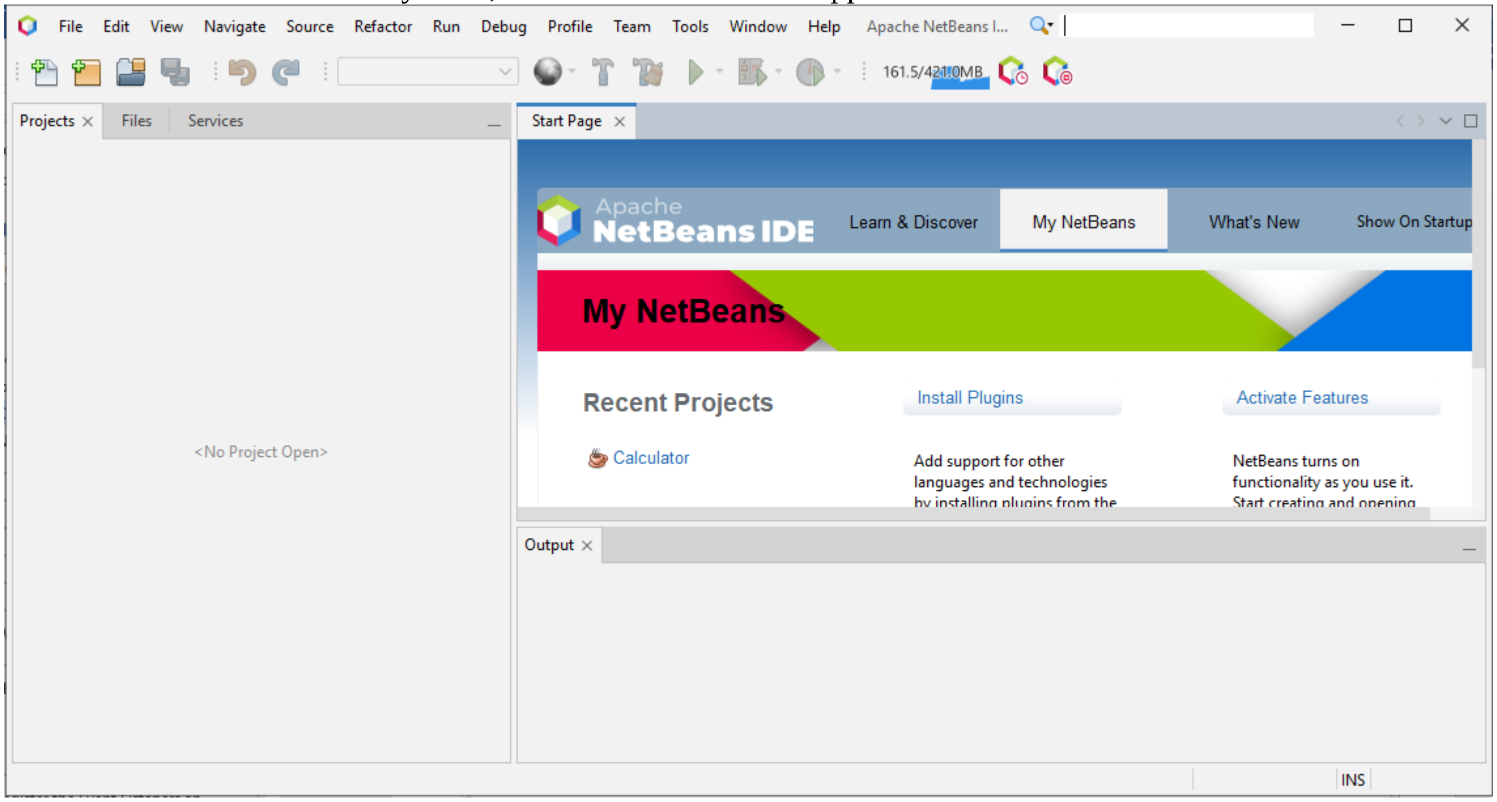
Laboratory Manual 2
Developing Java Applications using NetBeans IDE

1.1 Create an IDE Project

To create an IDE project:

1. Launch the NetBeans IDE.
   - On Microsoft Windows systems, you can use the NetBeans IDE item in the Start menu.
   - On Solaris OS and Linux systems, you execute the IDE launcher script by navigating to the IDE's `bin` directory and typing `./netbeans`.
   - On Mac OS X systems, click the NetBeans IDE application icon.



Add support for other languages and technologies by installing plugins from the

NetBeans turns on functionality as you use it. Start creating and opening

Figure 1: NetBeans IDE welcome start up screen

2. In the NetBeans IDE, choose **File** =>**New Project... or Click New Project Icon** from the tool bar menu

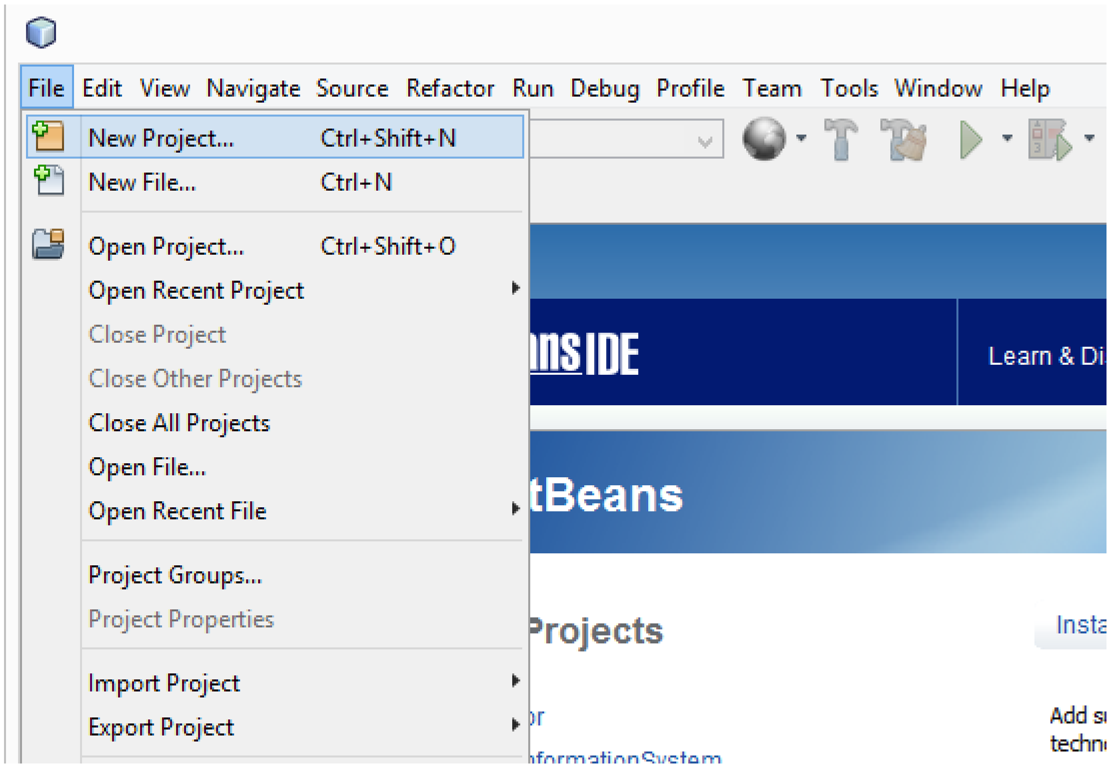
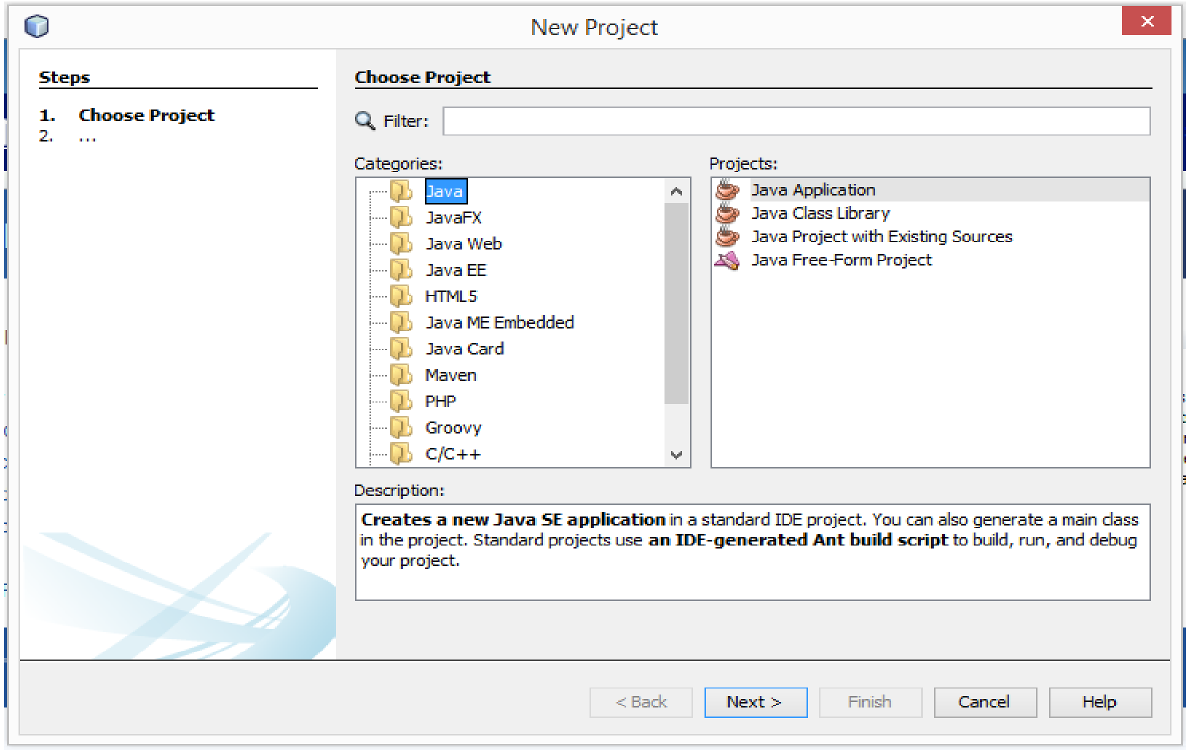


Figure 2: Create a new project

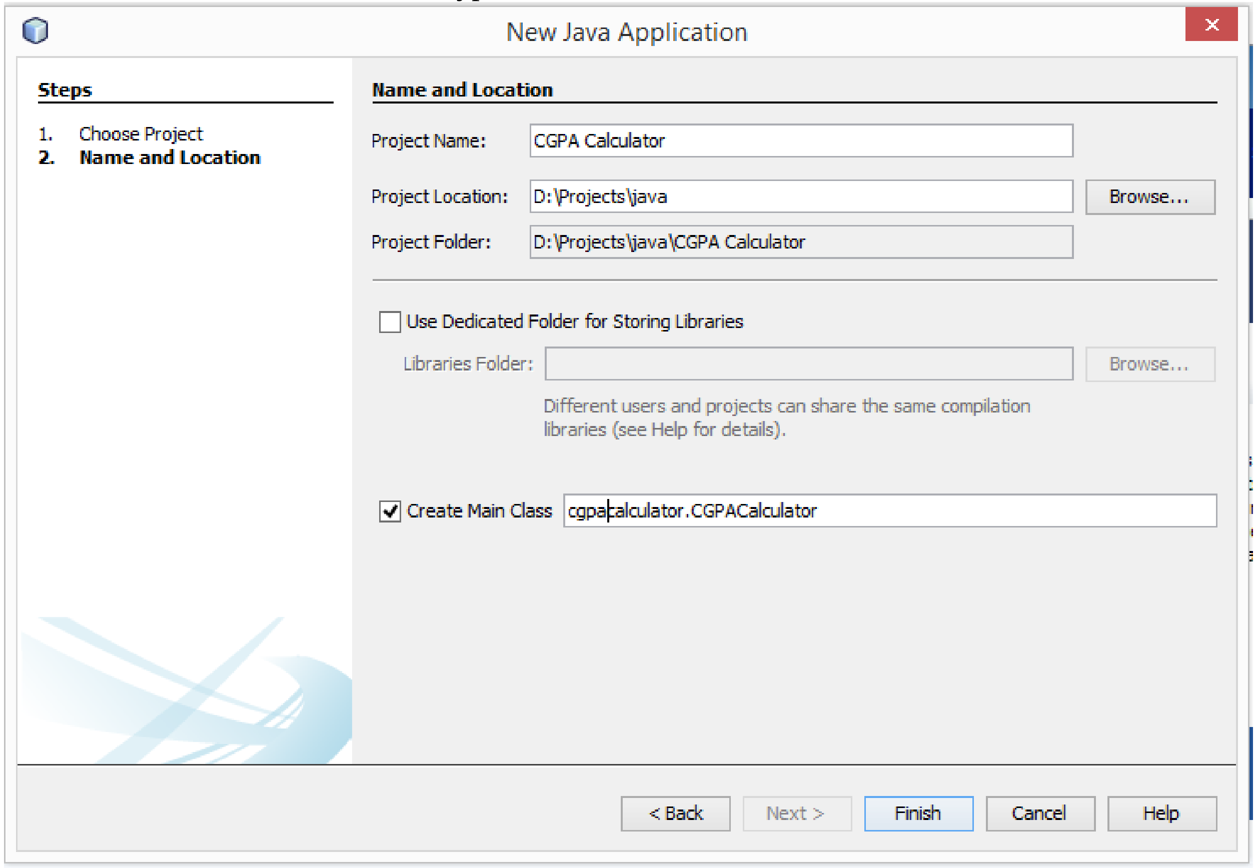
3. In the New Project wizard, expand the Java category and select Java Application as shown in the following figure:



