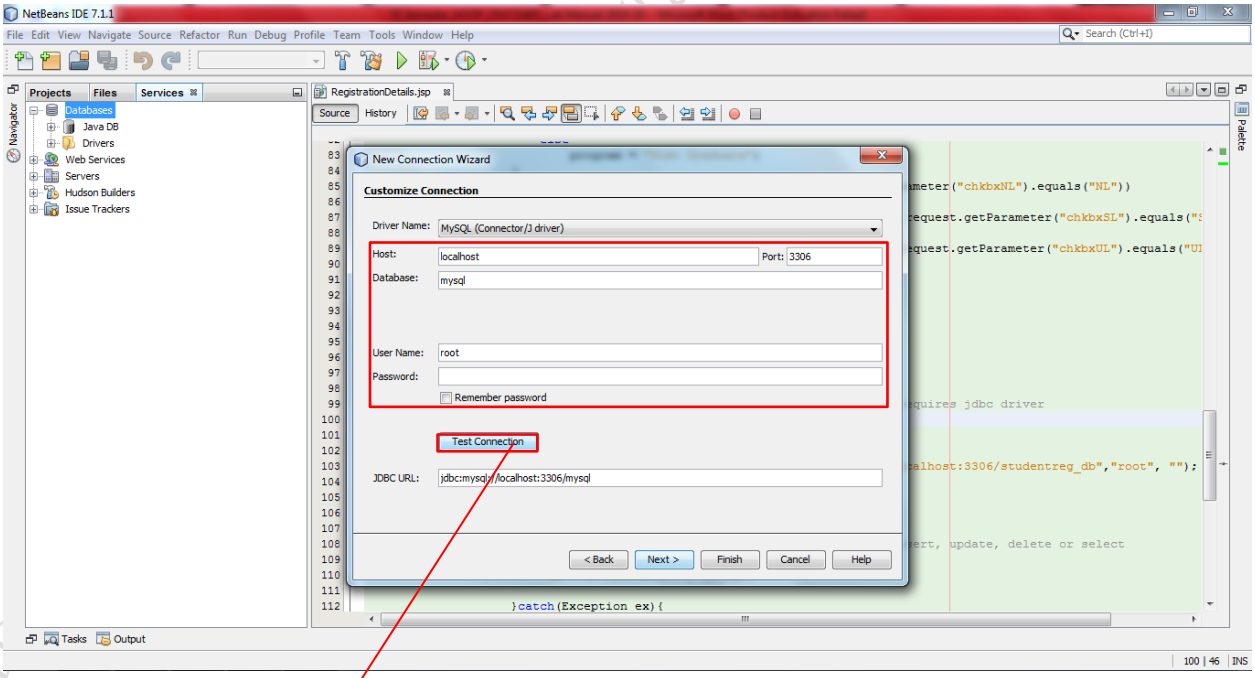
Step7:


NetBeans IDE 7.1.1
mysql-connector-java-5.1.23-bin.jar

Step8:

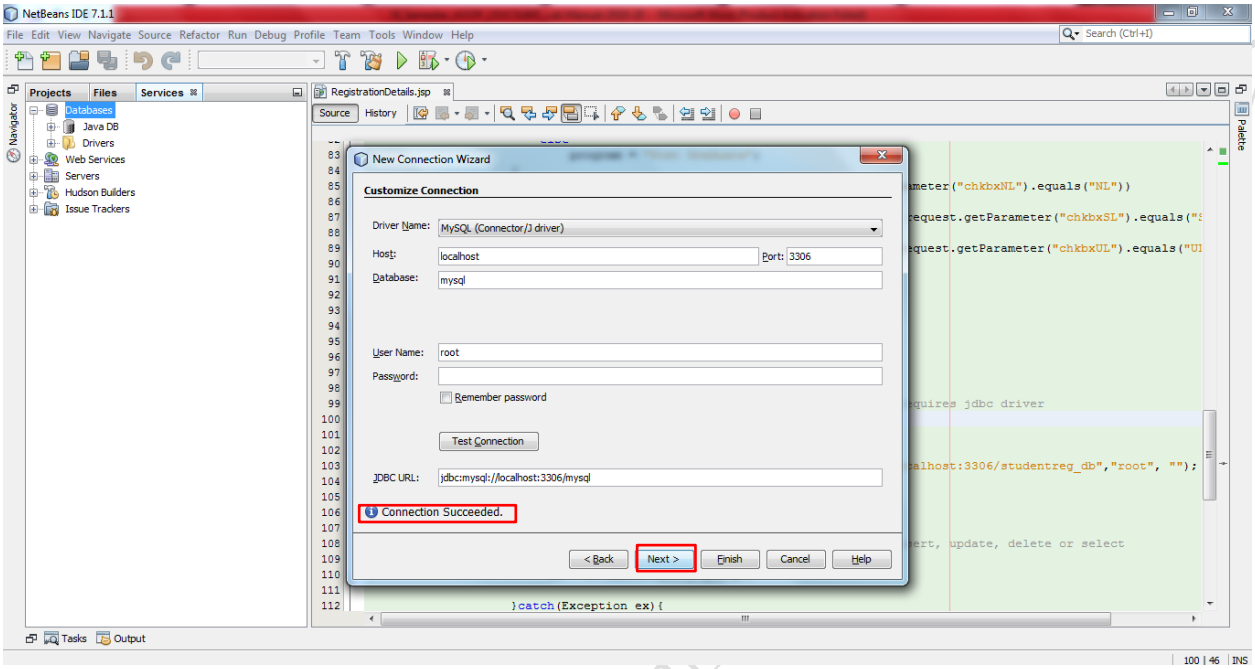


Step9:

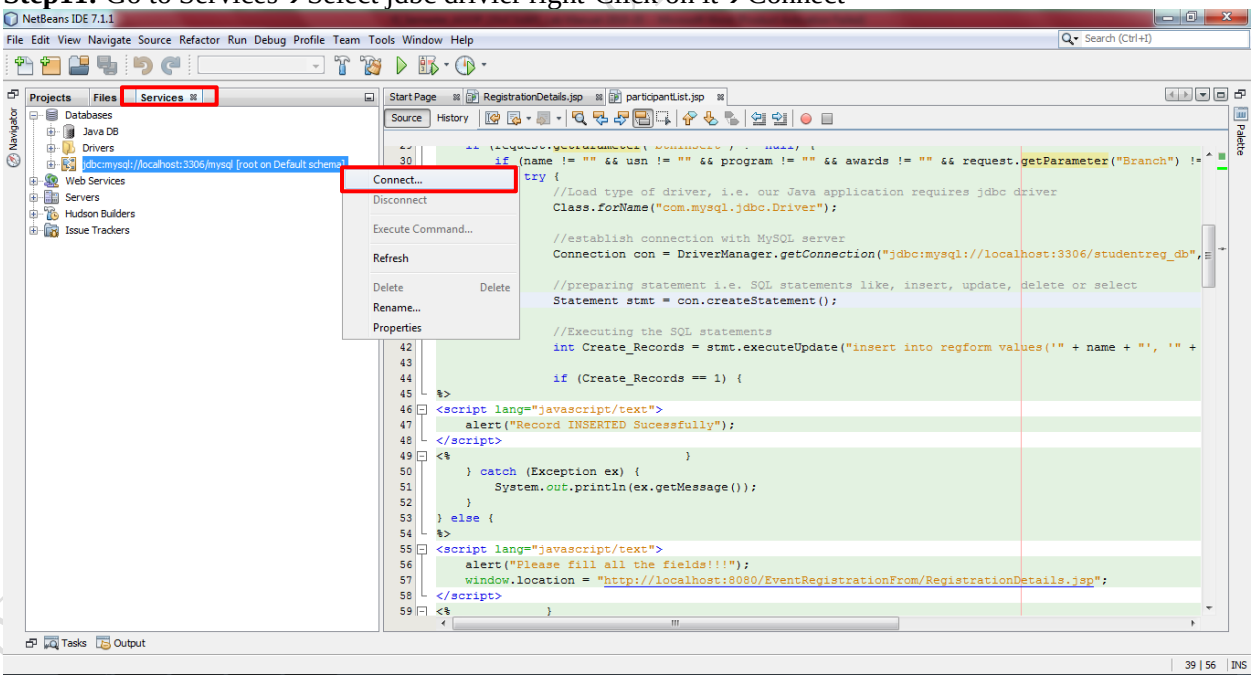


Note: Before performing the database connectivity part, make sure that Wamp server is up and running in the system. After this activity following steps can be verified.

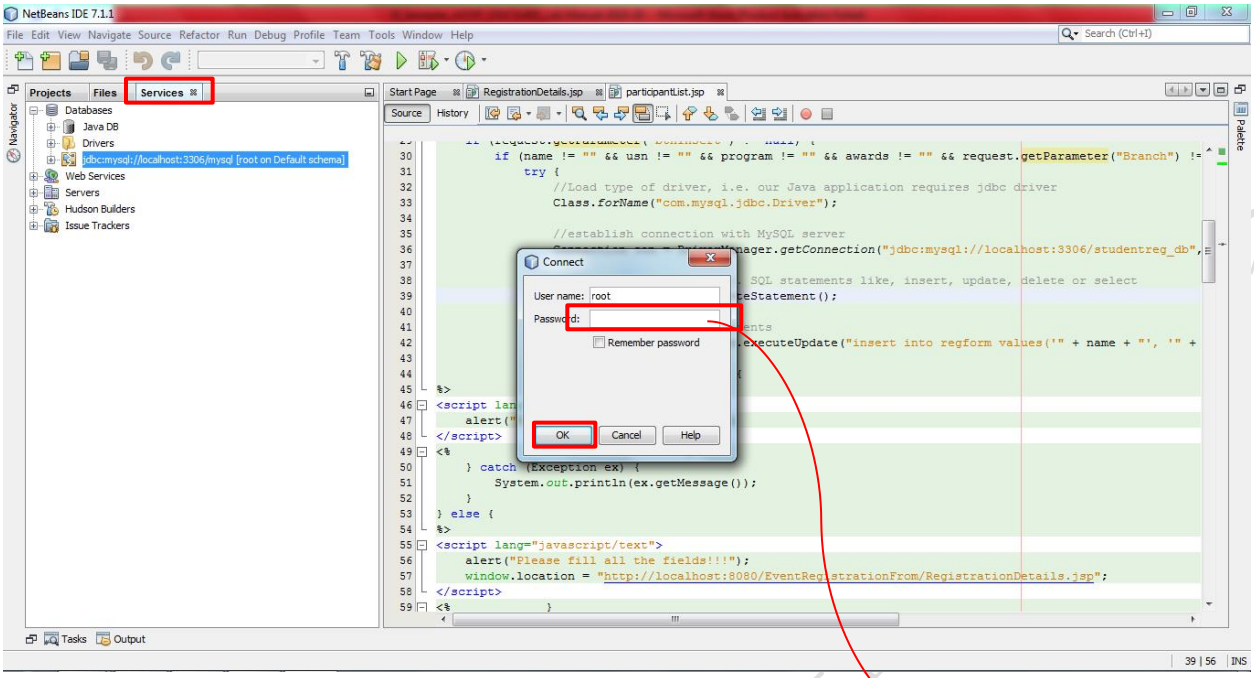
Step10:



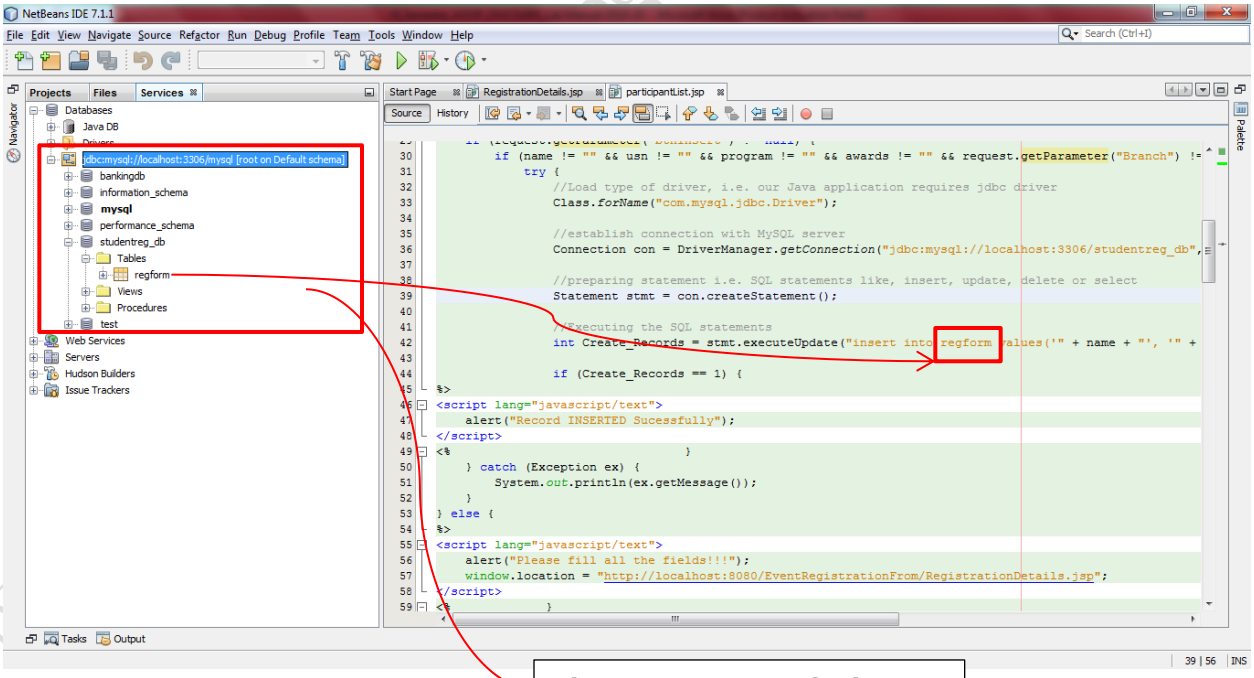
Step11: Go to Services→Select jdbc drivier right-Click on it→Connect



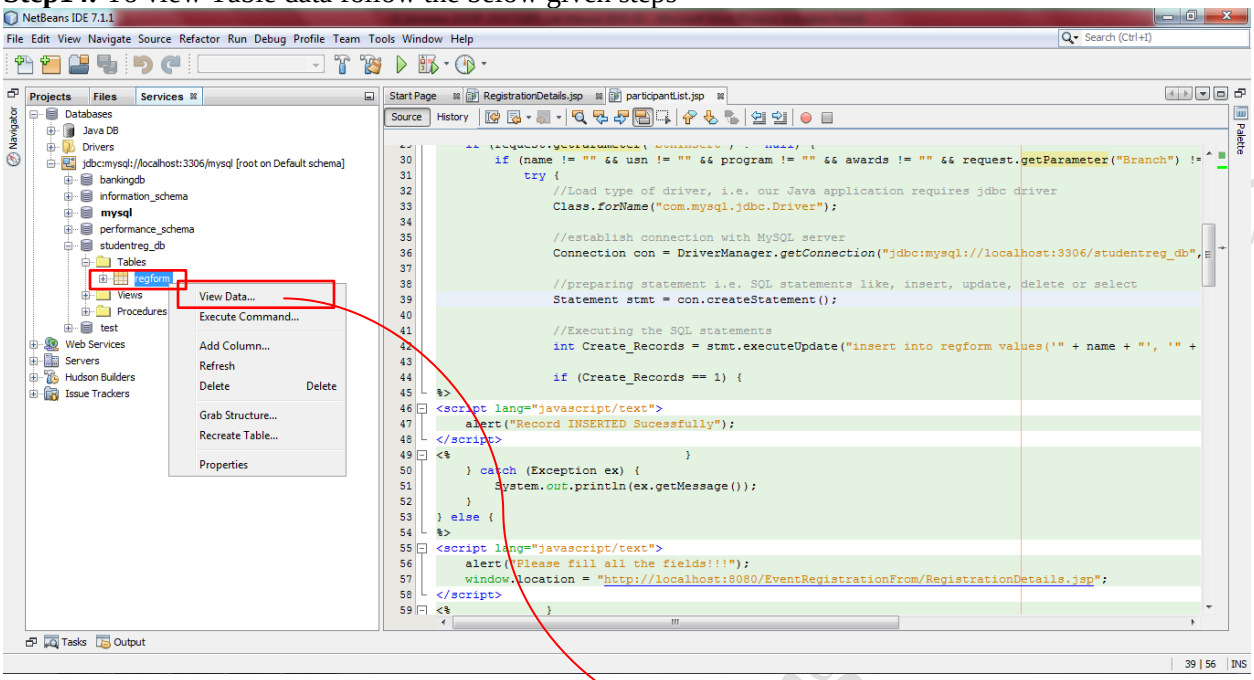
Step12: To connect to the MySQL server, enter username and password as shown below.



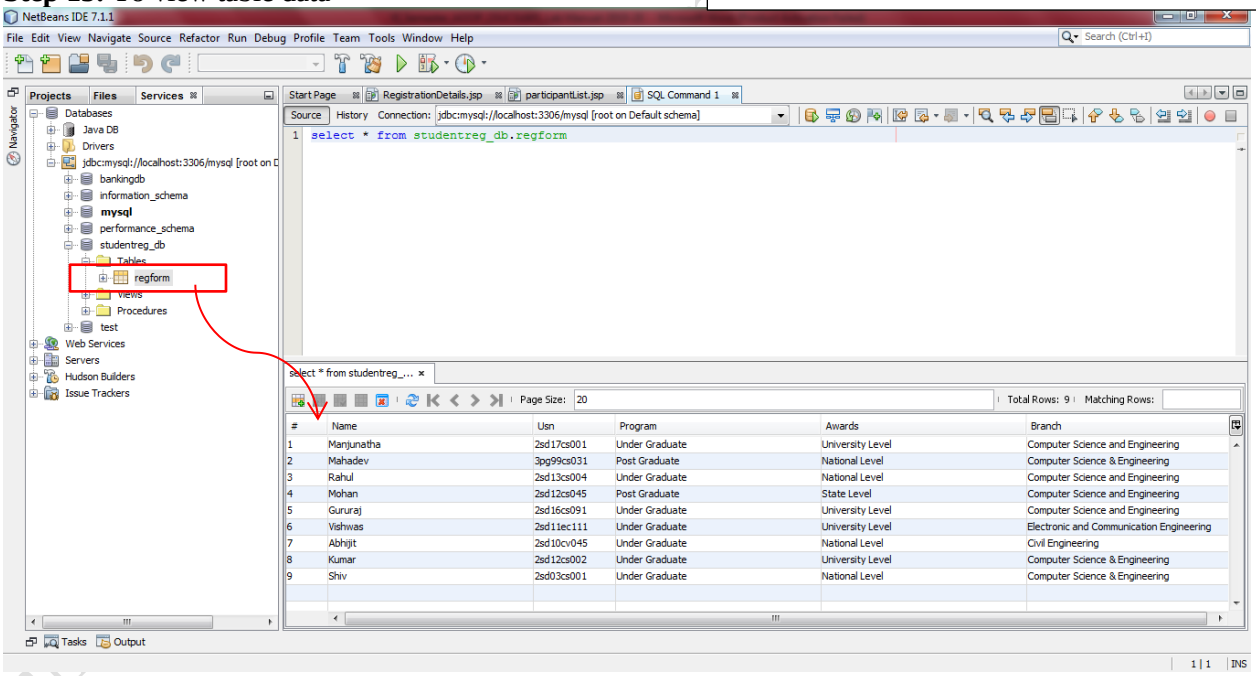
Step13: Now MySQL server databases and its tables are populated into NetBeans IDE.



Step14: To view Table data follow the below given steps

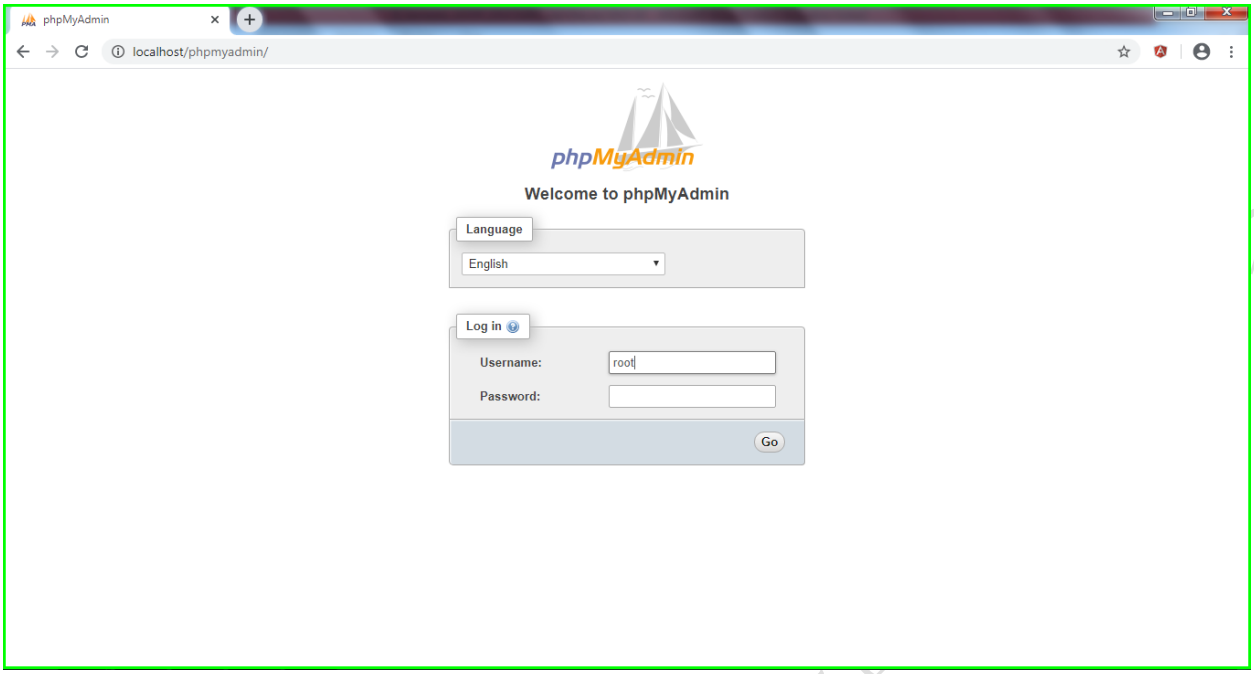


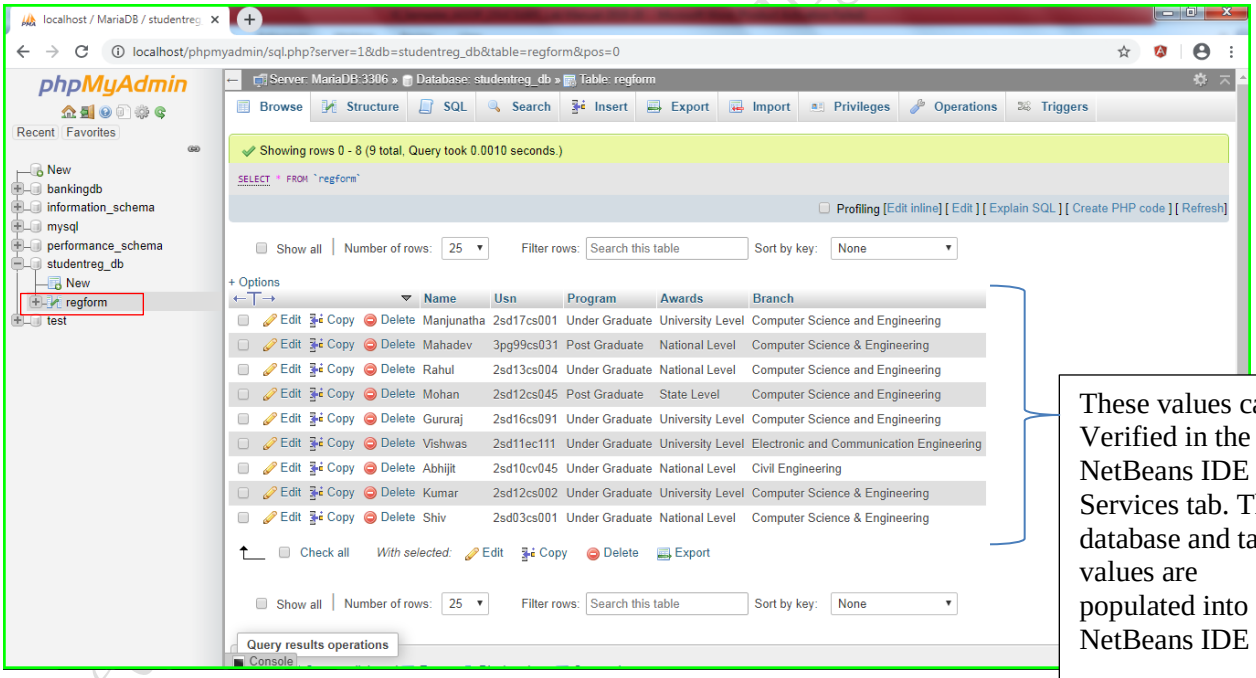
Step 15: To view table data



#	Name	Usn	Program	Awards	Branch
1	Manjunatha	2sd17cs001	Under Graduate	University Level	Computer Science and Engineering
2	Mahadev	3pg99cs031	Post Graduate	National Level	Computer Science & Engineering
3	Rahul	2sd13cs004	Under Graduate	National Level	Computer Science and Engineering
4	Mohan	2sd12cs045	Post Graduate	State Level	Computer Science and Engineering
5	Gururaj	2sd16cs091	Under Graduate	University Level	Computer Science and Engineering
6	Vishwas	2sd11ec111	Under Graduate	University Level	Electronic and Communication Engineering
7	Abhijit	2sd10cv045	Under Graduate	National Level	Civil Engineering
8	Kumar	2sd12cs002	Under Graduate	University Level	Computer Science & Engineering
9	Shiv	2sd03cs001	Under Graduate	National Level	Computer Science & Engineering

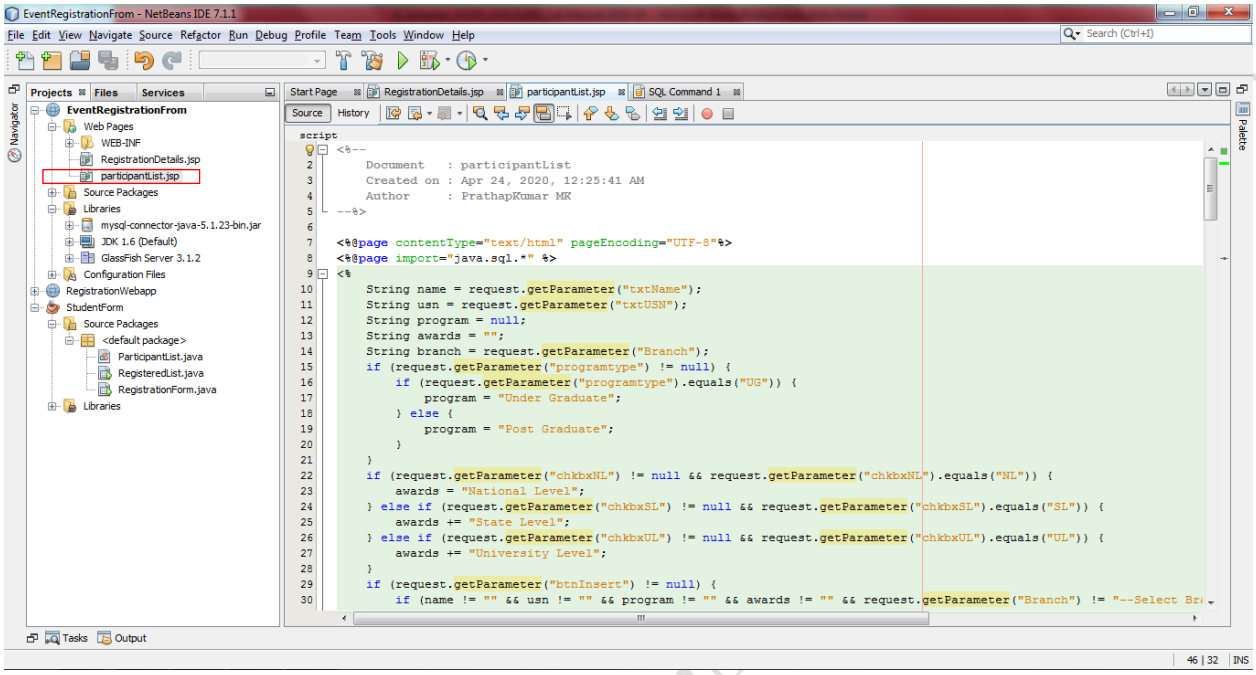
Above database and its tables are fetched from the MySQL server. Now we will cross verify the same using MySQL Server.