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Figure 3: NetBeans IDE, New Project wizard, Choose Project page

4. In the **Name and Location** page of the wizard, do the following (as shown in the figure below):
   - In the **Project Name** field, type `CGPA Calculator`.
   - In the **Create Main Class** field, type `CGPA Calculator. CGPA Calculator`



Name and Location

Project Name: CGPA Calculator

Project Location: D:\Projects\java

Project Folder: D:\Projects\java\CGPA Calculator

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

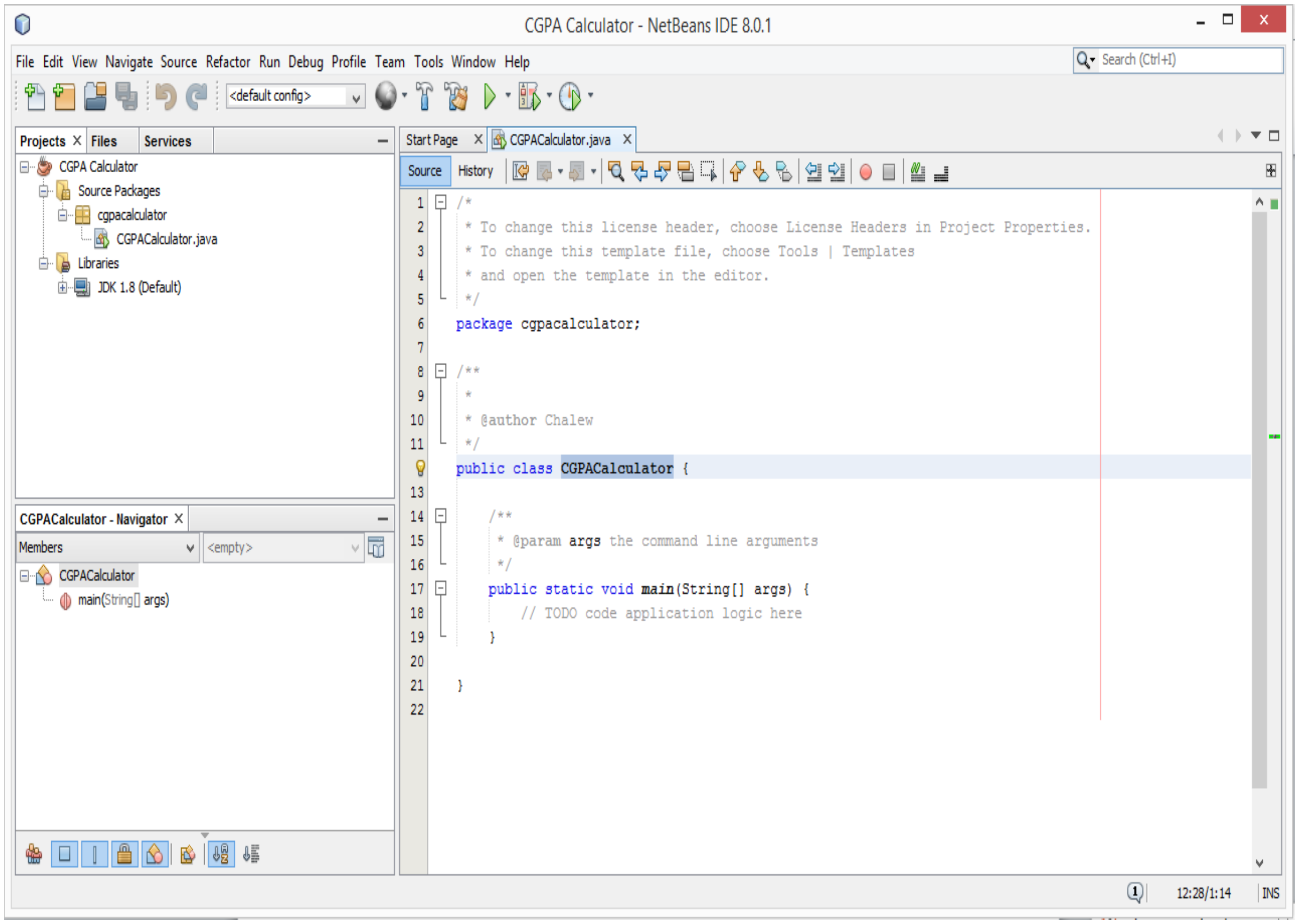
☑ Create Main Class cgpacalculator.CGPACalculator

Fig : NetBeans IDE, New Project wizard, Name and Location page.

5. Click Finish.

The project is created and opened in the IDE. You should see the following components:

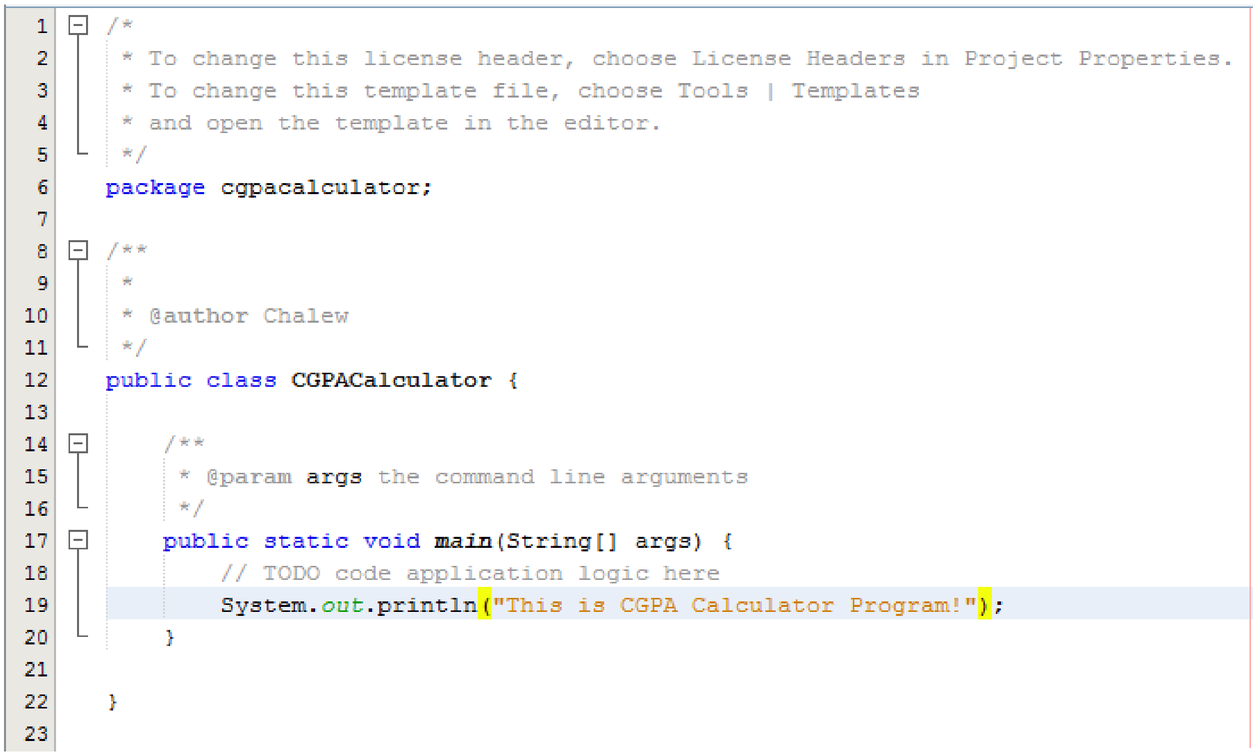
- The **Projects** window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The **Source Editor** window with a file called `CGPACalculator.java` open.
- The **Navigator** window, which you can use to quickly navigate between elements within the selected class.



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cgpacalculator;

/**
 *
 * @author Chalew
 */
public class CGPACalculator {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

Fig : NetBeans IDE with the CGPA Calculator project open.

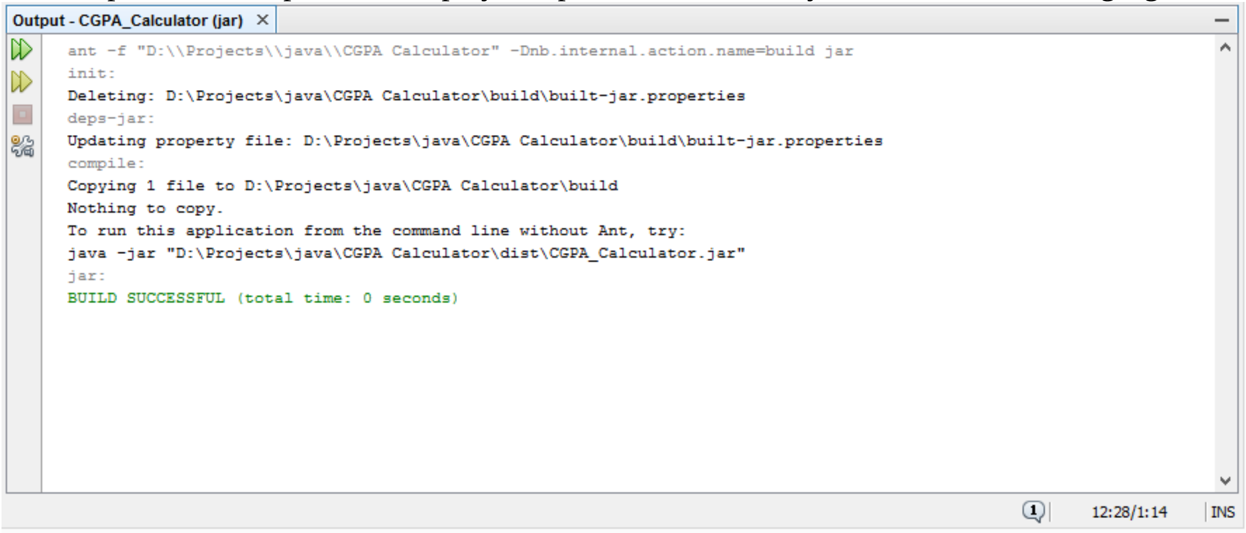
6. Add Code to the Generated Source File
   When you created this project, you left the **Create Main Class** checkbox selected in the **New Project** wizard. The IDE has therefore created a skeleton class for you. You can add the "This is CGPA Calculator Program!" message to the skeleton code by replacing the line:
   ```
   // TODO code application logic here
   ```
   with the line:
   ```
   System.out.println("This is CGPA Calculator Program!"); // Display the string.
   ```
   Save your changes by choosing **File** => **Save**.
   The file should look something like the following:



```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cgpacalculator;

/**
 *
 * @author Chalew
 */
public class CGPACalculator {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("This is CGPA Calculator Program!");
    }

}
```

Fig : First Code in NetBeans IDE

7. Compile the Source File into a .class File
   To compile your source file, choose **Run** | **Build Project (CGPA Calculator)** from the IDE's main menu. The Output window opens and displays output similar to what you see in the following figure:



```
ant -f "D:\\Projects\\java\\CGPA Calculator" -Dnb.internal.action.name=build jar
init:
Deleting: D:\Projects\java\CGPA Calculator\build\built-jar.properties
deps-jar:
Updating property file: D:\Projects\java\CGPA Calculator\build\built-jar.properties
compile:
Copying 1 file to D:\Projects\java\CGPA Calculator\build
Nothing to copy.
To run this application from the command line without Ant, try:
java -jar "D:\Projects\java\CGPA Calculator\dist\CGPA_Calculator.jar"
jar:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig : Output window showing results of building the CGPA Calculator project

If the build output concludes with the statement, `BUILD SUCCESSFUL`, congratulations! You have successfully compiled your program!

If the build output concludes with the statement `BUILD FAILED`, you probably have a syntax error in your code. Errors are reported in the Output window as hyperlinked text. You double-click such a hyperlink to navigate to the source of an error. You can then fix the error and once again choose **Run** | **Build Project**.

When you build the project, the bytecode file the CGPACalculator.class is generated. You can see where the new file is generated by opening the **Files** window and expanding the **CGPA Calculator /build/classes/ cgpacalculator** node as shown in the following figure.

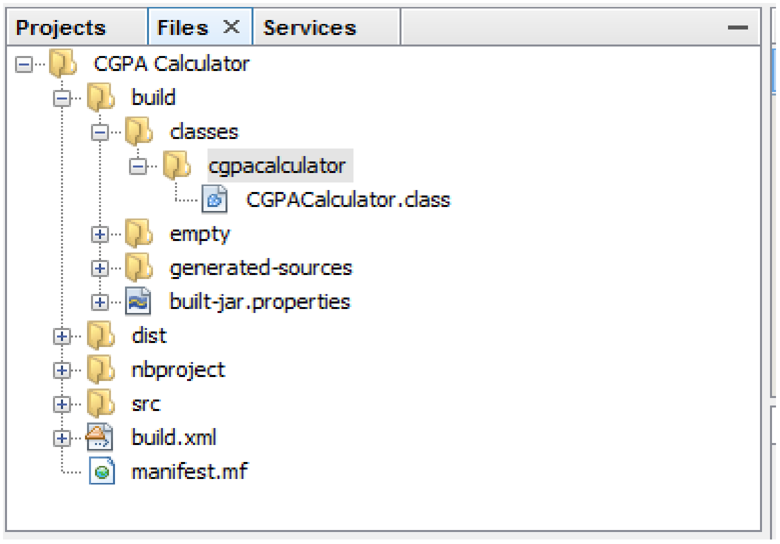
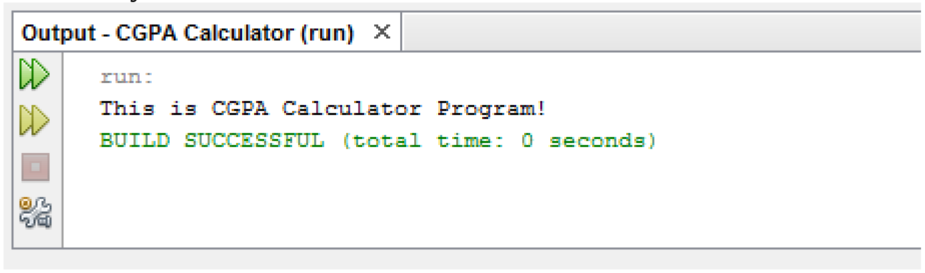


Fig : Files window, showing the generated .class file.

8. Run the Program
   From the IDE's menu bar, choose **Run** | **Run Main Project**.
   The next figure shows what you should now see.



```
run:
This is CGPA Calculator Program!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig : The program prints " This is CGPA Calculator Program!" to the Output window (along with other output from the build script).

The following are some tips on using the IDE and explanations of some IDE behavior that you are likely to see:

- Once you have created a project in the IDE, you can add files to the project using the **New File** wizard. Choose **File** | **New File**, and then select a template in the wizard, such as the Empty Java File template.
- You can compile and run an individual file (as opposed to a whole project) using the IDE's **Compile File** (F9) and **Run File** (Shift-F6) commands.
- You might want to create separate IDE projects for sample applications that include more than one source file.
- As you are typing in the IDE, a code completion box might periodically appear. You can either ignore the code completion box or keep typing, or you can select one of the suggested expressions.
- If you would prefer not to have the code completion box automatically appear, you can turn off the feature. Choose **Tools** | **Options** | **Editor**, click the **Code Completion** tab and clear the **Auto Popup Completion**

Window checkbox.

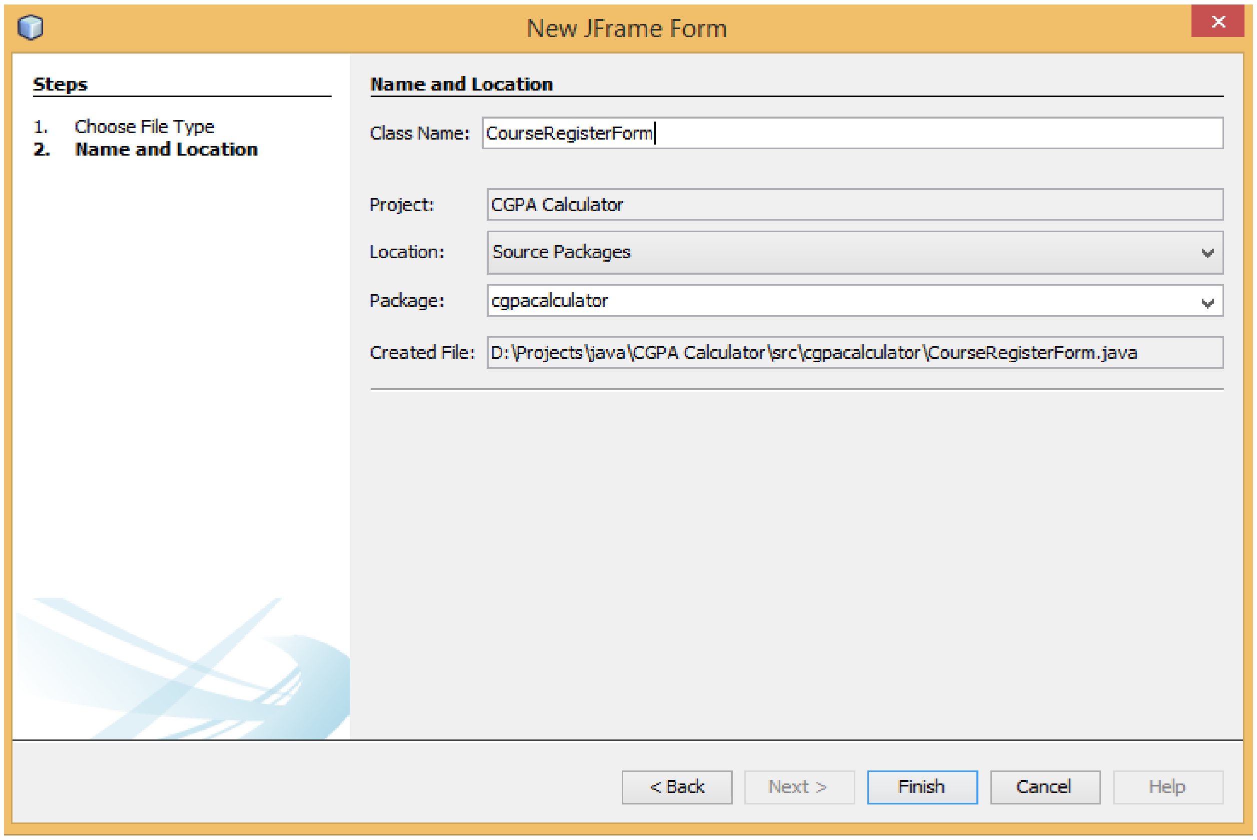
- If you want to rename the node for a source file in the **Projects** window, choose **Refactor** from IDE's main menu.
- The IDE prompts you with the **Rename** dialog box to lead you through the options of renaming the class and the updating of code that refers to that class.
- Make the changes and click **Refactor** to apply the changes.
- This sequence of clicks might seem unnecessary if you have just a single class in your project, but it is very useful when your changes affect other parts of your code in larger projects.
- For a more thorough guide to the features of the NetBeans IDE, see the NetBeans Documentation page.

1.2 NetBeans IDE Basics

The IDE provides several tools to simplify the process of building GUIs. The basic components of NetBeans IDE include the following main components along with others:-

- The Palette window
- The GUI Design Area /Code editor,
- The Property Editor, and
- The Inspector/Navigator window.

To explorer the GUI capabilities of NetBeans IDE start by adding JFrame form to your project.
Now right-click the **CGPA Calculator** name and choose New -> JFrame Form (JFrame is the Swing class responsible for the main frame for your application.)
Next, type “**CourseRegisterForm**” as the class name, and **cgpacalculator** as the package name



Class Name: CourseRegisterForm
Project: CGPA Calculator
Location: Source Packages
Package: cgpacalculator
Created File: D:\Projects\java\CGPA Calculator\src\cgpacalculator\CourseRegisterForm.java

Fig : JFrame Form Creation wizard

When the IDE finishes loading, the right pane will display a design-time, graphical view of the **CourseRegisterForm**. It is on this screen that you will visually drag, drop, and manipulate the various Swing components.

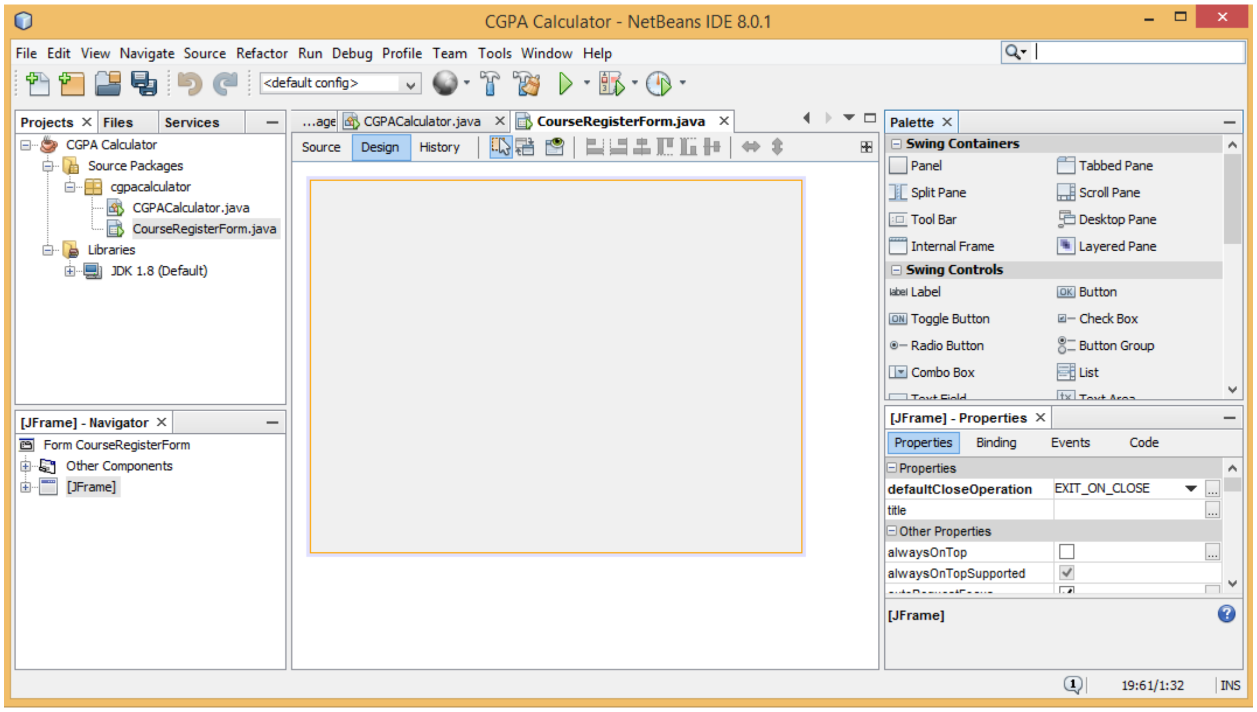


Fig : NetBeans IDE Basics

1.2.1 The Palette

The Palette contains all of the components offered by the Swing API. A list contains all the components that can be added to forms. You can customize the window to display its contents as icons only, or as icons with component names. Such as `JLabel` is a text label, `JList` is a drop-down list, etc. in group of Container, Menu, Controls, ..

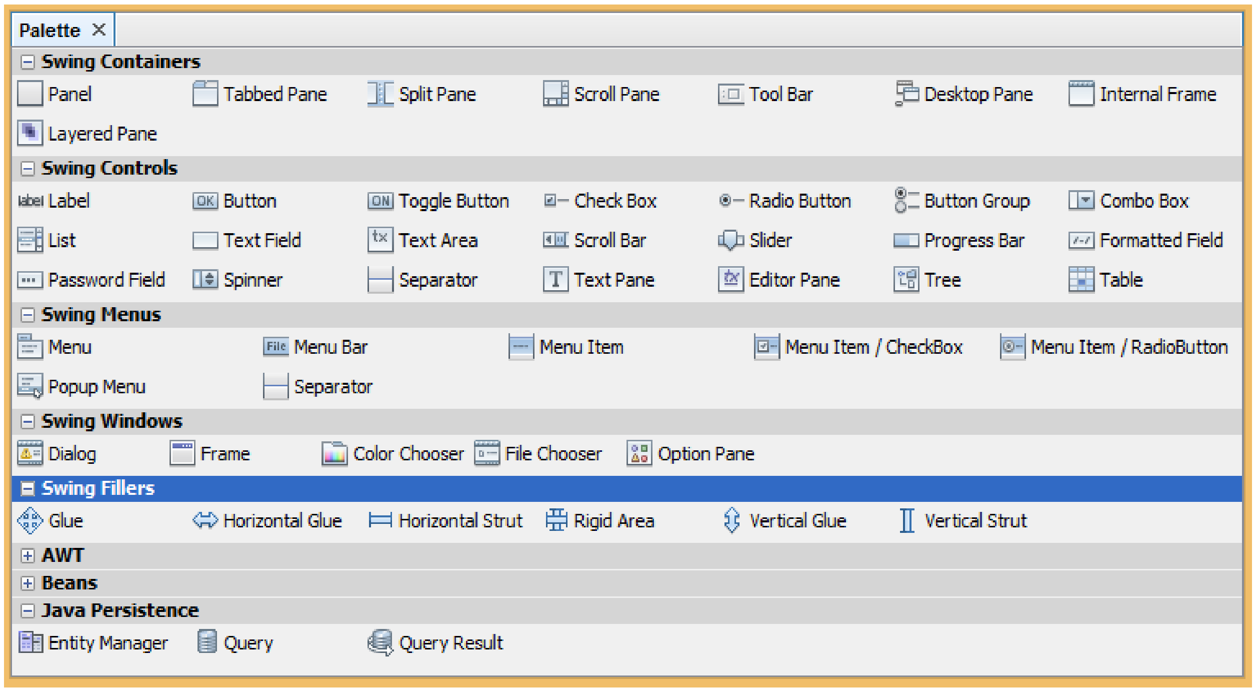


Fig : NetBeans IDE Palette window

1.2.2 The GUI Builder or Design Area

The Design Area is where you will visually construct your GUI. It is he primary workspace within which GUI design takes place in the IDE. The GUI Builder enables you to lay out forms by placing components where you want them and by providing visual feedback in the form of guidelines. It has two views: *source view*, and *design view*. Design view is the default. You can toggle between views at any time by clicking their respective tabs.