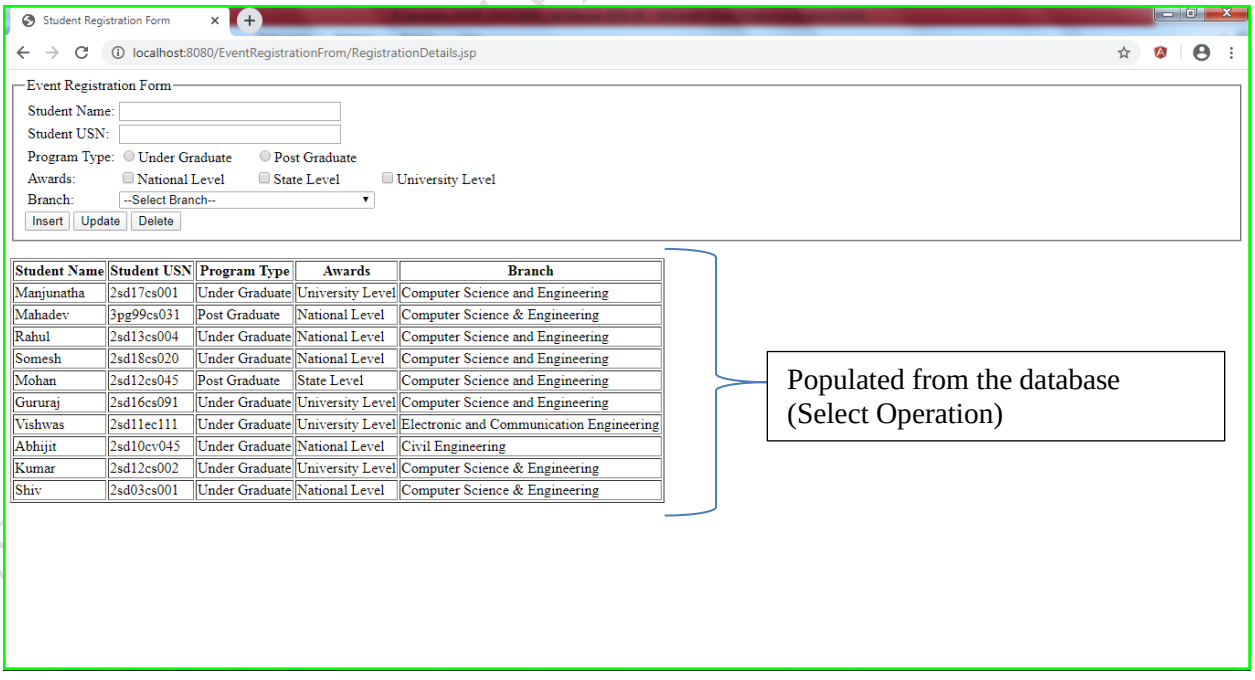
These values can be Verified in the NetBeans IDE of Services tab. These database and table values are populated into NetBeans IDE

Step16:
Add new JSP page and name it as "participantList.jsp" / give the page name as given in the action attribute of first JSP page. Go to Step4 to confirm the name of the JSP page.




```
<%--
    Document   : participantList
    Created on : Apr 24, 2020, 12:25:41 AM
    Author     : PrathapKumar MK
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%
    String name = request.getParameter("txtName");
    String usn = request.getParameter("txtUSN");
    String program = null;
    String awards = "";
    String branch = request.getParameter("Branch");
    if (request.getParameter("programtype") != null) {
        if (request.getParameter("programtype").equals("UG")) {
            program = "Under Graduate";
        } else {
            program = "Post Graduate";
        }
    }
    if (request.getParameter("chkbxNL") != null && request.getParameter("chkbxNL").equals("NL")) {
        awards = "National Level";
    } else if (request.getParameter("chkbxSL") != null && request.getParameter("chkbxSL").equals("SL")) {
        awards += "State Level";
    } else if (request.getParameter("chkbxUL") != null && request.getParameter("chkbxUL").equals("UL")) {
        awards += "University Level";
    }
    if (request.getParameter("btnInsert") != null) {
        if (name != "" && usn != "" && program != "" && awards != "" && request.getParameter("Branch") != "--Select Br
```

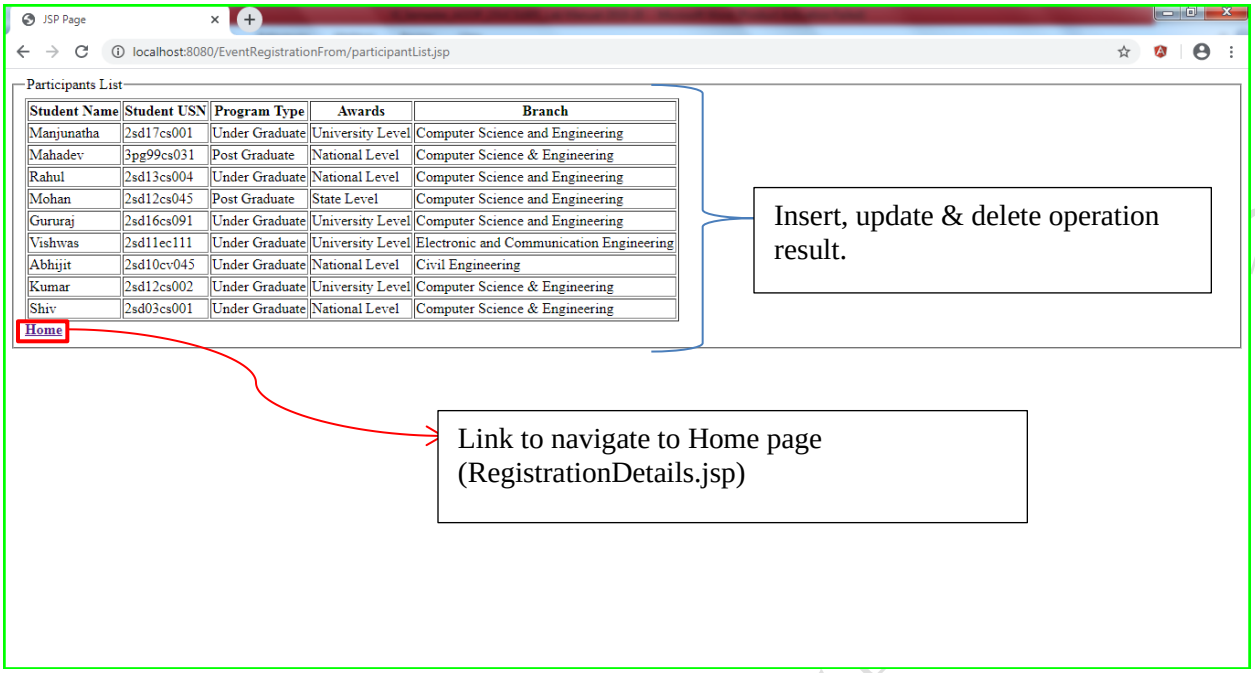
Step17: RegistrationDetails.jsp page to read user input and populate the table values.



Student Name	Student USN	Program Type	Awards	Branch
Manjunatha	2sd17cs001	Under Graduate	University Level	Computer Science and Engineering
Mahadev	3pg99cs031	Post Graduate	National Level	Computer Science & Engineering
Rahul	2sd13cs004	Under Graduate	National Level	Computer Science and Engineering
Somesh	2sd18cs020	Under Graduate	National Level	Computer Science and Engineering
Mohan	2sd12cs045	Post Graduate	State Level	Computer Science and Engineering
Gururaj	2sd16cs091	Under Graduate	University Level	Computer Science and Engineering
Vishwas	2sd11ec111	Under Graduate	University Level	Electronic and Communication Engineering
Abhijit	2sd10cv045	Under Graduate	National Level	Civil Engineering
Kumar	2sd12cs002	Under Graduate	University Level	Computer Science & Engineering
Shiv	2sd03cs001	Under Graduate	National Level	Computer Science & Engineering

Populated from the database (Select Operation)

Step18: ParticipantList.jsp page to display the updated values from the database. This page is loaded whenever Insert, Update & Delete operations are performed.



8.2 Problem Statement given in the section 8.1 is implemented using JavaScript and AJAX

Following is the source code for the given requirement. To design user interface, refer the steps given in the 8.1 section.

HomePage.jsp file:

```
<%--
    Document    : HomePage
    Author      : PrathapKumar MK
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%
    // Set refresh, autoload time as 5 seconds
    //response.setIntHeader("Refresh", 30);
%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Student Registration Form</title>

    </head>
    <body>
        <form name="RegForm">
            <fieldset>
                <legend>Event Registration Form</legend>
                <table>
                    <tr>
```

```
                    <td>Student Name:</td>
                    <td colspan="2"><input type="text"
                      id="txtName" size="30" /></td>
                </tr>
                <tr>
                    <td>Student USN: </td>
                    <td colspan="2"><input type="text"
                        id="txtUSN" size="30" /></td>
                </tr>
                <tr>
                    <td>Program Type:</td>
                    <td><input type="radio" id="UG"
                     name="programtype" value="UG" />Under
                        Graduate</td>
                    <td colspan="1"><input type="radio" id
                      ="PG" name="programtype" value="PG"
                        />Post Graduate</td>
                </tr>
                <tr>
                    <td>Awards: </td>
                    <td><input type="checkbox" name="awards"
                       value="NL" />National Level</td>
                    <td><input type="checkbox" name="awards"
                         value="SL" />State Level</td>
                    <td><input type="checkbox" name="awards"
                        value="UL" />University Level</td>
                </tr>
                <tr>
                    <td>Branch: </td>
                    <td colspan="2">
                        <select id="Branch" >
                          <option>--Select Branch--</option>
                          <option>Computer Science and
                                        Engineering</option>
                            <option>Information Science and
                                    Engineering</option>
                          <option>Civil Engineering</option>
                          <option>Electrical
                              Engineering</option>
                            <option>Electronic and
                                    Communication
                              Engineering</option>
                            <option>Chemical
                              Engineering</option>
                        </select>
                    </td>
                </tr>
            </table>
                <input type="button" value="Insert" id
                     ="btnInsert" name="Insert"
                      onclick="navigate(this.id);"/>
                <input type="button" value="Update" id
="btnUpdate" name="Update" onclick="navigate(this.id);"/>
```

```
                    <input type="button" value="Delete" id
="btnDelete" name="Delete" onclick="navigate(this.id);"/>
                <span id="responseMessage"></span>
            </fieldset>
        </form>
        <br />
        <div>
            <table border="1">
                <thead>
                    <tr>
                        <th>Student Name</th>
                        <th>Student USN</th>
                        <th>Program Type</th>
                        <th>Awards</th>
                        <th>Branch</th>
                    </tr>
                </thead>
                <tbody>
                    <%
                    //Data base connection part
                    Connection con;
                    Statement st;
                    ResultSet rs = null;
                    try{
                        //Load the type of driver, i.e. Java
application requires jdbc driver
                        Class.forName("com.mysql.jdbc.Driver");

                        //establish connection with MySQL server
                        con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/student
reg_db","root", "");
                        //Create select statement
                        String query = "SELECT * FROM regform";

                        //preparing statement i.e. SQL
                       statements like, insert, update, delete
                        or select
                         st = con.createStatement();
                         rs = st.executeQuery(query);

                    }catch(Exception ex){
                        System.out.println(ex.getMessage());
                    }
                      while(rs.next()){
                    %>
                    <tr>
                        <td><%= rs.getString("Name")%></td>
                        <td><%= rs.getString("Usn")%></td>
                        <td><%= rs.getString("Program")%></td>
                        <td><%= rs.getString("Awards")%></td>
                        <td><%= rs.getString("Branch")%></td>
                    </tr>
```

```
                        <%
                         }
                        %>
                    </tbody>
                </table>
            </div>
             <script type="text/javascript">
                 function navigate(button_Action){
                     var tname = document.RegForm.txtName.value;
                     var tusn = document.RegForm.txtUSN.value;
                     var tprogram = null;
                     //To Get the Selected value of radio button
                     var ele = document.RegForm.programtype;
                     for(i = 0; i < ele.length; i++)
                     {
                         if(ele[i].checked == true)
                             tprogram = ele[i].value;
                     }
                    var opt = document.getElementById("Branch");
                    var optdItems =
                             opt.options[opt.selectedIndex].text;

                     //To Get the Selected value of checkbox
                     var items=document.RegForm.awards;
                     var selectedItems="";
                     for(var i=0; i<items.length; i++){
                         if(items[i].checked==true)
                             selectedItems += items[i].value+",";
                     }
                 //To Check button actions, Which button is clicked
                     var btnInsert =
                        document.getElementById("btnInsert").value;
                     var  btnUpdate =
                        document.getElementById("btnUpdate").value;
                     var btnDelete =
                        document.getElementById("btnDelete").value;
                     var ActionValue = 0;
                     if(button_Action == "btnInsert"){
                         ActionValue = 1;
                     }
                     if(button_Action == "btnUpdate"){
                         ActionValue = 2;
                     }
                    if(button_Action == "btnDelete"){
                        ActionValue = 3;
                    }

                     if(tname ==="" && tusn == "" && (tprogram == ""
|| tprogram == null) && optdItems == "--Select Branch--" &&
selectedItems == ""){
                         alert("Please fill all the details!!!");
                     }
                     else{
```

```
                    //AJAX to send form data to DataPage.jsp
                    var xhr = new XMLHttpRequest();
                    if(window.XMLHttpRequest)
                    {
                        xhr=new XMLHttpRequest();
                    }
                    else
                    {
                        xhr=new ActiveXObject("Microsoft.XMLHTTP");
                    }
                    xhr.onreadystatechange=function ()
                    {
                        if((xhr.readyState==4) &&
(xhr.status==200))
                        {

document.getElementById("responseMessage").innerHTML =
xhr.responseText;
                        }
                    }

xhr.open("GET","DataPage.jsp?name="+tname+"&usn="+tusn+"&program
="+tprogram+"&awards="+selectedItems+"&branch="+optdItems+"&btnI
nsaction="+btnInsert+"&btnUpdaction="+btnUpdate+"&btnDltaction="
+btnDelete+"&Add="+ActionValue+"&Update="+ActionValue+"&Delete="
+ActionValue,"true");
                    xhr.send();
                }
                }
        </script>
    </body>
</html>
```