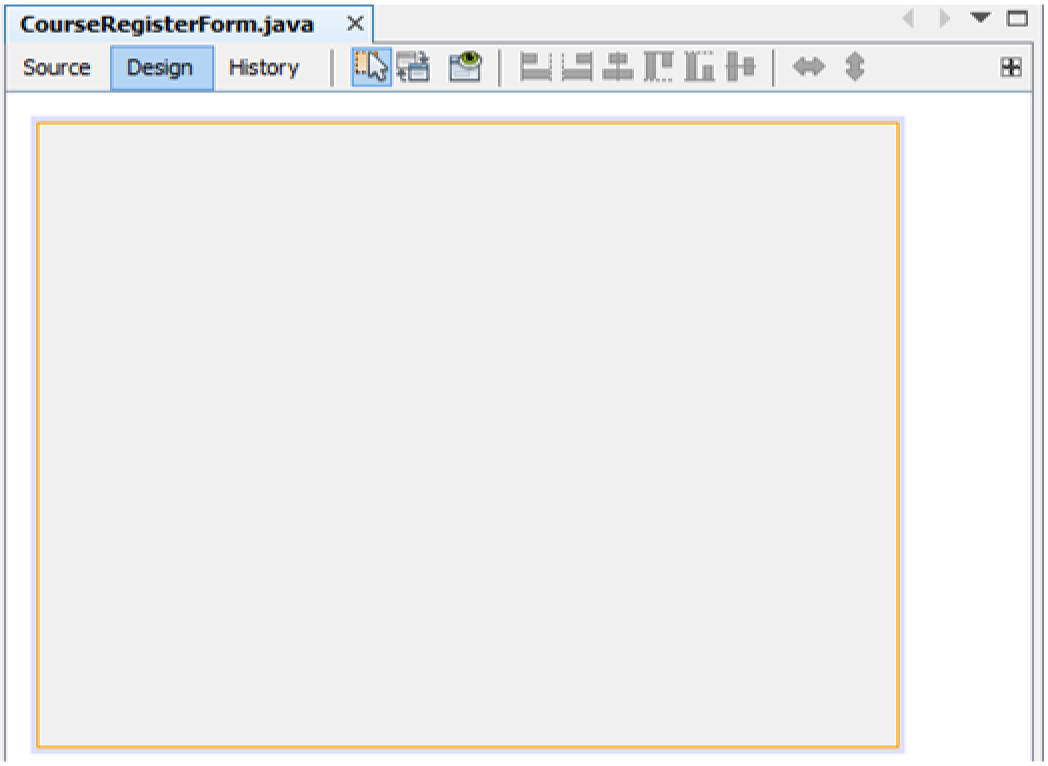
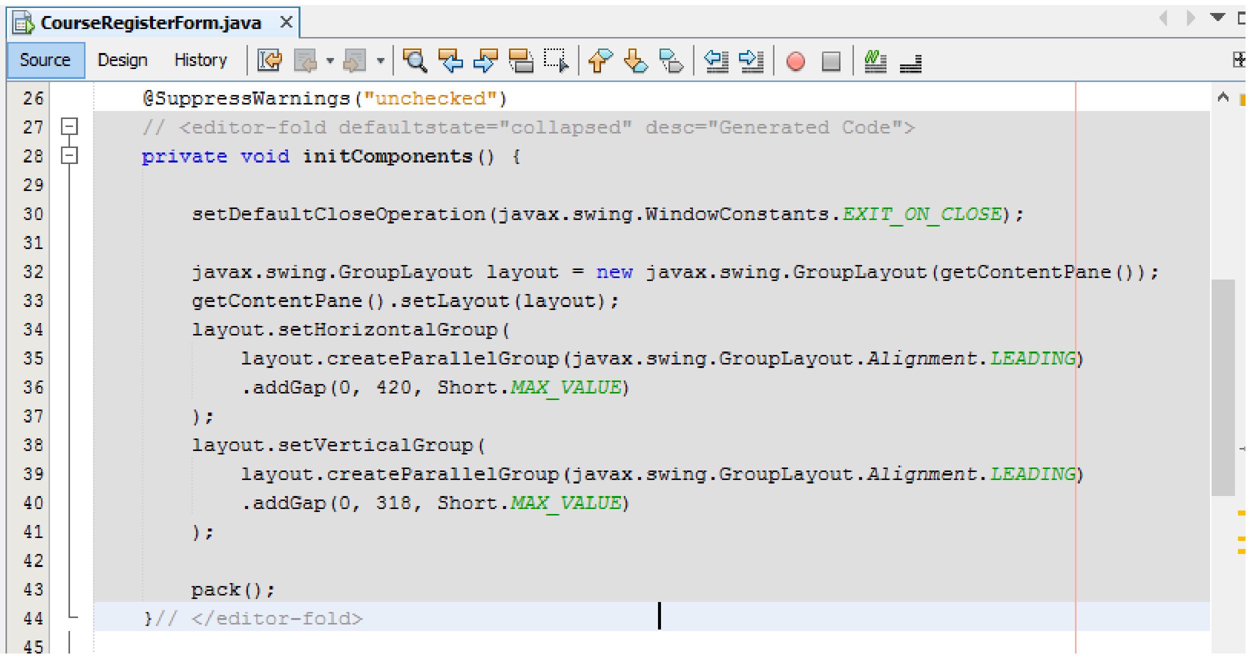


Fig : The GUI Builder or Design area



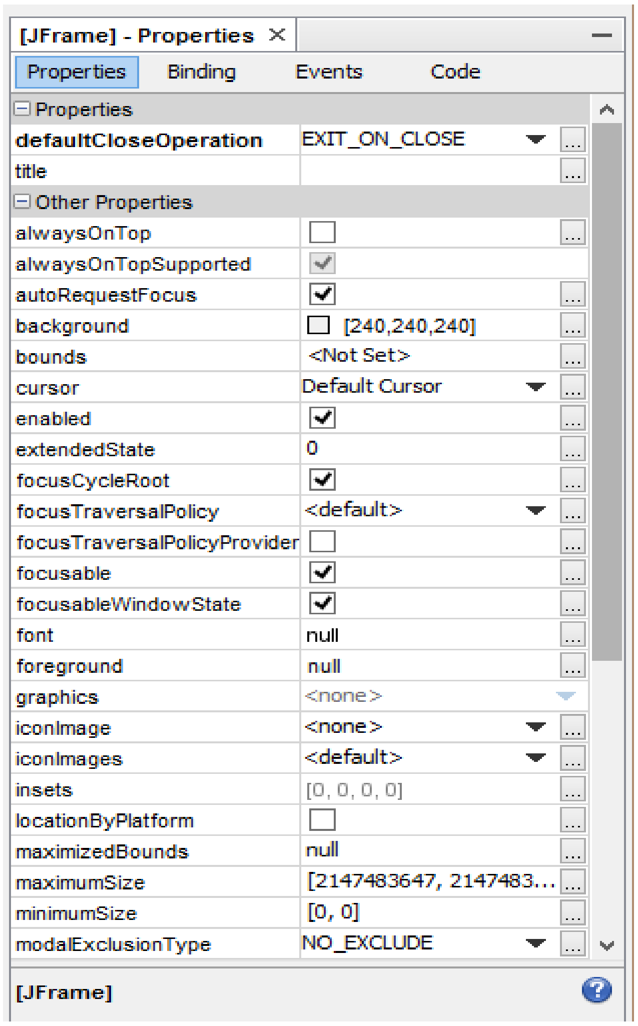
```java
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 420, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 318, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>
```

Fig : The code editor window

A quick look at the source view reveals that the IDE has created a private method named initComponents, which initializes the various components of the GUI. It also tells the application to "exit on close", performs some layout-specific tasks, and then packs the components together on screen.

1.2.3 The Property Editor

The Property Editor does what its name implies: it displays the editable settings for the currently selected component. it allows you to edit the properties of each component. The Property Editor is intuitive to use; in it you will see a series of rows — one row per property — that you can click and edit without entering the source code directly. The following figure shows the Property Editor for the newly added `JFrame` object:



Properties	
defaultCloseOperation	EXIT_ON_CLOSE
title	
Other Properties	
alwaysOnTop	☐
alwaysOnTopSupported	✓
autoRequestFocus	✓
background	☐ [240,240,240]
bounds	<Not Set>
cursor	Default Cursor
enabled	✓
extendedState	0
focusCycleRoot	✓
focusTraversalPolicy	<default>
focusTraversalPolicyProvider	☐
focusable	✓
focusableWindowState	✓
font	null
foreground	null
graphics	<none>
iconImage	<none>
iconImages	<default>
insets	[0, 0, 0, 0]
locationByPlatform	☐
maximizedBounds	null
maximumSize	[2147483647, 2147483...
minimumSize	[0, 0]
modalExclusionType	NO_EXCLUDE

[JFrame]

Fig : JFrame properties window

The screenshot above shows the various properties of this object, such as background color, foreground color, font, and cursor.

1.2.4 Navigator window

It displays a tree hierarchy of all components contained in the currently opened form. Displayed items include visual components and containers, such as buttons, labels, menus, and panels, as well as non-visual components such as timers and data sources.

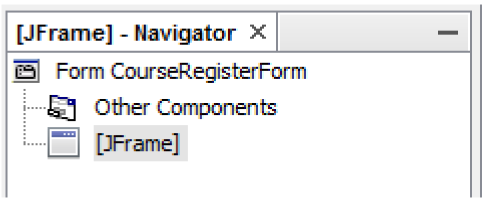


Fig : The Navigator window

1.2.5 Connection wizard

Assists in setting events between components in a form without the need of writing code manually

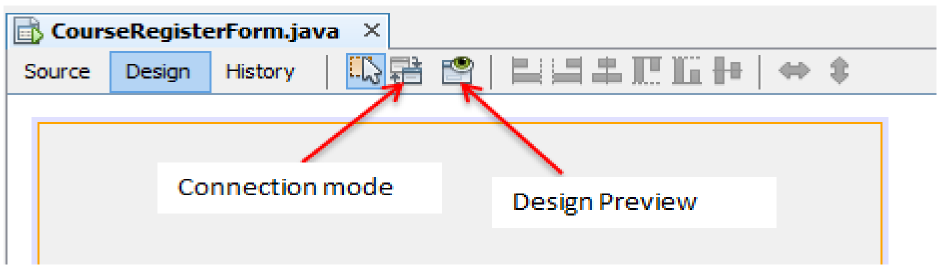


Fig : Connection mode and Design preview

1.2.6 Manager

Enables you to add, remove, and organize window components such as Swing components, AWT components, Layouts, and beans.

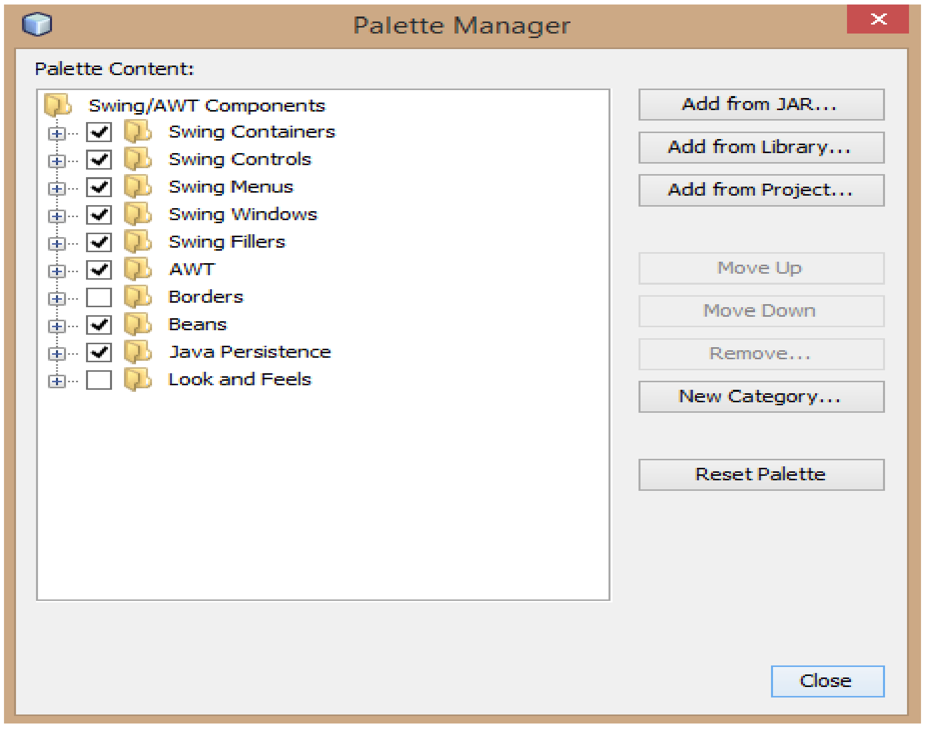


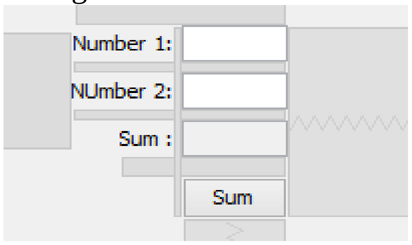
Fig : Palette manager window

In addition, the IDE provides support for the Beans Binding specification which provides a way to synchronize the values of different bean properties. This support also simplifies the creation of desktop database applications.
To access binding features, right-click a component, select Bind in the context menu, and select a property from the drop-down list.

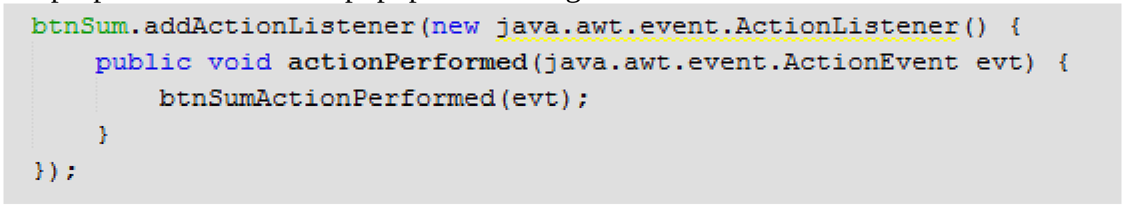
1.3 Working with the GUI Builder

The GUI Builder is a tool for designing GUIs visually. As you create and modify your GUI, the IDE automatically generates the Java code to implement the interface. GUI forms are indicated by form nodes (▤) in the Projects, Files, and Favorites windows.
When we develop Java application using NetBeans IDE, we will follow the following basic and important steps:-

1. Deign the User interface
   - Drag and drop user controls from the palette window to the form
   - Set properties for the controls
   - Change the default variable name of the control,

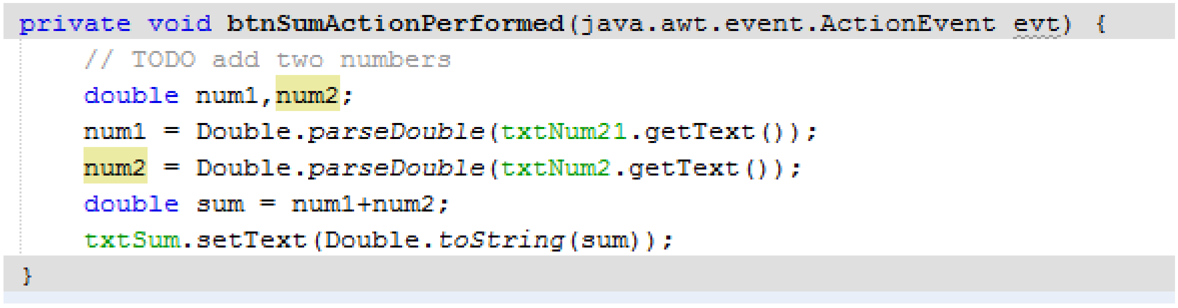


2. Create the event listeners in order to wired the controls with event
   - This java code generated by the IDE automatically, when we add event for the controls from the properties windows or popup menu using GUI Builder.



```
btnSum.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnSumActionPerformed(evt);
    }
});
```

3. Write the event handler/code that going to be executed when the event is fired



```
private void btnSumActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add two numbers
    double num1,num2;
    num1 = Double.parseDouble(txtNum21.getText());
    num2 = Double.parseDouble(txtNum2.getText());
    double sum = num1+num2;
    txtSum.setText(Double.toString(sum));
}
```

When you open a GUI form, the IDE displays it in an Editor tab with toggle buttons that enable switching between Source and Design views. The Design view enables you to work with GUI forms visually while the Source view permits the form's source code to be edited directly. Each time you select a form's Design toggle button, the Palette, Navigator, and Properties windows appear automatically. If not appear or closed unintentionally, go to **Window** =>(look for the required window)

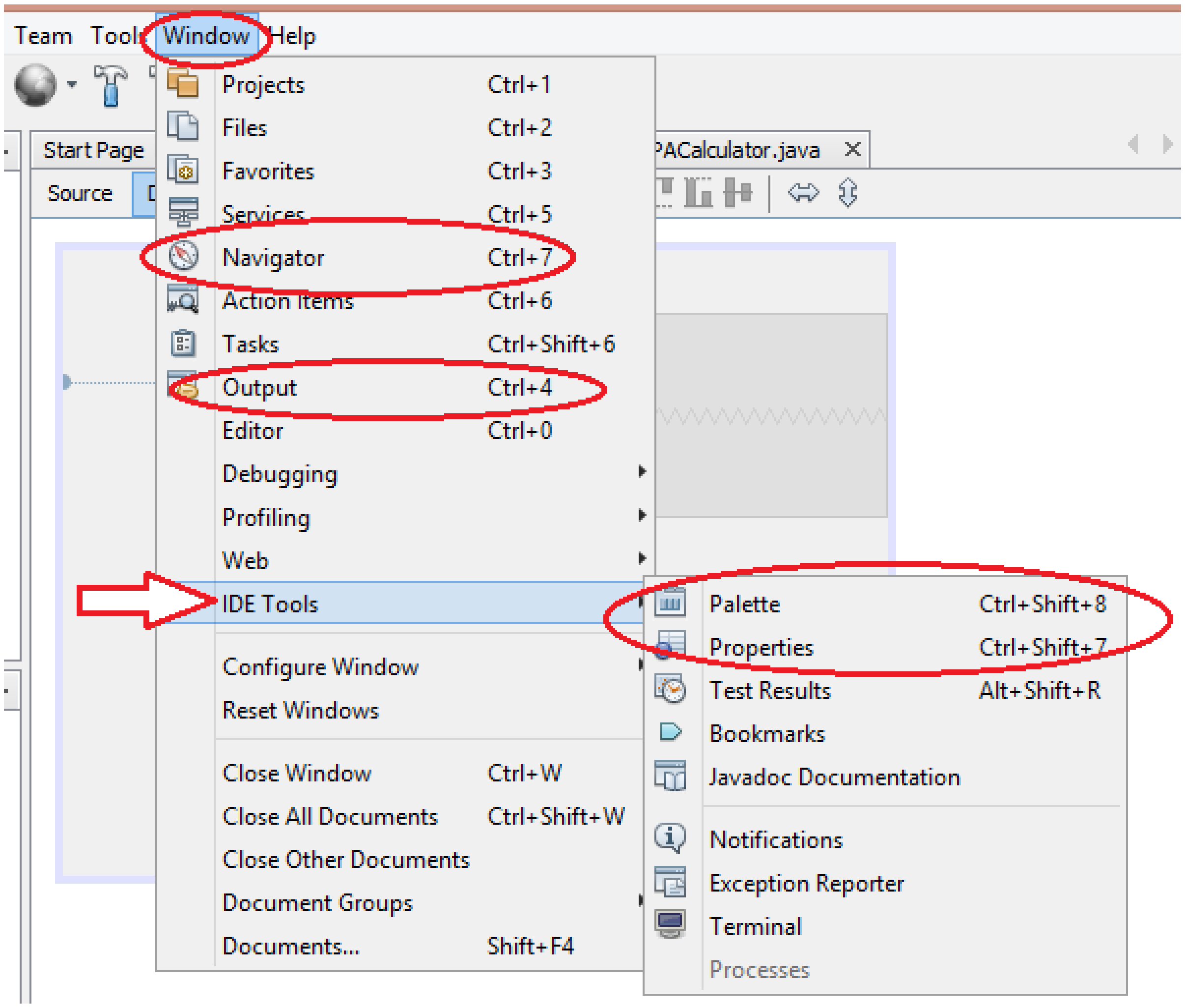


Fig : Open the required window

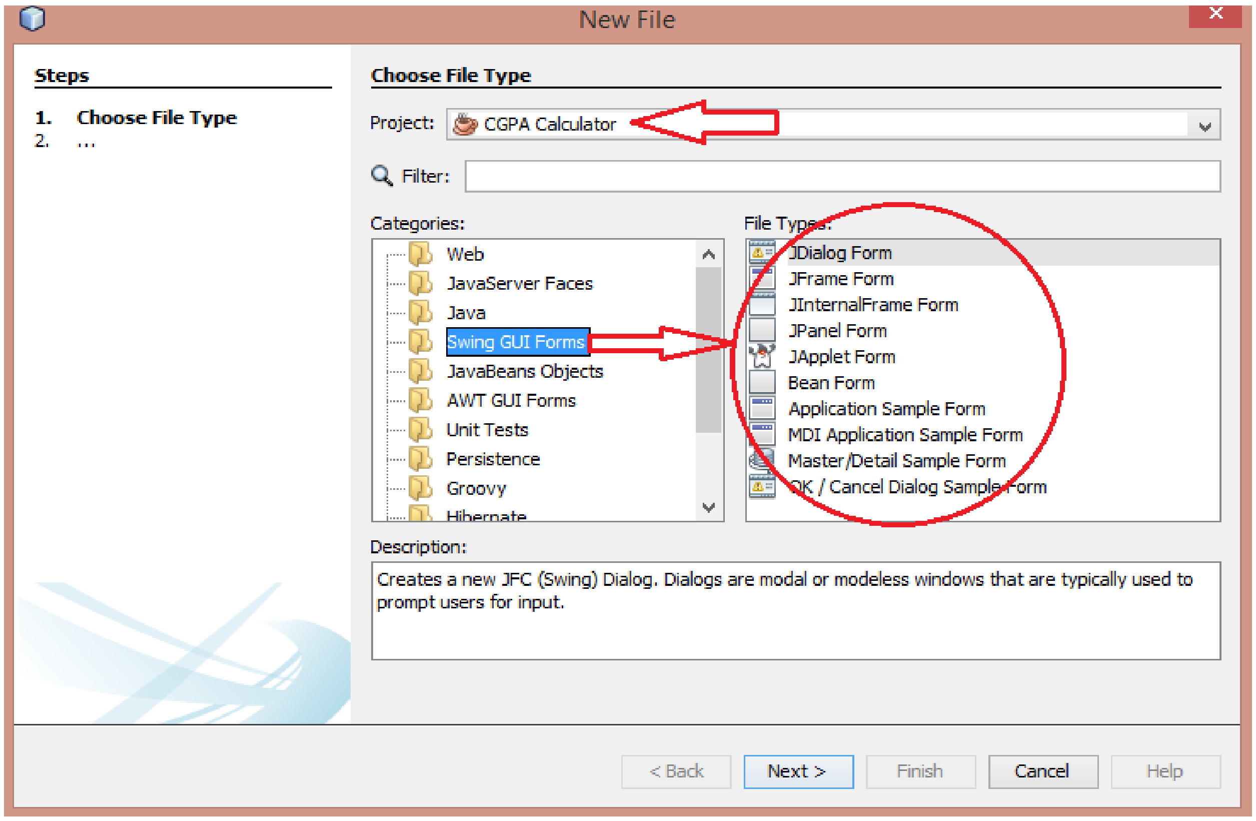
Components are typically added to a form using the window and arranged in the GUI Builder workspace. As you work, the GUI Builder automatically displays guidelines suggesting preferred alignment and anchoring for the components you add. Use the **Navigator window** in conjunction with the **Properties window** to examine a form's component and **layout manager properties**, manage component event handlers, and define how code is generated.

1.3.1 To Create a New Form

In the IDE you can create JFC/Swing or AWT (Abstract Window Toolkit) forms, pre-built sample application skeletons, or any class that is based on the JavaBeans component architecture using the provided templates.

To create a new GUI form in an existing project:

1. Choose **File** > **New File** from the main menu.
2. In the New wizard's Project combo box, select the project for which you want to create the form.
3. Expand the **Swing GUI Forms** or AWT GUI forms node in the Categories pane and select the desired form template. Click **Next**.