DataPage.jsp **file:- Here only jsp coding is done, we are not designing user interface, as we are using AJAX in this application. Response message has to be sent back to the HomePage.jsp file. This file is used as a server side file for processing of requests.**

```
<%--
    Document   : DataPage
    Created on : Apr 27, 2020, 11:01:44 PM
    Author     : PrathapKumar MK
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.sql.*" %>
<%
    String name = request.getParameter("name");
    String usn = request.getParameter("usn");
    String programs = request.getParameter("program");
    String pgmtxt = "";
    if(programs.equals("UG"))
        pgmtxt = "Under Graduate";
    else
```

```
            pgmtxt = "Post Graduate";
        String awards = request.getParameter("awards");
        String[] awrdsTxt = awards.split(",");
        String Awd_Value = "";
        for(int j=0;j<awrdsTxt.length;j++){
            if(awrdsTxt[j].equals("NL"))
                Awd_Value = "National Level"+",";
            if(awrdsTxt[j].equals("SL"))
               Awd_Value += "State Level";
            else
             Awd_Value += "University Level";
        }
        String branch = request.getParameter("branch");


          int InsertClicked =
Integer.parseInt(request.getParameter("Add"));
          int UpdateClicked =
Integer.parseInt(request.getParameter("Update"));
          int DeleteClicked =
Integer.parseInt(request.getParameter("Delete"));

        if (InsertClicked == 1) {
                if (name != "" && usn != "" && pgmtxt != "" && Awd_Value !=
"" && branch != "--Select Branch--") {
                    try {
                        //Load type of driver, i.e. our Java application
requires jdbc driver
                        Class.forName("com.mysql.jdbc.Driver");

                        //establish connection with MySQL server
                        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/studentreg_db"
, "root", "");

                        //preparing statement i.e. SQL statements like,
insert, update, delete or select
                        Statement stmt = con.createStatement();

                        //Executing the SQL statements
                        int Create_Records = stmt.executeUpdate("insert
into regform values('" + name + "', '" + usn + "','" + pgmtxt + "','" +
Awd_Value + "','" + branch + "');");

                        if (Create_Records == 1) {
                            out.print("<h3>Record Inserted
sucessfully</h3>");
                        }
        } catch (Exception ex) {
            out.print(ex.getMessage());
        }
    } else {
    %>
```

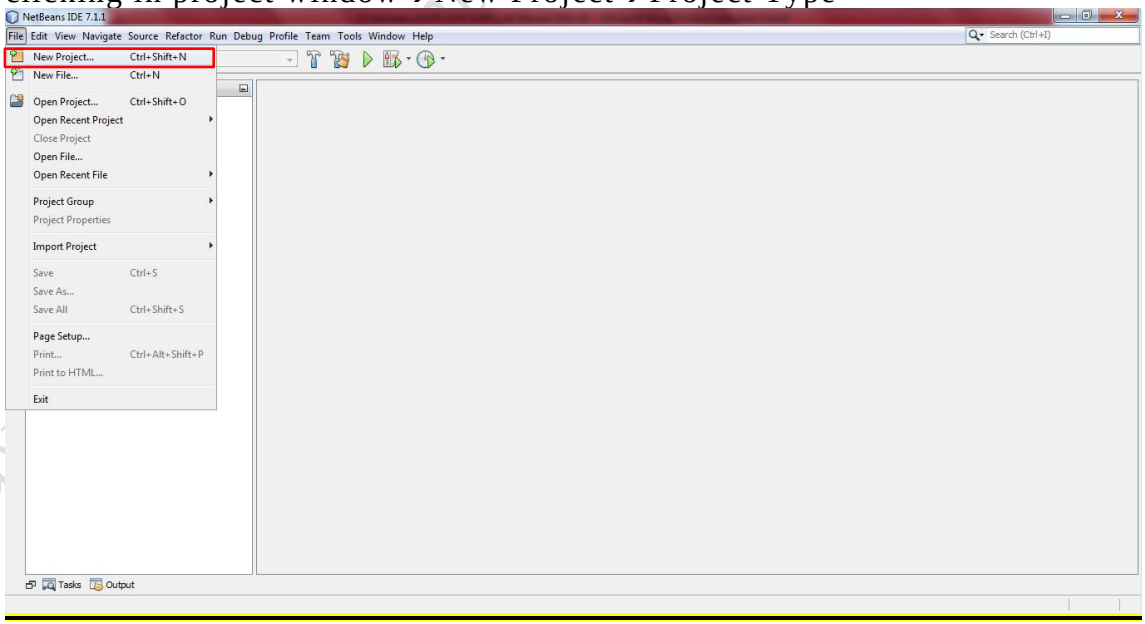
```
<script lang="javascript/text">
    alert("Please fill all the fields!!!");
    window.location =
"http://localhost:8080/EventRegistrationFrom/RegistrationDetails.jsp";
</script>
<%             }
    }
 if (UpdateClicked == 2) {
        //Data base connection part
        Connection con;
        Statement st;
        ResultSet rs = null;
        PreparedStatement pst;

        try {
            //Load the type of Driver
            Class.forName("com.mysql.jdbc.Driver");

            //Establish connection with MySQL server
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/studentreg_db"
, "root", "");

            //Update Query statement
            String query = "UPDATE REGFORM set Name=?, Program=?,
Awards=?, Branch=? where Usn = ?";

            //Prepare sql statement
            pst = con.prepareStatement(query);
            pst.setString(1, name);
            pst.setString(2, pgmtxt);
            pst.setString(3, Awd_Value);
            pst.setString(4, branch);
            pst.setString(5, usn);
            pst.executeUpdate();
            if(pst.executeUpdate() == 1){
                out.print("<h3>Record UPDATED sucessfully</h3>");
            }
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }

    }
    if(DeleteClicked == 3){
          //Data base connection part
        Connection con;
        Statement st;
        ResultSet rs = null;
        PreparedStatement pst;
        try {
            //Load the type of Driver
            Class.forName("com.mysql.jdbc.Driver");
```

```
            //Establish connection with MySQL server
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/studentreg_db"
, "root", "");

            //Prepare sql statement
            String query = "DELETE FROM REGFORM WHERE Usn=?";
            pst = con.prepareStatement(query);
            pst.setString(1, usn);
            pst.executeUpdate();
            if(pst.executeUpdate() == 0){
                  out.print("<h3>Record DELETED sucessfully</h3>");
            }
            con.close();
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    }
%>
```

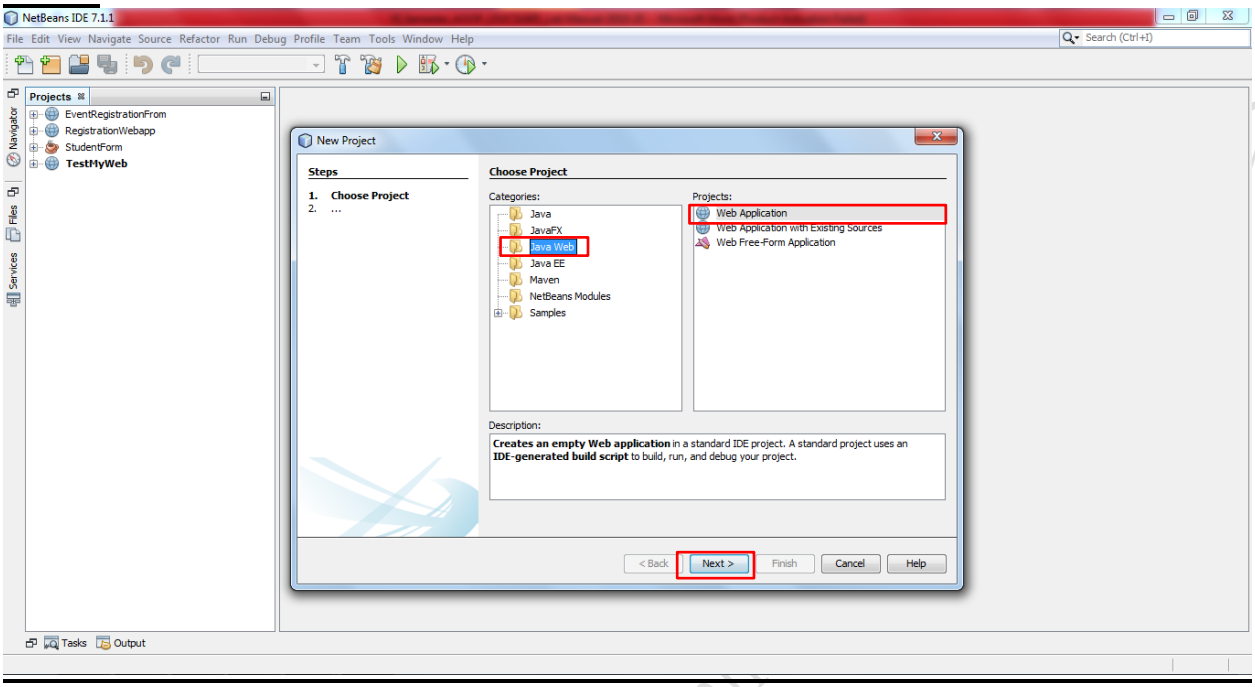
8.3 Design and development of dynamic web application using Servlet in NetBeans IDE:

Here, once again the problem statement defined in the sub section 7.1 of section 7 is implemented. User request is sent via HTML page to the servlet page. Here servlet page is a server side file used to process the request data and generate the response message.

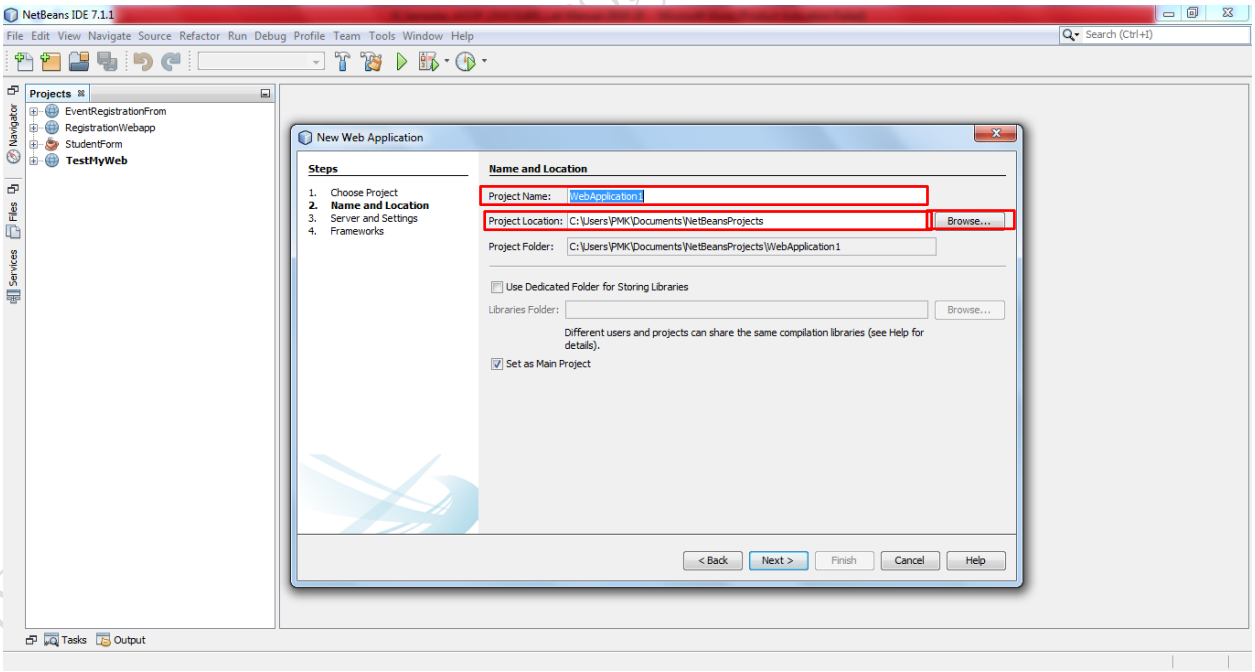
Step 1: Now very beginning, create a web project as shown below, or by right-clicking in project window→New Project→Project Type