Project: CGPA Calculator

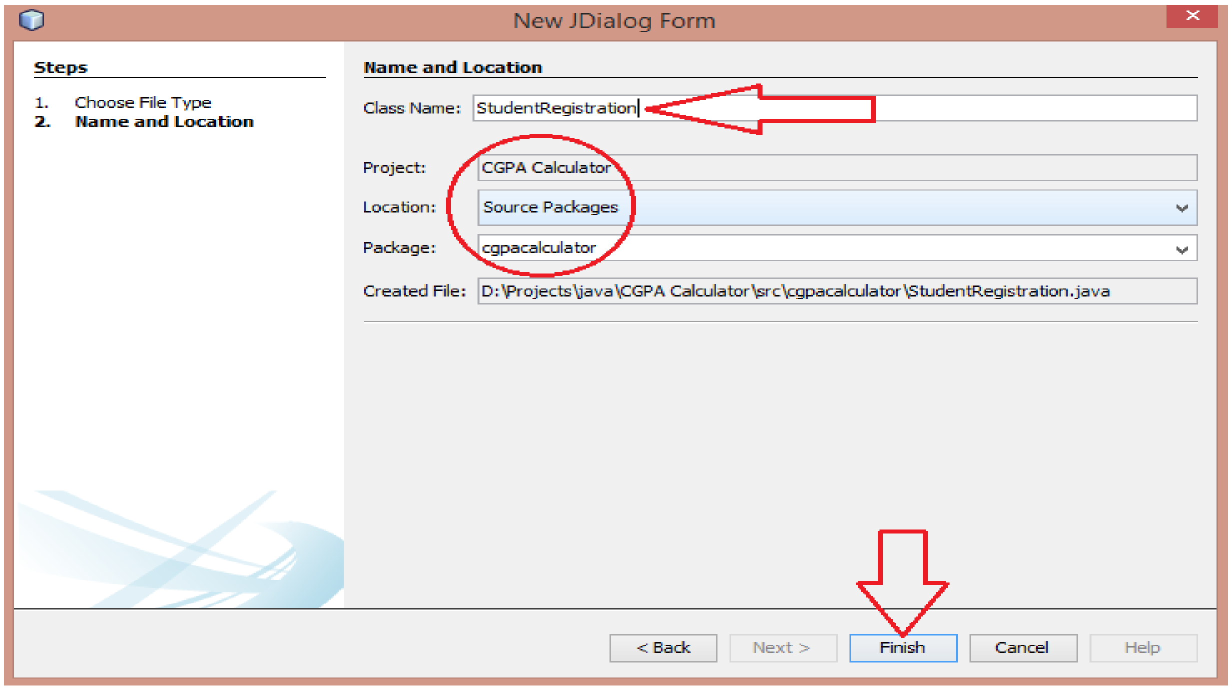
Categories: Web, JavaServer Faces, Java, Swing GUI Forms, JavaBeans Objects, AWT GUI Forms, Unit Tests, Persistence, Groovy, Hibernate

File Types: JDialog Form, JFrame Form, JInternalFrame Form, JPanel Form, JApplet Form, Bean Form, Application Sample Form, MDI Application Sample Form, Master/Detail Sample Form, OK / Cancel Dialog Sample Form

Description: Creates a new JFC (Swing) Dialog. Dialogs are modal or modeless windows that are typically used to prompt users for input.

Fig : New Form wizard

4. Enter the GUI form's class name and location. Click **Finish**.



Class Name: StudentRegistration

Project: CGPA Calculator

Location: Source Packages

Package: cgpacalculator

Created File: D:\Projects\java\CGPA Calculator\src\cgpacalculator\StudentRegistration.java

Fig : New form wizard

The IDE creates a blank form of the selected type and opens the form in the Source Editor's Design view.

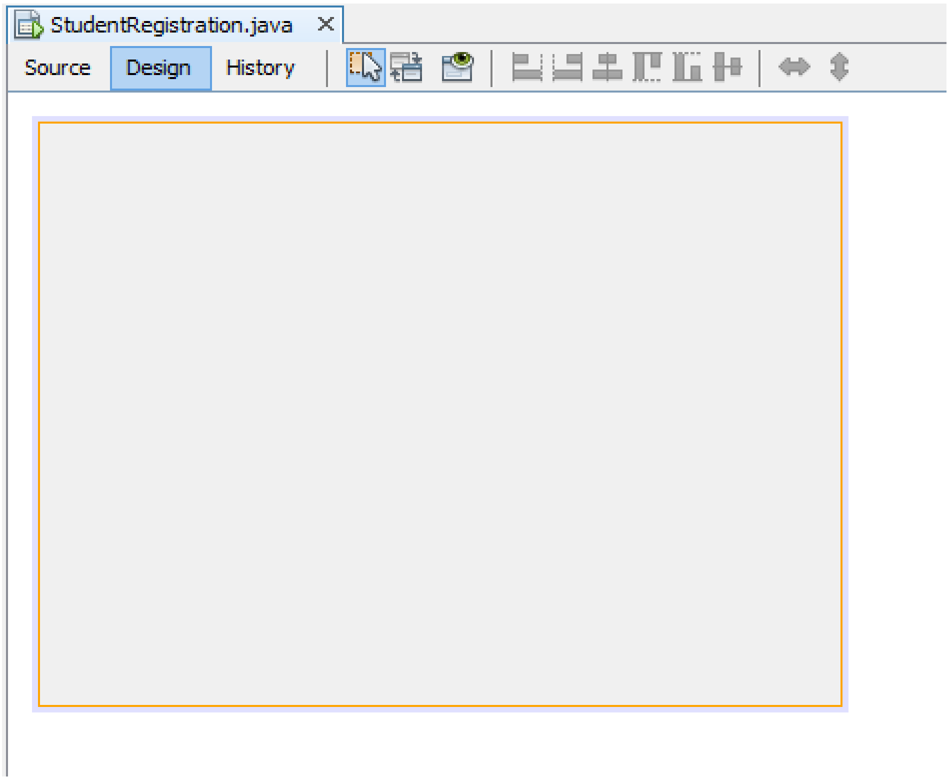


Fig : New blank Form

GUI Form Types in the New File Wizard

The following table lists the types of form templates available in the IDE. Each differs in the design time and run time look of the form, as well as in the code generated for the form's class.

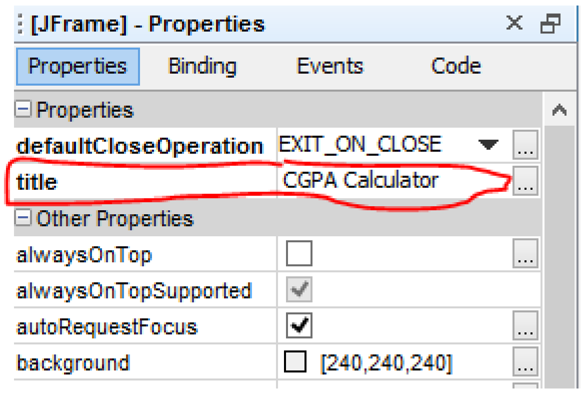
Table: NetBeans IDE Form template types

JApplet	Program run by a Java-enabled web browser or other applet viewer.
JDialog	Modal or modeless window for collecting user input.
JFrame	Top-level application window.
JInternalFrame	An internal frame that can be placed on a JDesktopPane component to create an MDI application.
JPanel	Lightweight container for holding parts of an interface. In turn, the container can be used in any other container, such as a JFrame, JPanel, JApplet, or JDialog component.
Bean Form	The template used to create a new form based on any JavaBeans component. The new form can be visual or nonvisual. In the New wizard go to the Superclass page and on the Form Superclass page specify the bean class. The bean class that you specify when creating the new form must be in the classpath and must be already compiled.

Set the Title

First, set the title of the application's `JFrame` to "CGPA Calculator", by:-

- single-clicking the `JFrame` in the Navigator window => properties window or
- Select the JFrame => properties window or
- Right Clicjk JFrame =>Property from the popup menu => properties window. Then, set its title with the Property Editor



defaultCloseOperation	EXIT_ON_CLOSE
title	CGPA Calculator
alwaysOnTop	☐
alwaysOnTopSupported	✓
autoRequestFocus	✓
background	☐ [240,240,240]

As a shortcut, you could single-click the *JFrame* of the inspector and enter its new text directly without using the property editor.

Add controls to a Form

When you add a form to your project, NetBeans displays a blank form in design view. In this view, you can use the palette window to add controls to the form.

Add a JLabel

Drag a `JLabel` from the Palette to the upper left corner of the Design Area. As you approach the upper left corner, the GUI builder provides visual cues (dashed lines) that suggest the appropriate spacing. Using these cues as a guide, drop a `JLabel` into the upper left hand corner of the window.

Add a JTextField

Drag a `JTextField` onto the Design Area. Place it to the right of the `JLabel`, again watching for visual cues that suggest an appropriate amount of spacing. Make sure that text base for this component is aligned with that of the `JLabel`. The visual cues provided by the IDE should make this easy to determine.
In the same way, by dragging, dropping and using the visual cues of the IDE you can design you user interface easily.

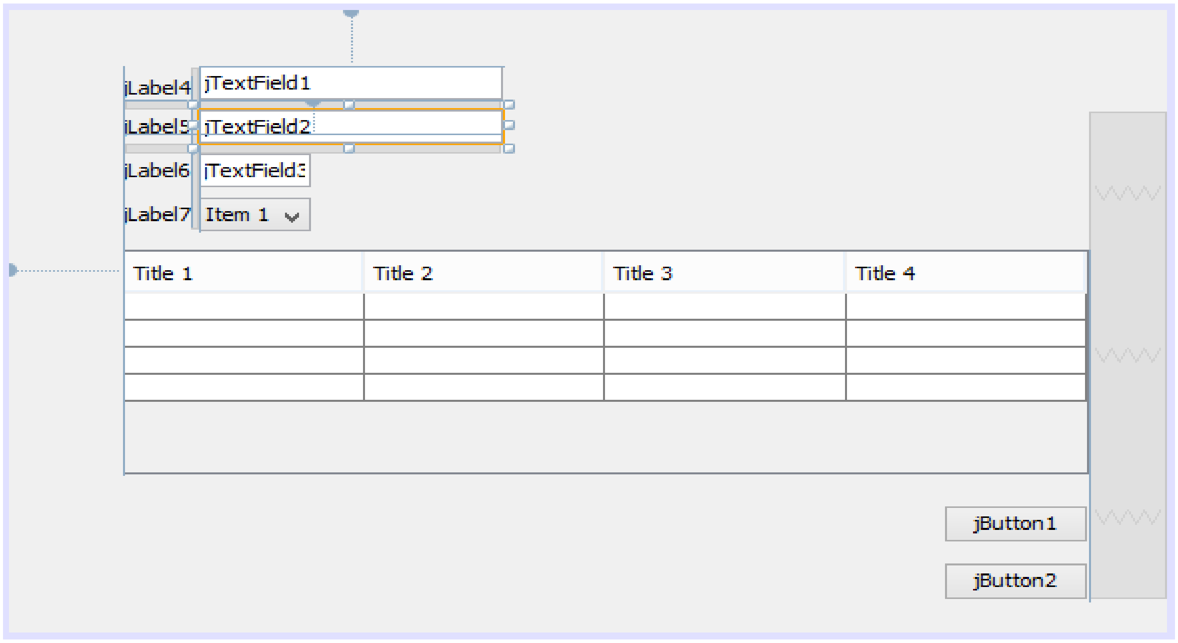
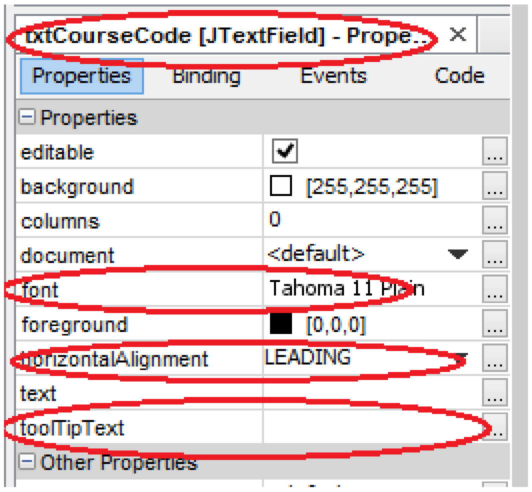


Fig : A form after some controls have been added to it

Set form and controls properties

After you add controls to a form, you can set each control's properties. These are the values that determine how a control look and work when the form is displayed. In addition you need to set some of the properties for the form itself. You can use the guide line of the IDE and the properties window to set properties for controls.

- ➢ Select the control => go to the properties window and adjust the control's properties



Properties	
editable	✓
background	[255,255,255]
columns	0
document	<default>
font	Tahoma 11 Plain
foreground	■ [0,0,0]
horizontalAlignment	LEADING
text	
toolTipText	
Other Properties	

Fig : Set control's properties

Common properties for forms

Property	Description
`title`	Sets the text that's displayed in the title bar for the frame.
`defaultCloseOperation`	Sets the action that's performed when the user clicks on the Close button in the upper right corner of the frame. The default value, EXIT_ON_CLOSE, causes the application to exit.
`resizable`	Determines whether the user can resize the frame by dragging its edges.

Common properties for controls

Property	Description
`text`	Sets the text that's displayed on the control.
`editable`	Determines whether the user can edit the text that's stored in the control. Typically used for text fields and other controls that contain text.
`enabled`	Determines whether a control is enabled or disabled.
`focusable`	Determines whether the control accepts the focus when the user presses the Tab key to move the focus through the controls on the form.
`horizontalAlignment`	Determines how the text on a button or in a text field is aligned: left, right, center, trailing, or leading.
`mnemonic`	Specifies a keyboard character that allows the user to quickly access the control by holding down the Alt key and pressing the specified character. Typically used for buttons.
`preferredSize`	Sets the width and height in pixels for the control.

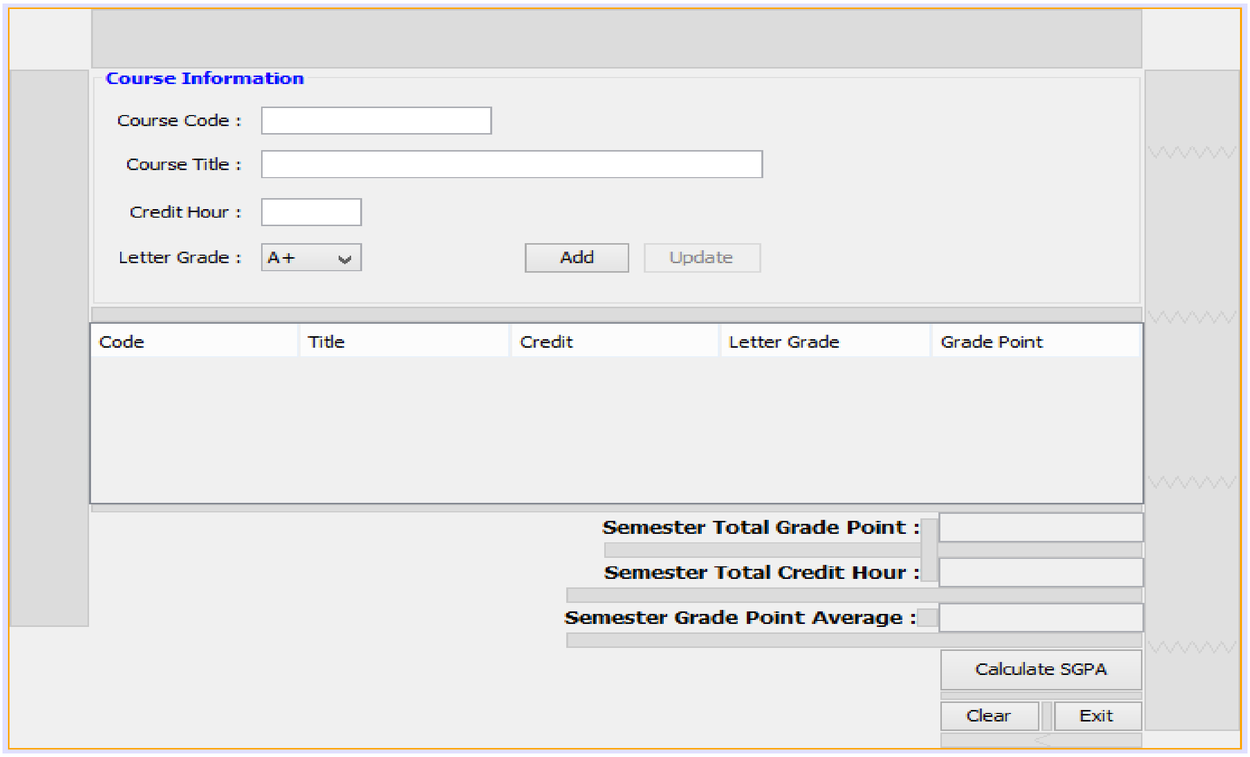
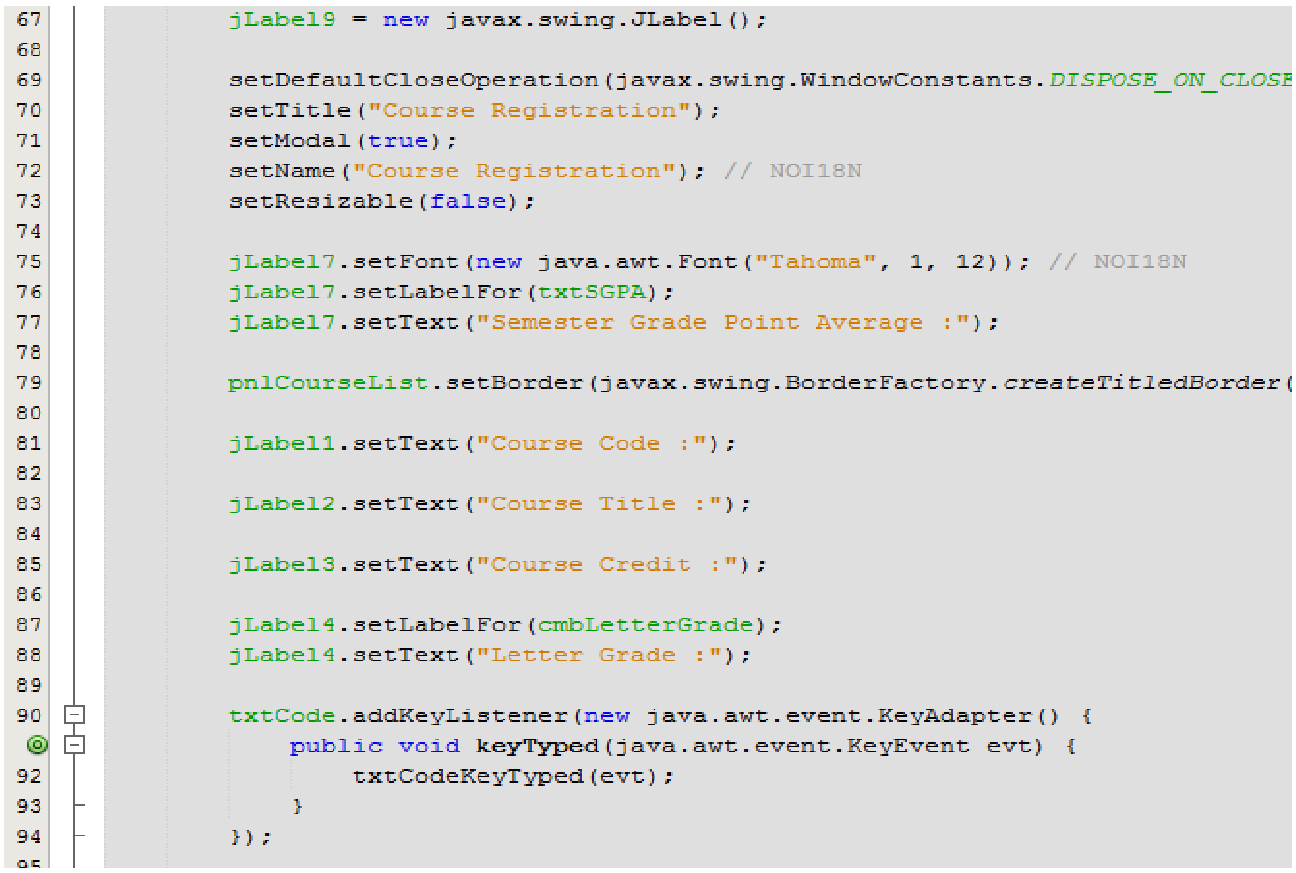


Fig : A form after the properties have been set.

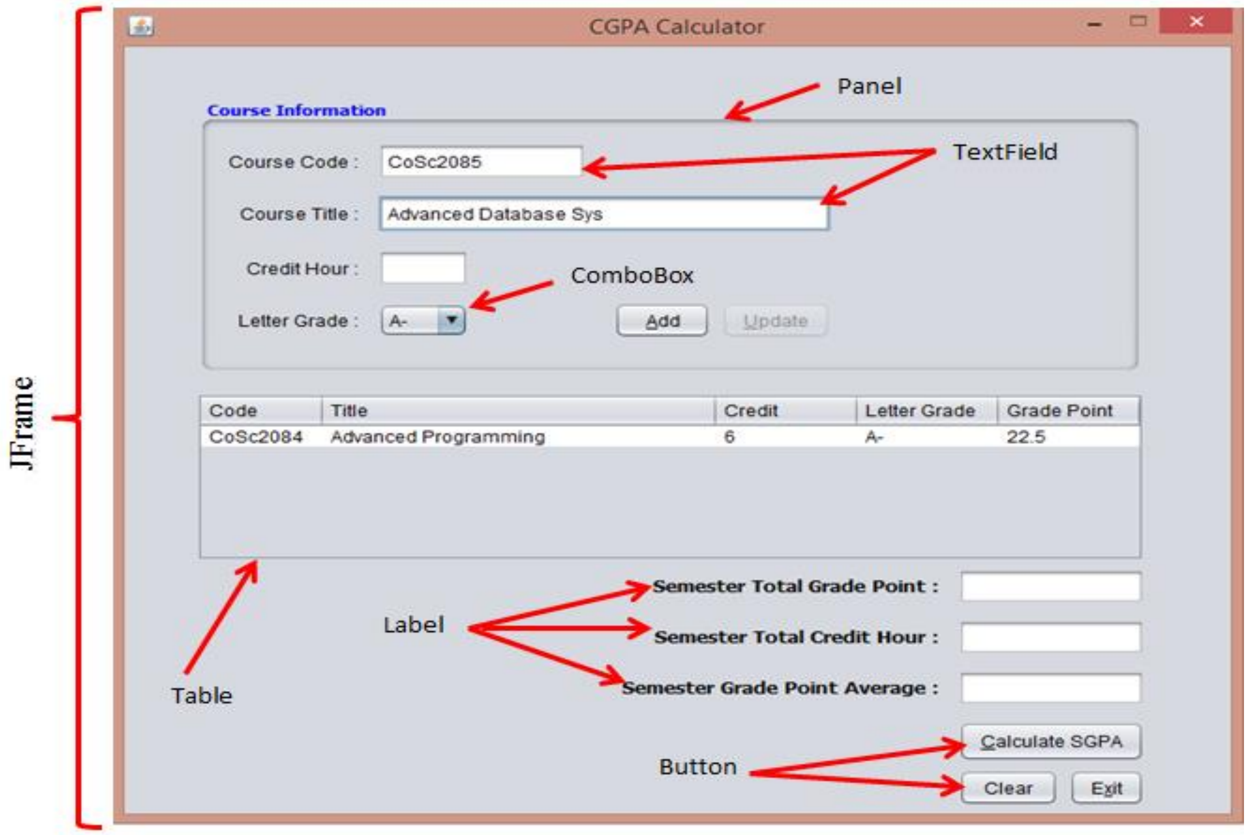
The GUI portion of this application is now complete! If the NetBeans IDE has done its job, you should feel that creating this GUI was a simple task. But take a minute to click on the source tab; you might be surprised at the amount of code that has been generated.

To see the code in its entirety, scroll up and down within the IDE as necessary. You can expand or collapse certain blocks of code (such as method bodies) by clicking the + or - symbol on the left-hand side of the source editor.



```
        jLabel9 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE
        setTitle("Course Registration");
        setModal(true);
        setName("Course Registration"); // NOI18N
        setResizable(false);

        jLabel7.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel7.setLabelFor(txtSGPA);
        jLabel7.setText("Semester Grade Point Average :");

        pnlCourseList.setBorder(javax.swing.BorderFactory.createTitledBorder(

        jLabel1.setText("Course Code :");

        jLabel2.setText("Course Title :");

        jLabel3.setText("Course Credit :");

        jLabel4.setLabelFor(cmbLetterGrade);
        jLabel4.setText("Letter Grade :");

        txtCode.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                txtCodeKeyTyped(evt);
            }
        });
```

Fig : Auto generated UI design code



Course Code : CoSc2085

Course Title : Advanced Database Sys

Credit Hour :

Letter Grade : A-

Code	Title	Credit	Letter Grade	Grade Point
CoSc2084	Advanced Programming	6	A-	22.5

Semester Total Grade Point :

Semester Total Credit Hour :

Semester Grade Point Average :

Fig : The user interface for CGPA Calculator application

1.3.2 Adding a code to a Form

After you add controls to a form and set their properties you can run the form and it should display properly. However, you will not be able to interact with the controls on the forms until you add the required java code, i.e. adding the application logic.

Setting the variable name for the controls

When you add a control to a form, NetBeans creates a generic variable name for the control. For example, it uses jTextField1 for the name of the first JTextField control you add to the form, jTextField2 for the name of the second JTextField control you add, and so on.

Before you write the code that uses any of the controls on a form, you should change this generic name to meaningful names that are easier to remember and use.