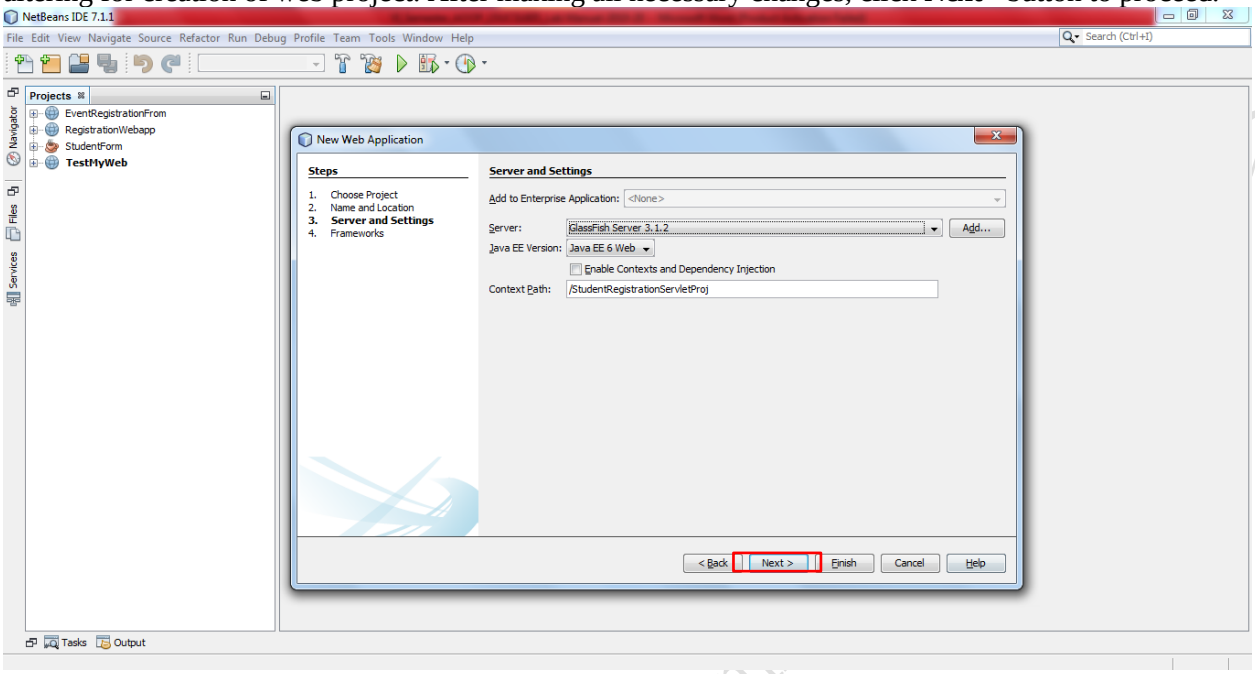
Step 2: Select Project type i.e. select option from Categories and Project panels and click Next> button



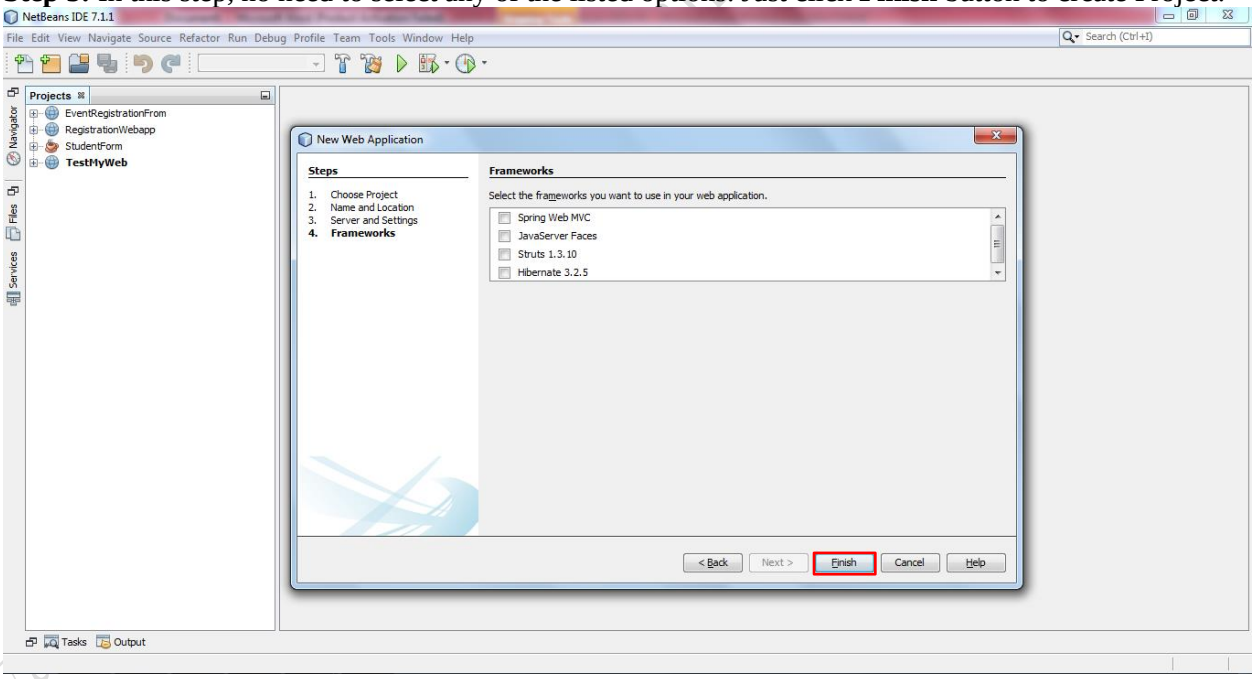
Step 3: Here we are creating new project by entering project name in Project name field as highlighted. Project location can be edited by clicking **Browse** button as show below. Other settings are unaltered.



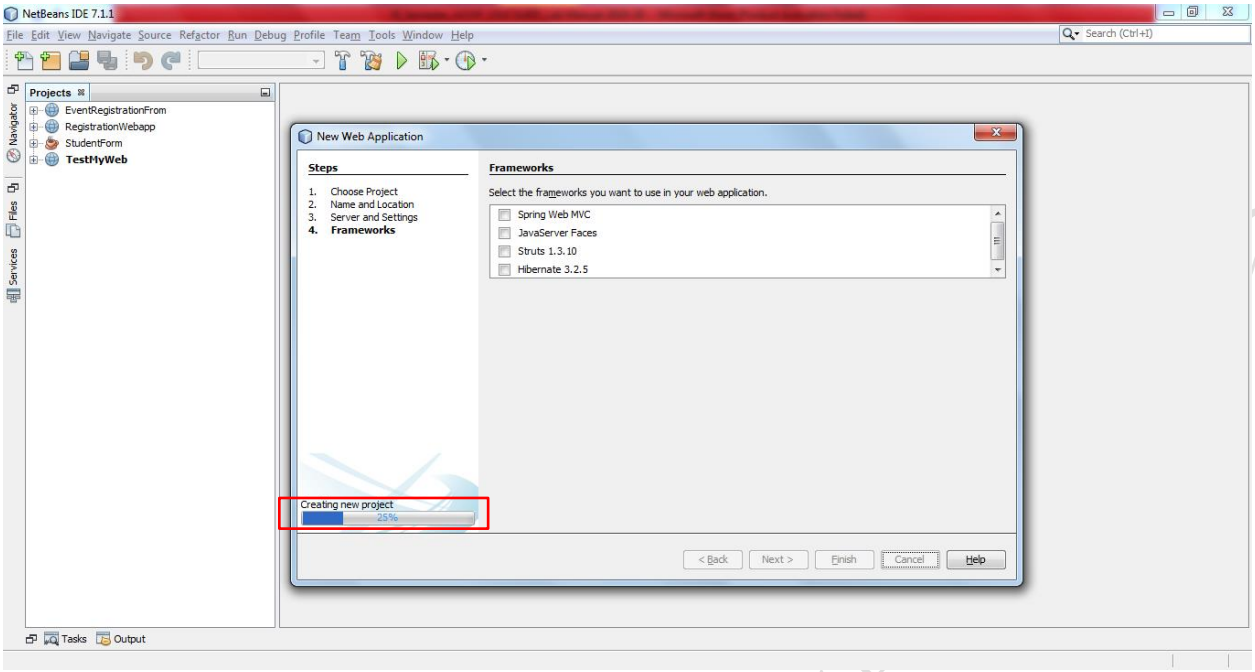
Step 4: Here IDE will display server settings, if any one wants to map to different server instead of GlassFish server, click Add button to choose the different server. Other default settings are used without altering for creation of web project. After making all necessary changes, click **Next>** button to proceed.



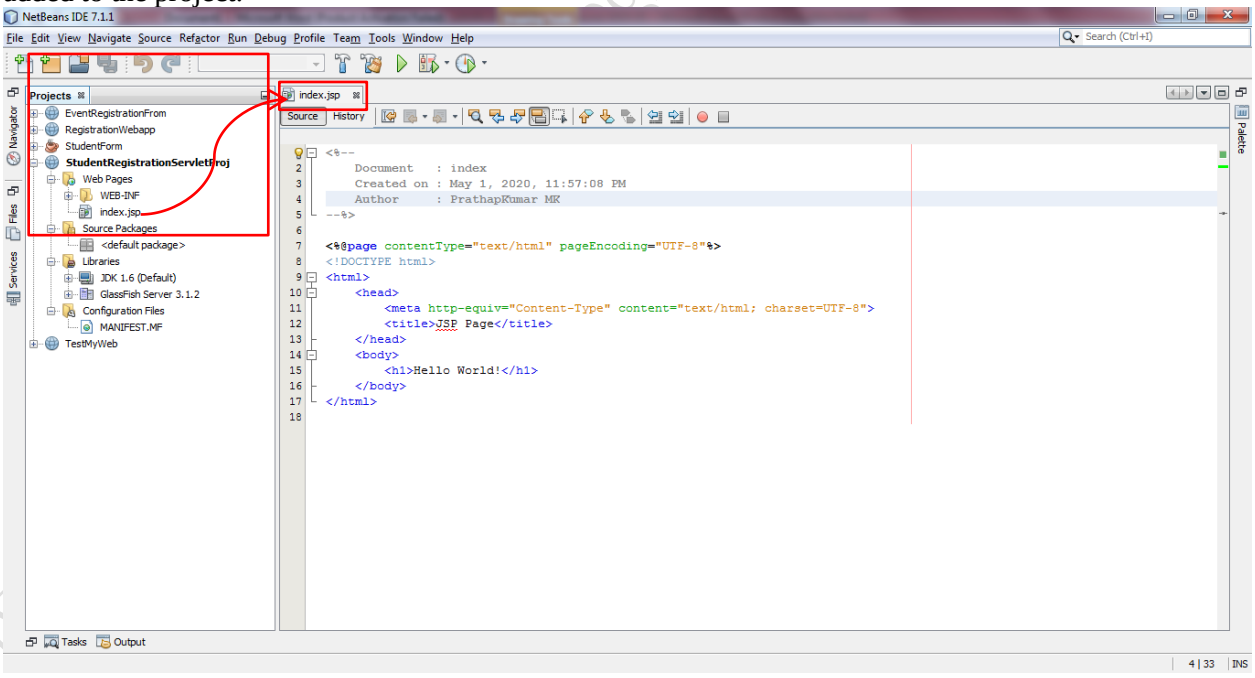
Step 5: In this step, no need to select any of the listed options. Just click **Finish** button to create Project.



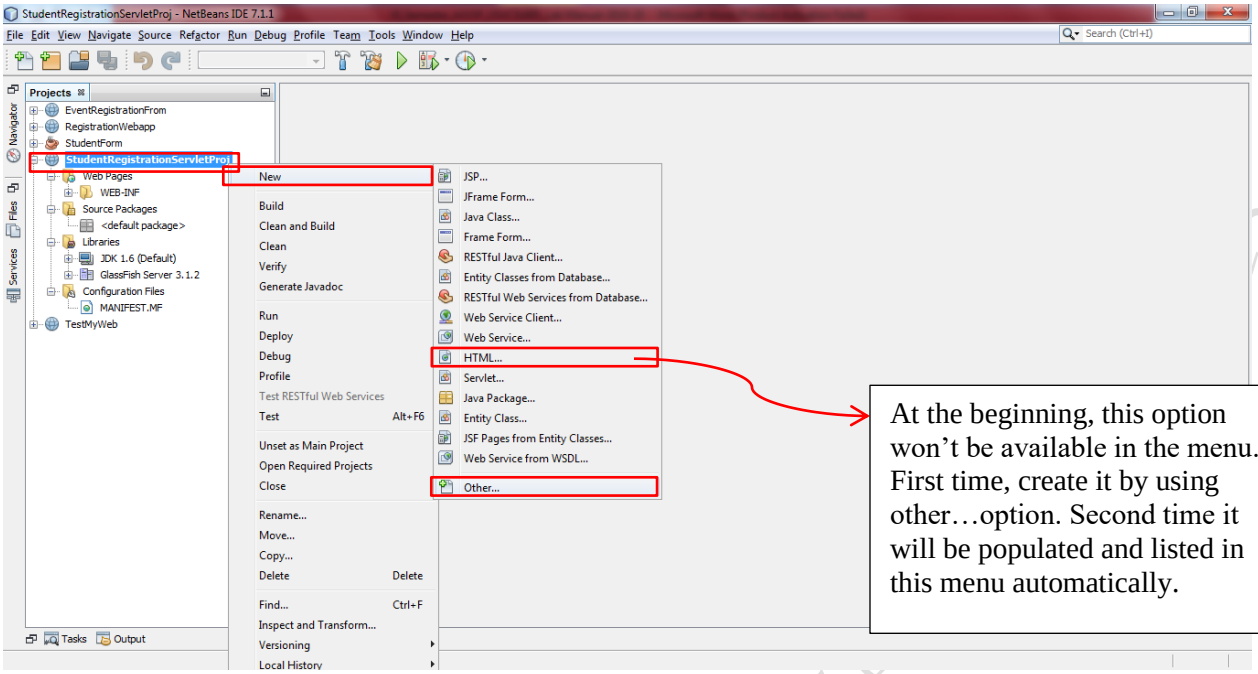
After clicking Finish button, IDE will start creating required project folder structure for the web project as highlighted below:



Step 6: Below screen shot shows generic web application with all required folders, while creating web project, IDE will add index.jsp page as a default page for designing of user interface. As we are using HTML page to collect user information, we are going to **delete** this jsp page and new HTML page will be added to the project.

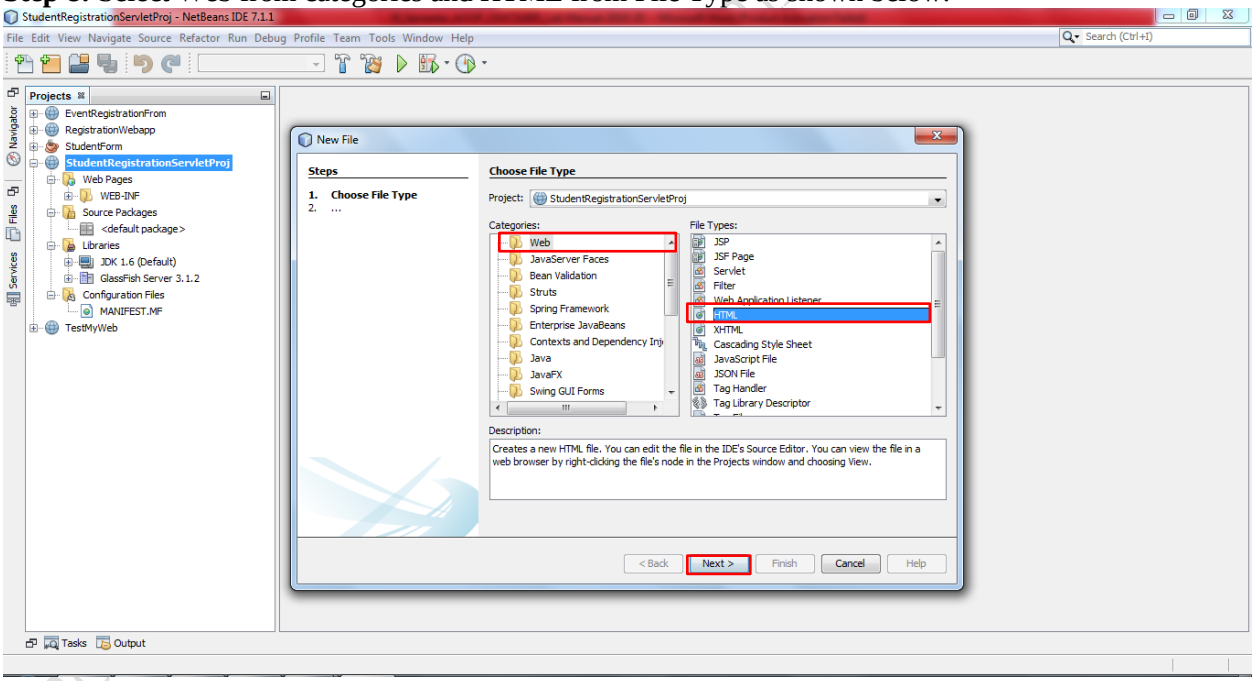


Step 7: Now we will add HTML page to design user interface to collect the information. Select your main **project folder→Right-Click→New→Other…**

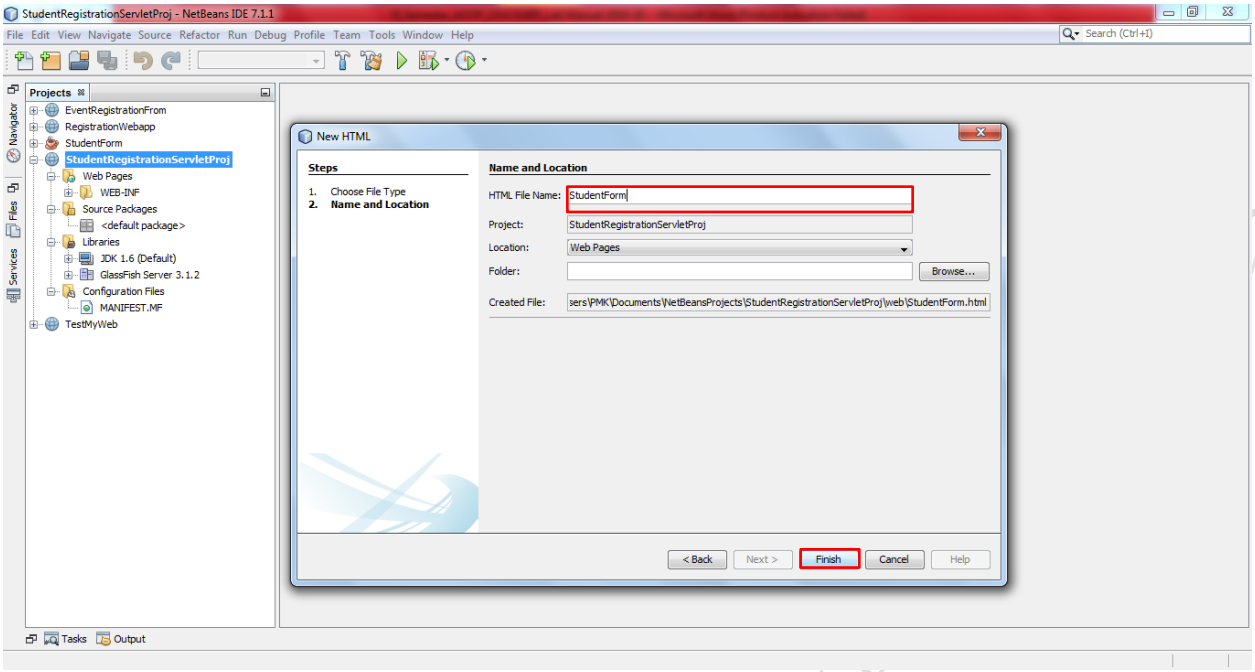


At the beginning, this option won't be available in the menu. First time, create it by using other…option. Second time it will be populated and listed in this menu automatically.

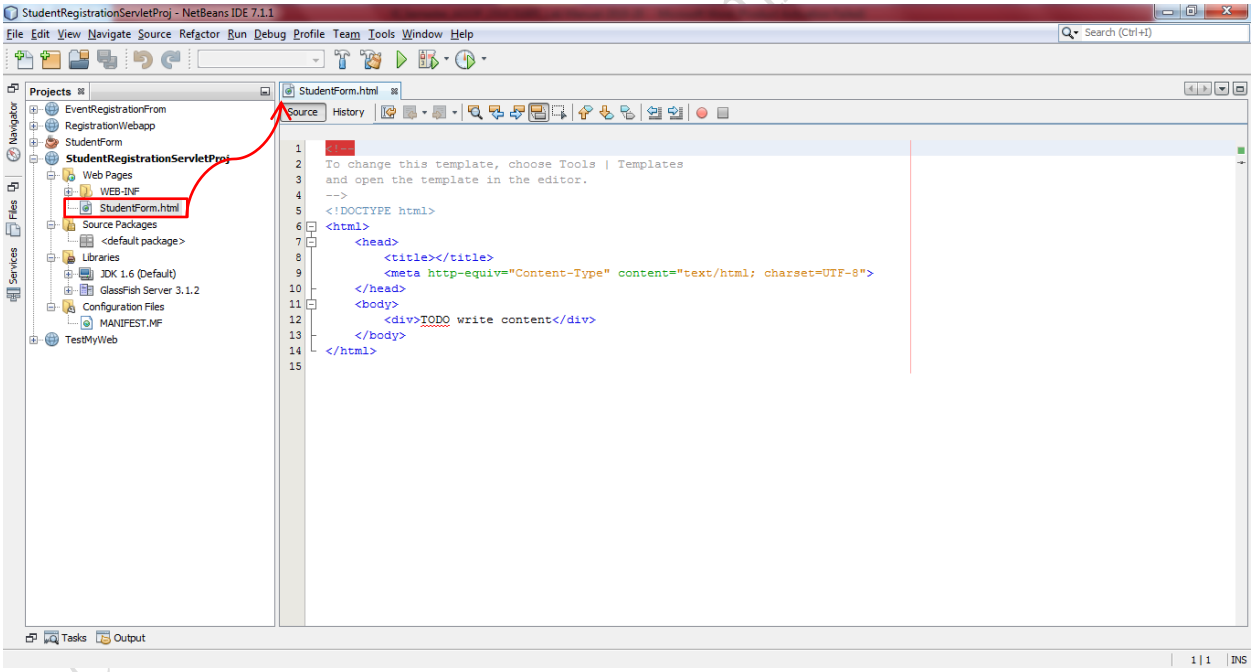
Step 8: Select Web from categories and HTML from File Type as shown below:



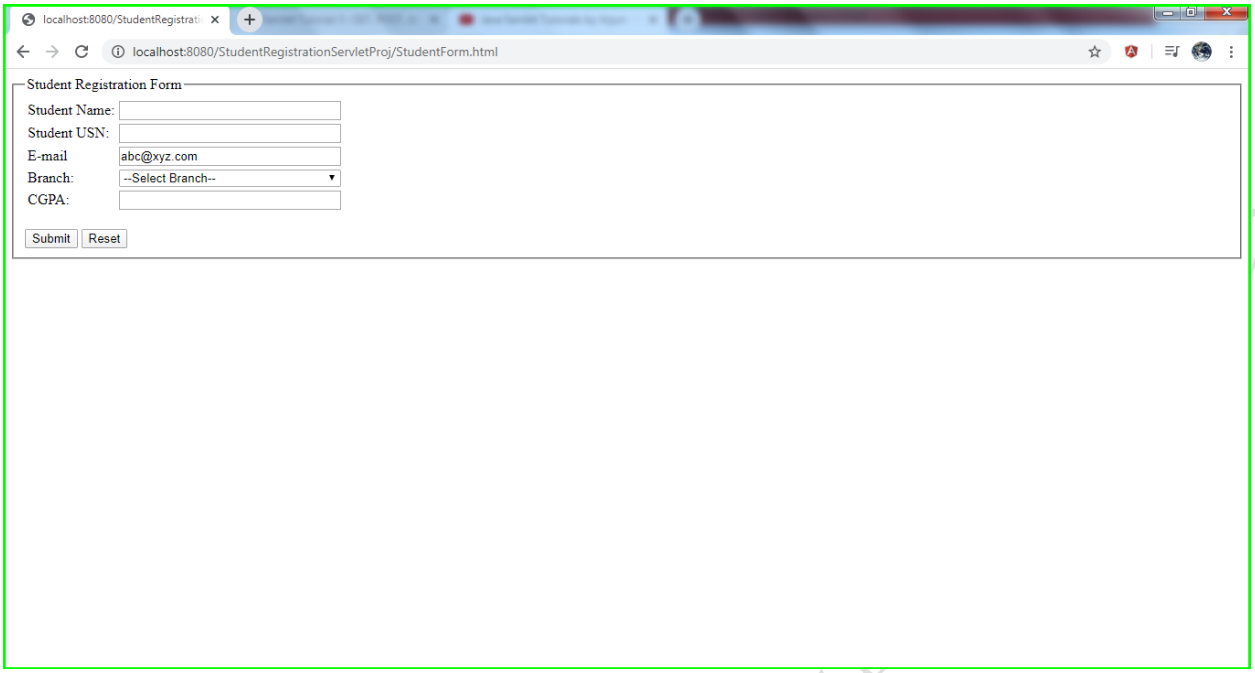
Step 9: Enter the page name without giving file extension, this is because in the previous step, we already selected the file type option as HTML, so no need to give file extension once again here.



Step 10: After adding HTML file to the project folder, by default HTML page will be added in web Pages folder as shown below.



Step 11: Now we will design the web page as per the given problem statement.

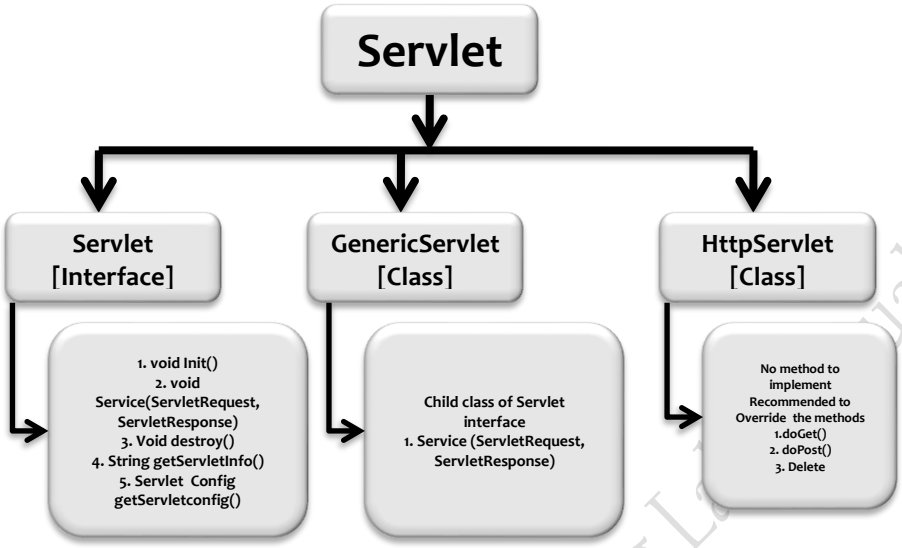


Source Code: **StudentForm.html** file

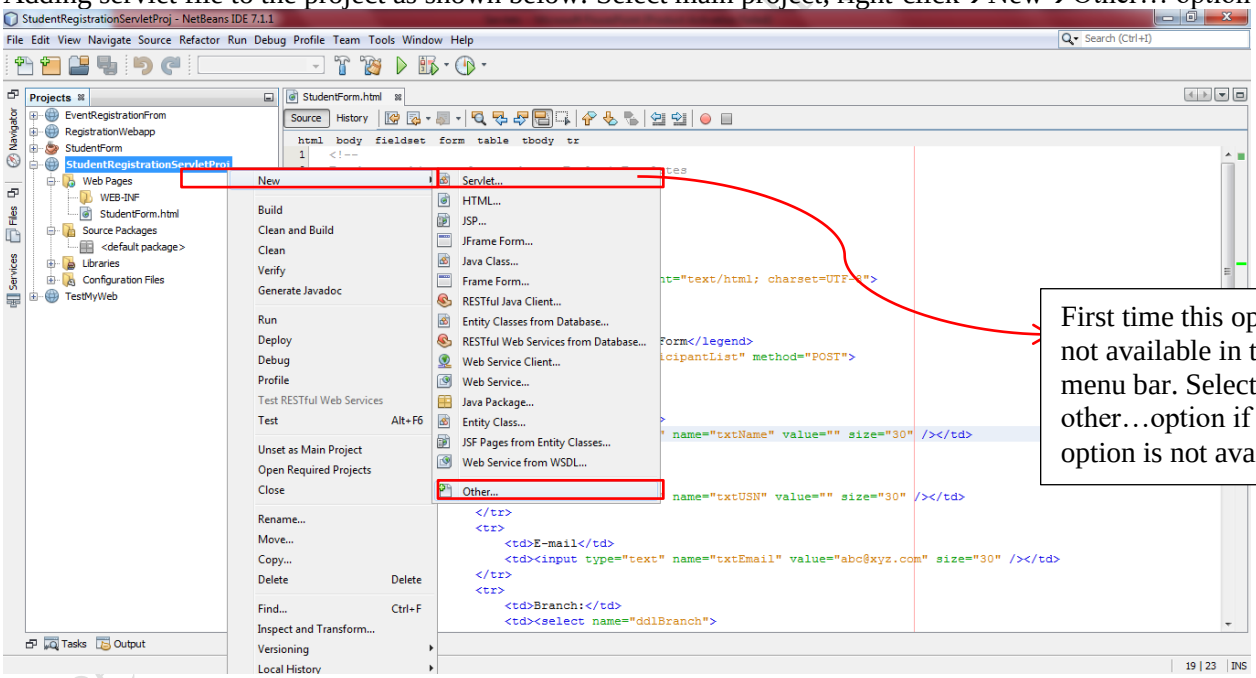
```
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
         <fieldset>
                <legend>Student Registration Form</legend>
        <form name="DetailsForm" action="ParticipantList" method="POST">
            <table border="0">
                <tbody>
                    <tr>
                        <td>Student Name:</td>
                        <td><input type="text" name="txtName" value="" size="30" /></td>
                    </tr>
                    <tr>
                        <td>Student USN:</td>
                        <td><input type="text" name="txtUSN" value="" size="30" /></td>
                    </tr>
                    <tr>
                        <td>E-mail</td>
                        <td><input type="text" name="txtEmail" value="abc@xyz.com" size="30" /></td>
                    </tr>
                    <tr>
                        <td>Branch:</td>
                        <td><select name="ddlBranch">
                                <option>--Select Branch--</option>
                                <option>Computer Science and Engineering</option>
                                <option>Information Science and Engineering</option>
                                <option>Mechanical Engineering</option>
                                <option>Civil Engineering</option>
                                <option>Electronics Engineering</option>
                                <option>Electrical Engineering</option>
                                <option>Chemical Engineering</option>
                            </select></td>
                    </tr>
                    <tr>
                        <td>CGPA:</td>
                        <td><input type="text" name="txtcgpa" value="" size="30" /></td>
                    </tr>
                </tbody>
            </table><br/>
            <input type="submit" value="Submit" name="btnSubmit" />
            <input type="reset" value="Reset" name="btnreset" />
        </form>
    </fieldset>
    </body>
</html>
```

Step 12: Now it's time to add Servlet file. This file is used to process the html file data.

- There are three different ways to implement servlet class; here we are using third approach i.e. extending servlet class with HttpServlet class.

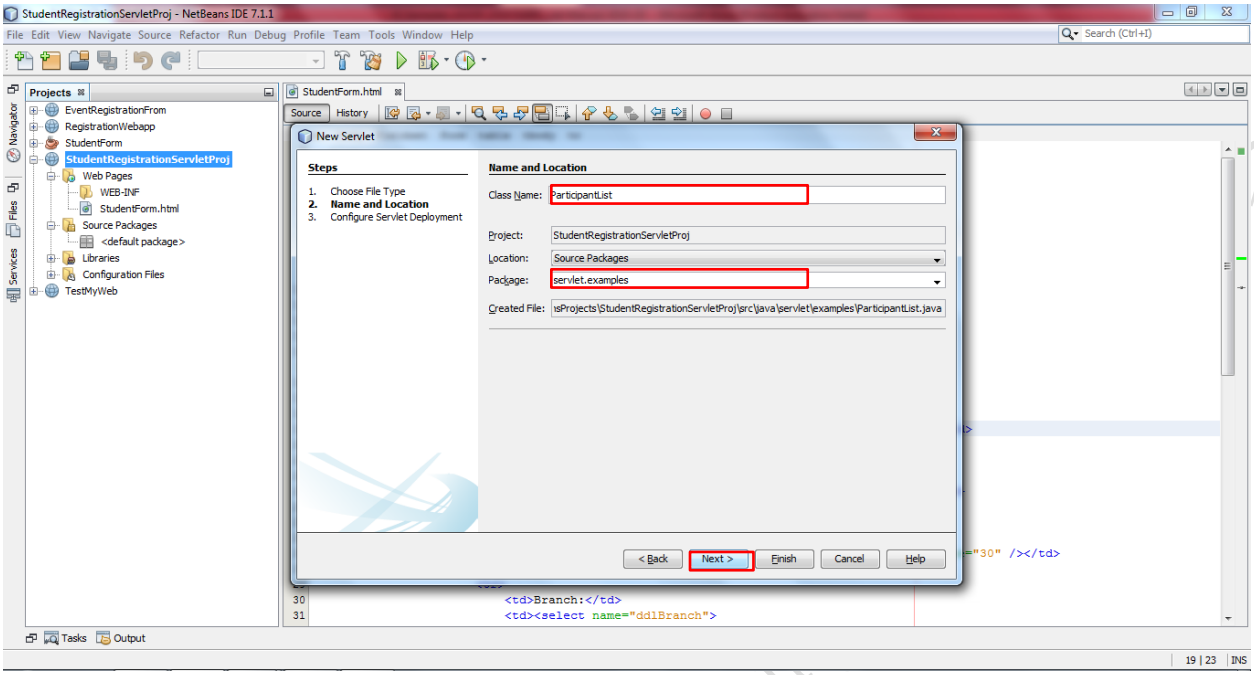


Adding servlet file to the project as shown below: Select main project, right-click→New→Other… option

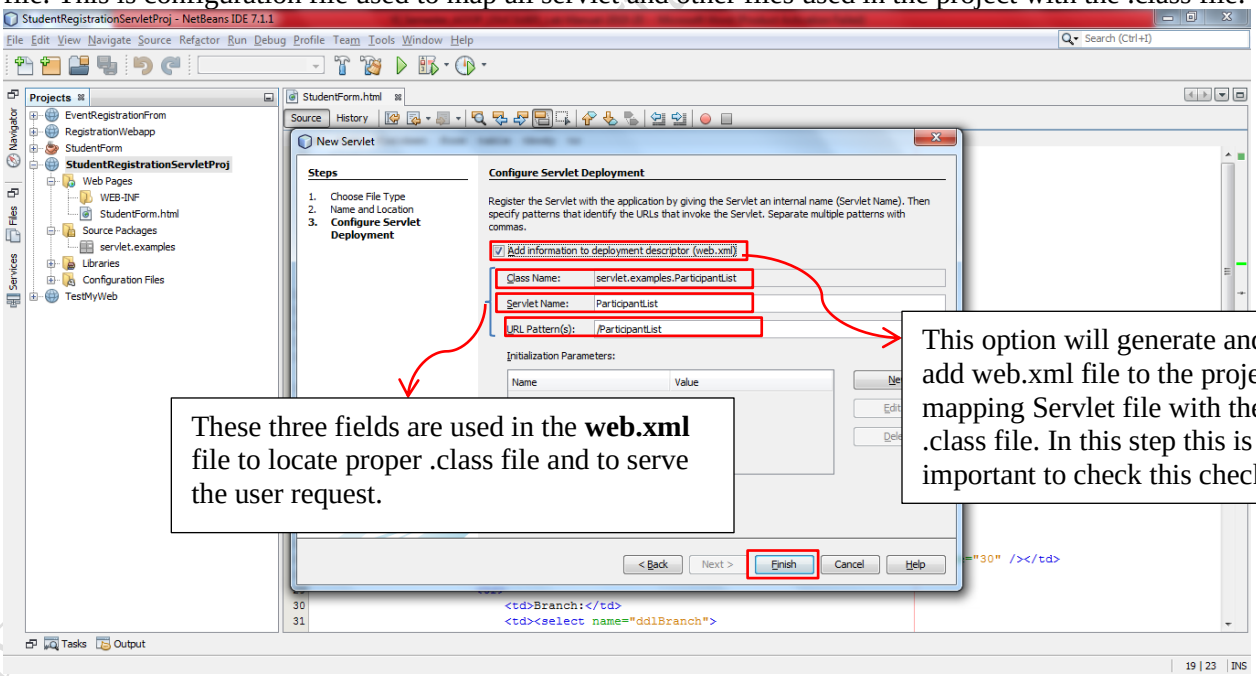


Step 13:

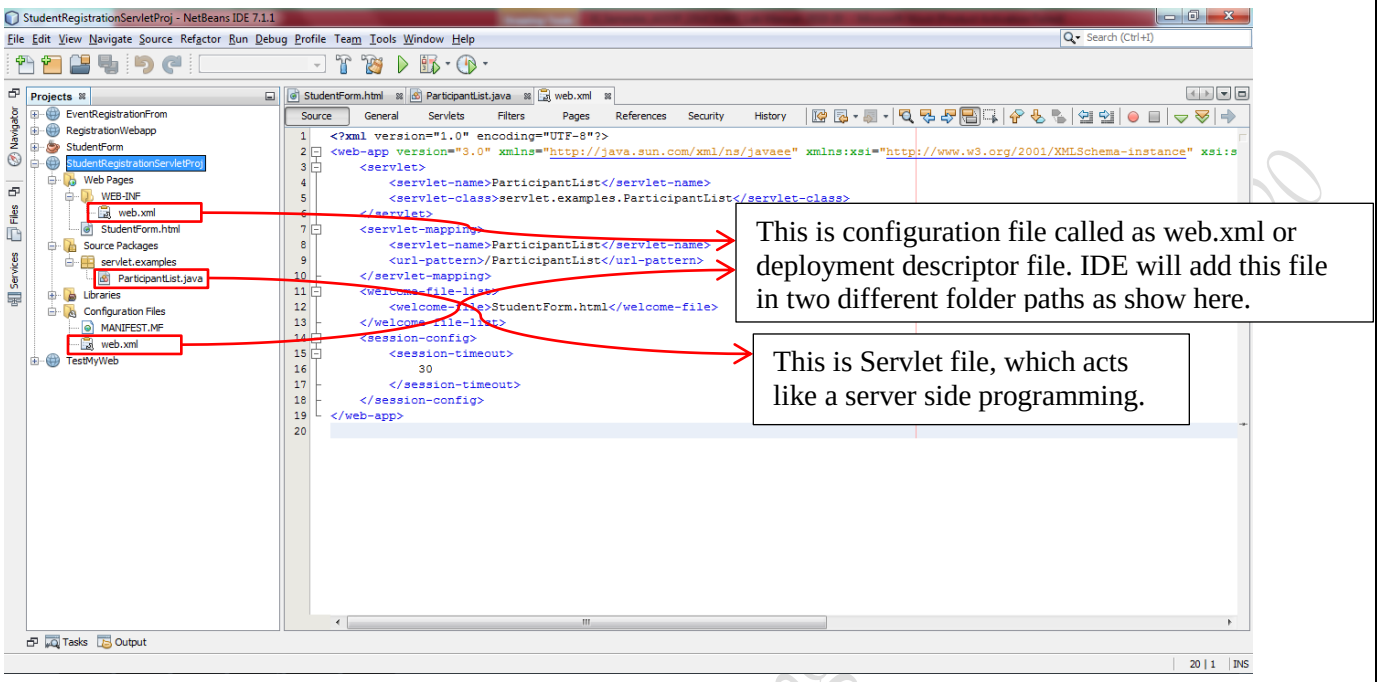
Now Enter the Servlet file name and its package as shown below: do not enter file extension, as we already selected file type as Servlet.



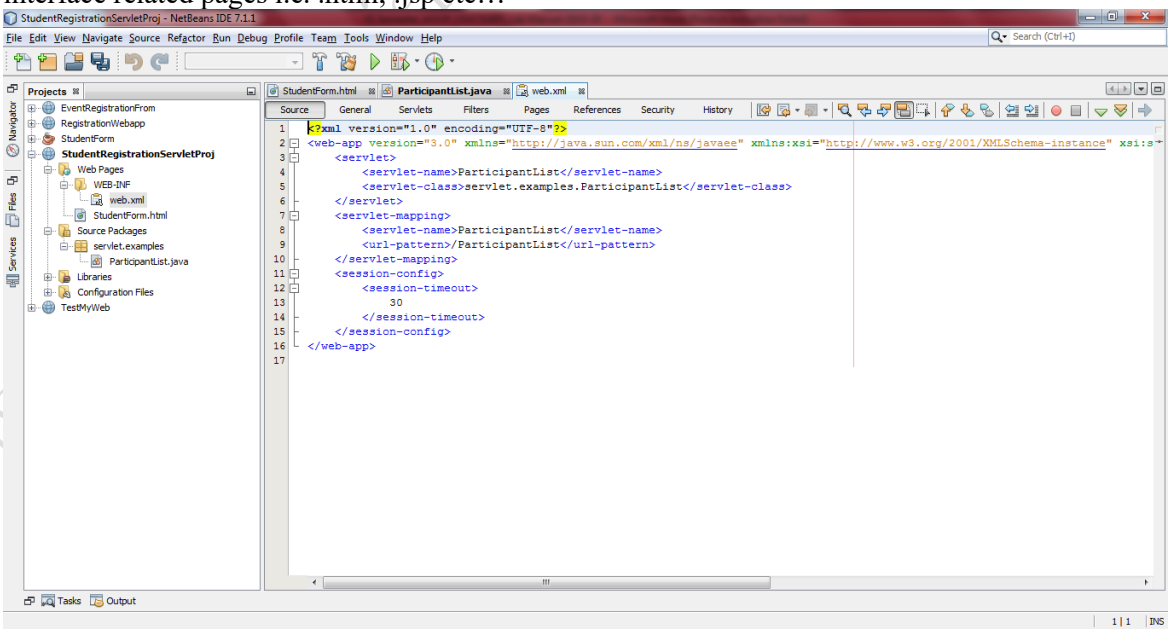
Step 14: This is very important step, here we need to check checkbox to add deployment descriptor (web.xml) file to the project. This can be added manually, but better to use this option to generate web.xml file. This is configuration file used to map all servlet and other files used in the project with the .class file.



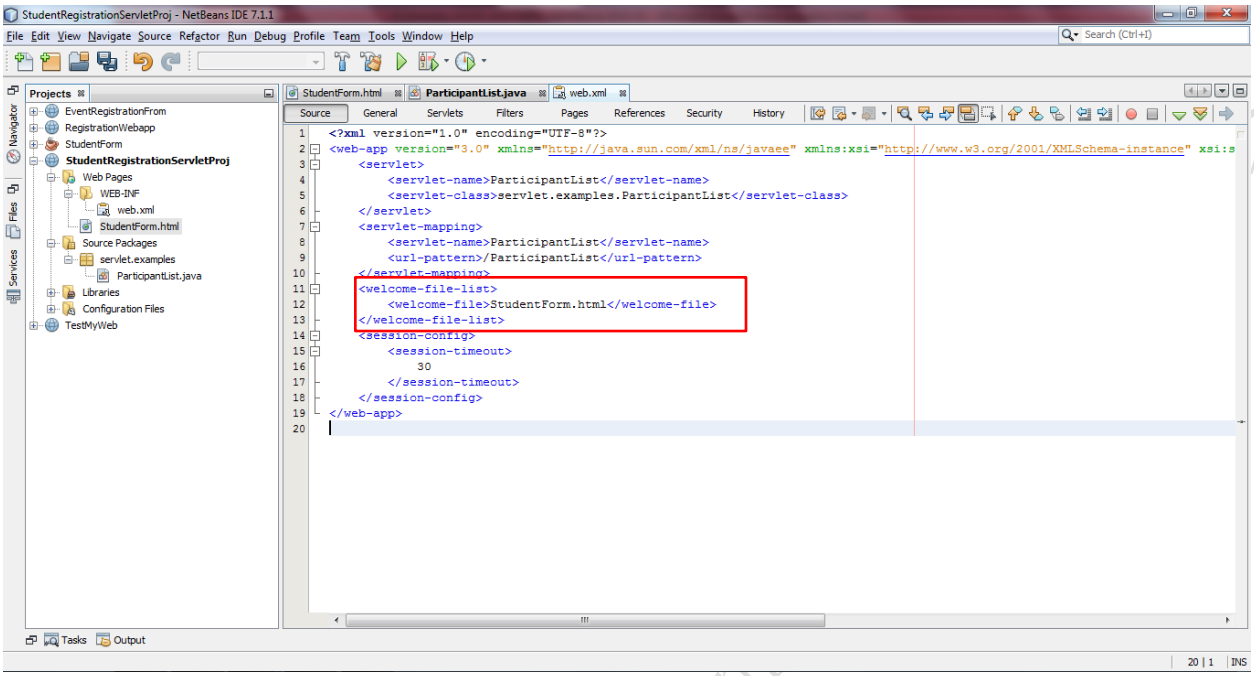
Now project folder structure is updated with the two new files i.e. Servlet file and web.xml file as shown below. Here file path of these two files are to be remembered.



Now we will go through the configuration (**web.xml/deployment descriptor**) file. Here xml nodes are auto generated; do not modify the file structure. Basically this file is divided into four parts: **First part** i.e. <servlet> element contains all servlet files and its corresponding .class files. It provides information about servlet file and its .class file path. Mapping can be done using element <**Servlet-name>.** This element holds the Servlet file name, which is Java file (here its name is ParticipantList.java) and another element <servlet-class>, this element holds path of the .class file i.e. Servlet file class path. **Second part** is basically used for mapping purpose. Here root element is <servlet-mapping> under this two child elements are used to map files. <servlet-name> holds the Servlet file and it is mapped with <url-pattern> of the project. **Third part** contains about project session time out. Finally **Fourth part** is listing out all user interface related pages i.e. .html, .jsp etc…



Step 15: Now we will add **StudentForm.html** file to the web.xml file, which was added to the project at the beginning for designing of user interface.



Step 16: Now we will add code to process the httprequest message. In Servlet file (i.e. .java file only but file type is servlet file) under **doPost ()** method of servlet class, we will add code. Calling of servlet class method depends on method attribute of <form> element in the HTML file. If attribute value is "GET", then **doGet ()** method is used to code in the servlet file otherwise **doPost ()** method.
Source code of doPost () method:

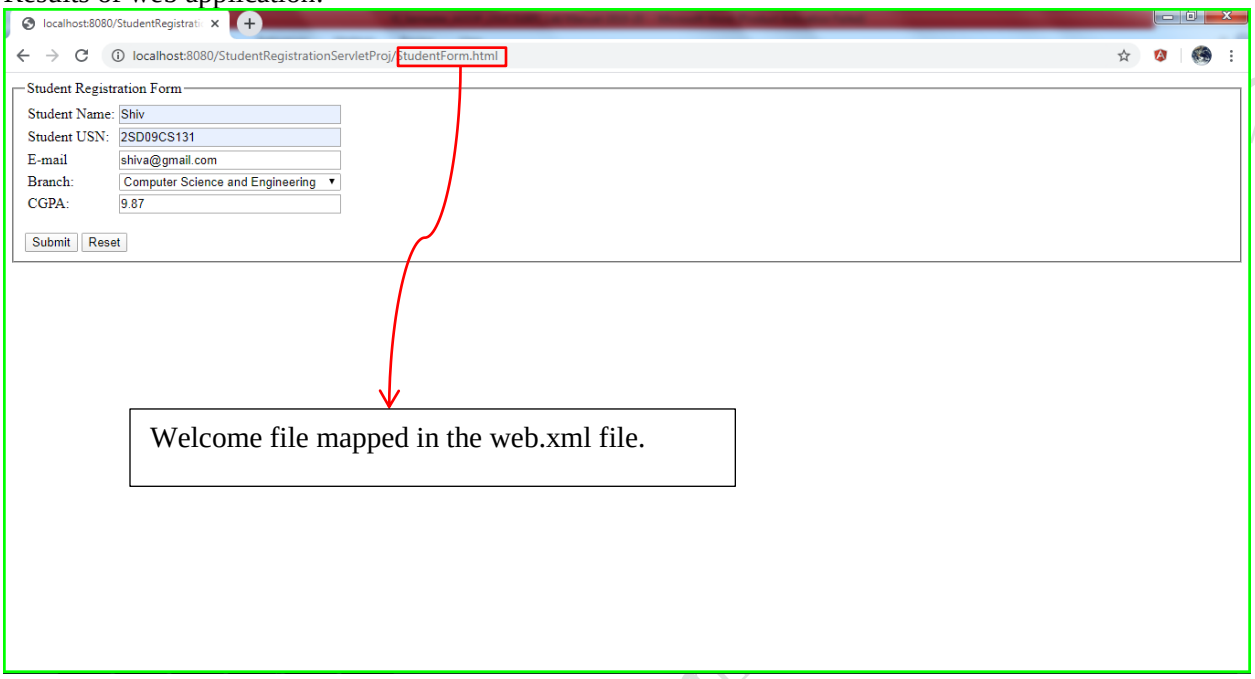
```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
         // collecting the HTML page data, i.e. user input data
            String name = request.getParameter("txtName");
            String usn = request.getParameter("txtUSN");
            String email = request.getParameter("txtEmail");
            String branch = request.getParameter("ddlBranch");
            String cgpa = request.getParameter("txtcgpa");

      response.setContentType("text/html;charset=UTF-8");
      PrintWriter pout = response.getWriter();
        pout.println("<html><head><title>Participants
List</title></head><body><table border=\"1\"><tbody><tr><td><b>Student
Name</b></td><td><b>USN</b></td><td><b>Email</b></td><td><b>Branch</b></td>"
                + "<td><b>CGPA</b></td></tr>
<tr><td>"+name+"</td><td>"+usn+"</td><td>"+email+"</td><td>"+branch+"</td><td>"+cg
pa+"</td></tr></tbody>"
        + "</table><br/><a href=\"StudentForm.html\">Home
Page</a></body></html>");
    }
```

Step 17:

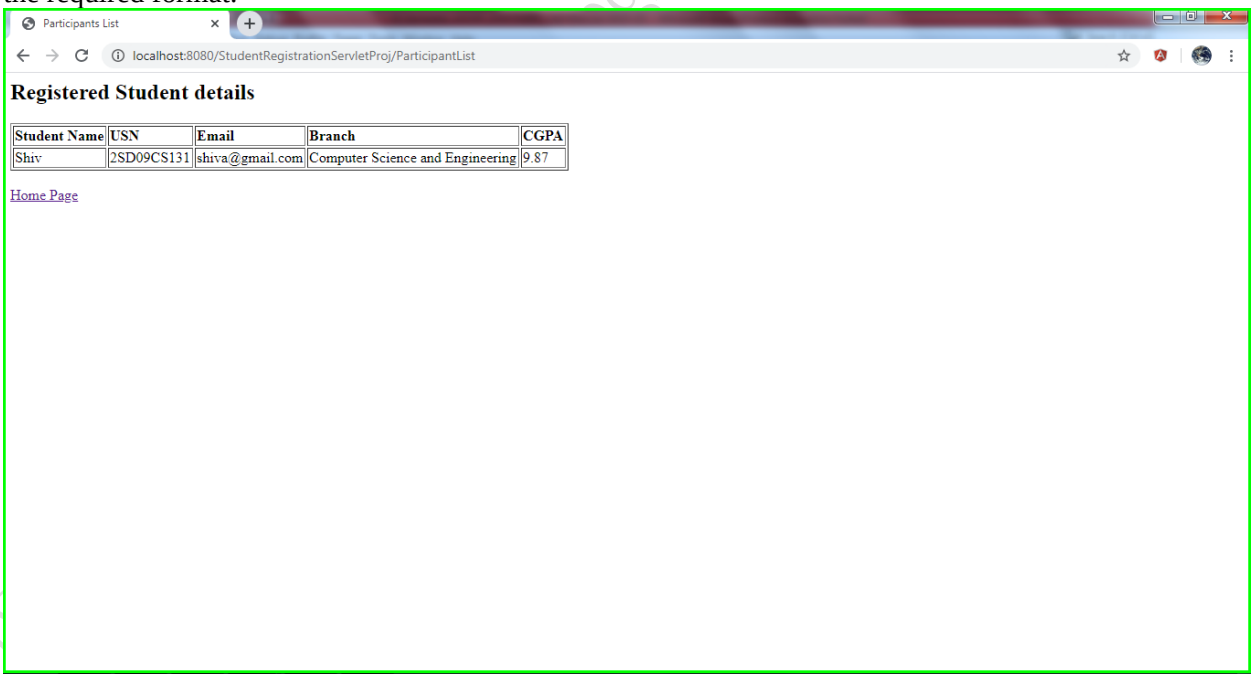
Now we will execute the project. First we need to compile servlet file and then select main project folder right-click on it→Run option. This option will execute the .class file of servlet file and deploy the

project on the Glassfish server by starting the server if it is not started and then it will load the welcome file onto the web browser which is mapped in the **web.xml** file.

Results of web application:



After submitting above screen, it will read user request and process it using Servlet file and displayed in the required format.

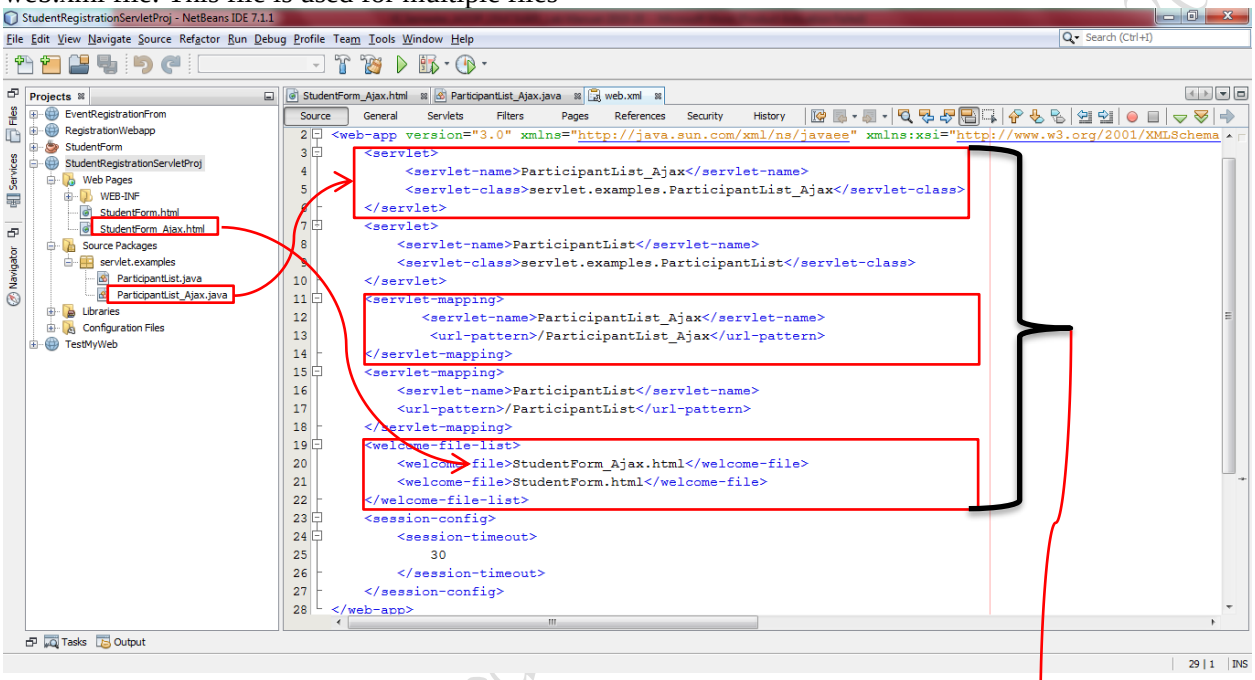


8.4 Design and development of Dynamic web application using HTML, Servlet, JavaScript, AJAX & MySQL database.

Here, once again we will consider the problem statement defined in the sub section 8.1 of section 8 to develop web application.

To add the required files like HTML, Servlet refer the above given steps. To add SQL connector API jar file to the project, follow the steps (database connection steps) given in the 8.1 section.

web.xml file: This file is used for multiple files



The highlighted rectangular boxes contain elements related to project/file which uses AJAX. All startup pages like HTML, JSP files are listed using single welcome-file-list element. Other elements for non-AJAX based project. Here you're learning how to use single web.xml file to load multiple files.