To set variable name:

- Select the control => Properties window => Code tab/button => edit the default variable name or
- Right click on the control => select change variable name from the popup menu => edit the default variable name or
- Right click on the control in the navigator window => select change variable name from the popup menu =>

edit the default variable name

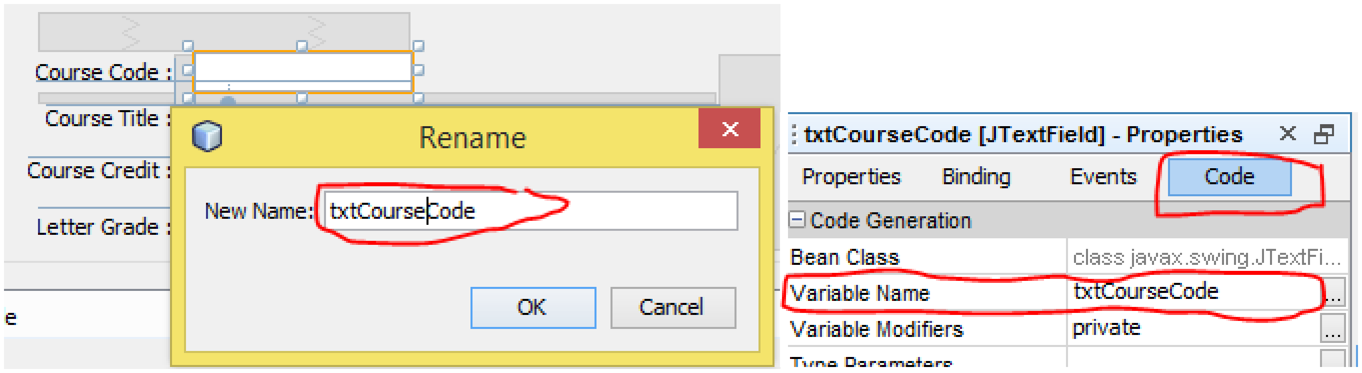


Fig :Change the defualt variable name of a control

Register the Event Listeners and create an event handler for a controls

When an end-user interacts with a Swing GUI component (such as clicking the Save button), that component will generate a special kind of object — called an *event object* — which it will then broadcast to any other objects that have previously registered themselves as *listeners* for that event.

An Event handler is a special type of method that responds to an event that is triggered when a user interacts with the controls. For example, the most common type of event handler is a method that is executed when a user clicks on a button. For an event handler to work, it must be connected or wired to the event listener. This is known as event wiring and it is generated automatically when you use NetBeans to generate an event handler.

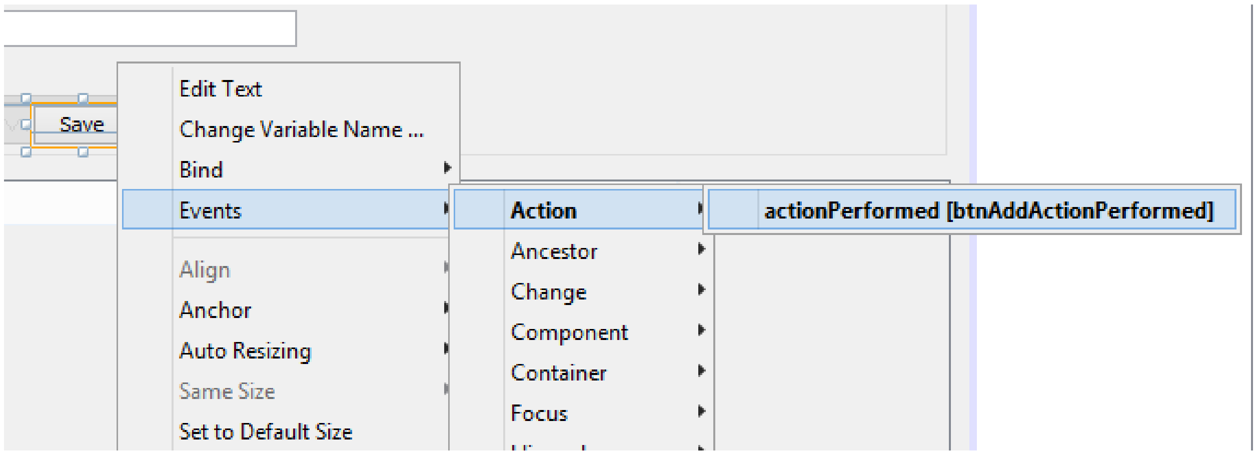
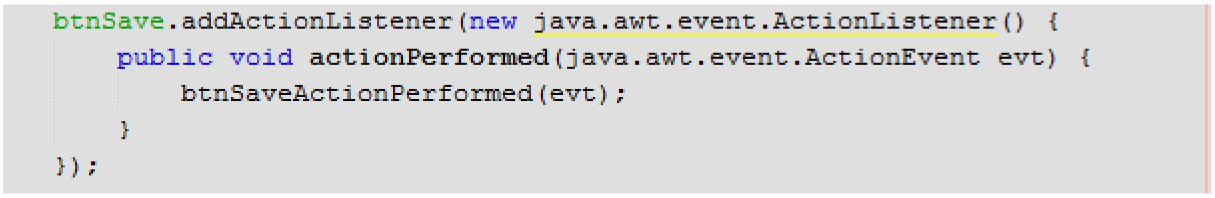


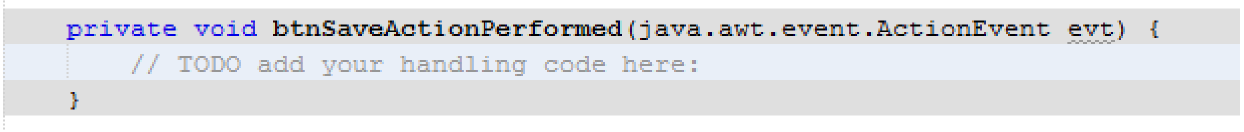
Fig : Creating event and event handler

The code generated that wires the event handler to the event



```
btnSave.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnSaveActionPerformed(evt);
    }
});
```

The code that is generated from the **actionPerformed** event of the Save button

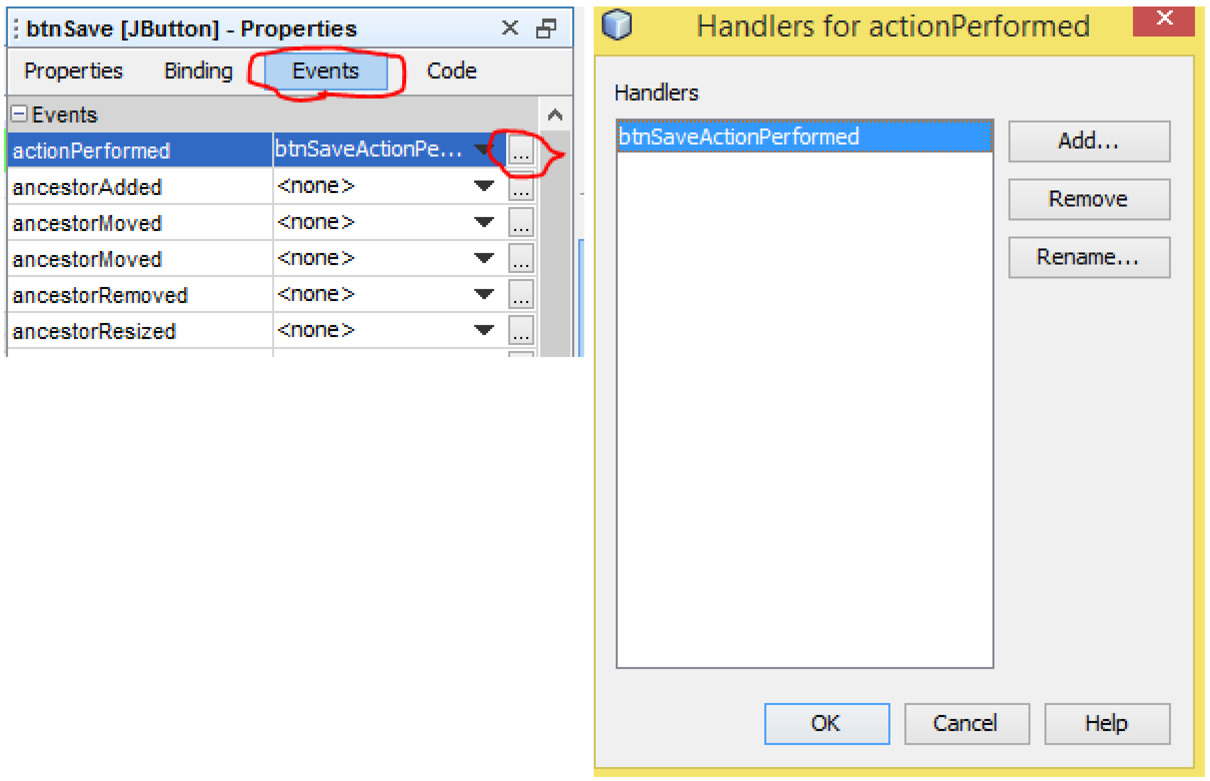


```
private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

How to rename and remove an event handler

In some cases, you'll need to rename or remove an event handler that you have generated. To do that use the Hndler dialog box:

- Select the control => properties window => click the event button => click on the ellipsis button that you want to work with
- The use the handler dialog window to rename or remove the event handler



btnSave [JButton] - Properties

Events	
actionPerformed	btnSaveActionPe...
ancestorAdded	<none>
ancestorMoved	<none>
ancestorMoved	<none>
ancestorRemoved	<none>
ancestorResized	<none>

Handlers for actionPerformed

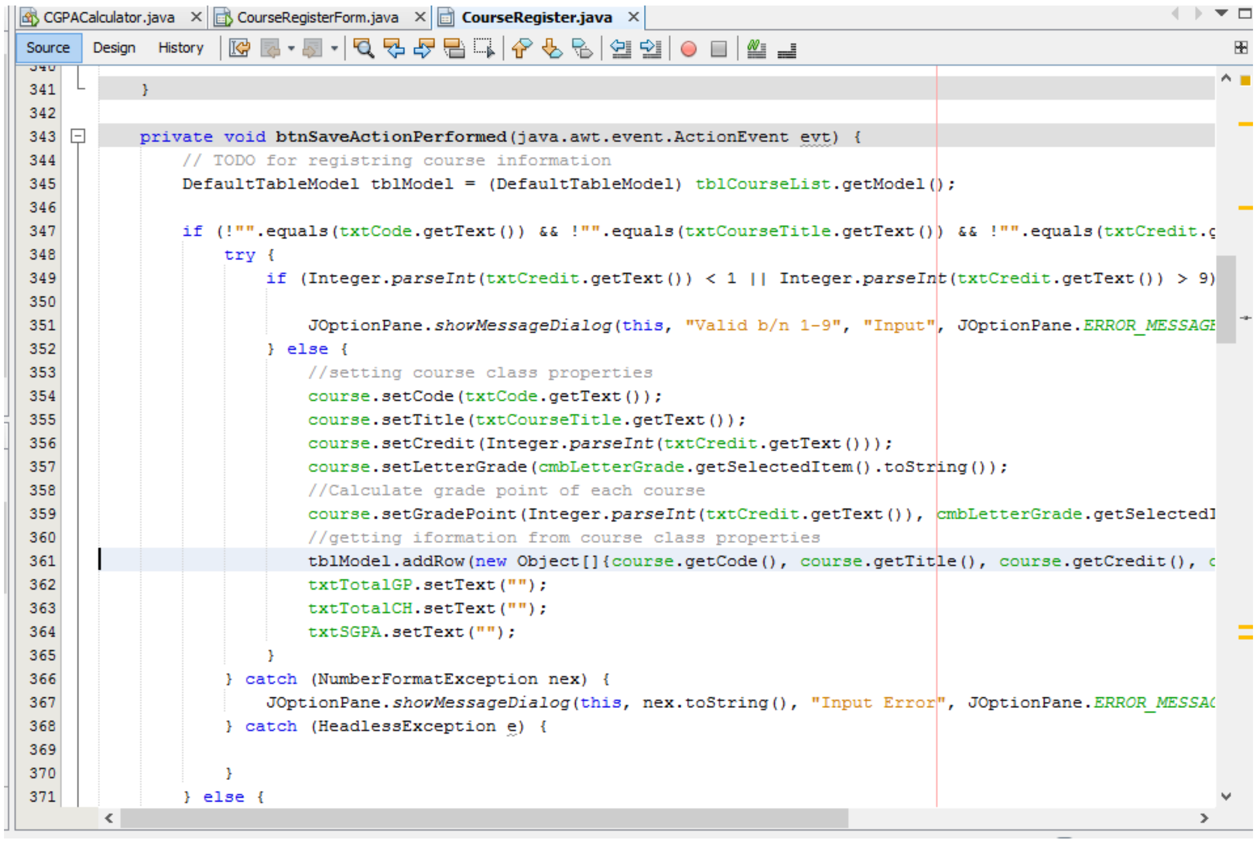
Handlers

btnSaveActionPerformed

Fig : Rename or remove an event handler

Entering the application logic/code to a control

You can use the code editor to enter the code for an event handler just as you enter any other code.to refer to the control on the form, you use the variable names that you assigned to the controls.



```java
    }

    private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO for registring course information
        DefaultTableModel tblModel = (DefaultTableModel) tblCourseList.getModel();

        if (!"".equals(txtCode.getText()) && !"".equals(txtCourseTitle.getText()) && !"".equals(txtCredit.g
            try {
                if (Integer.parseInt(txtCredit.getText()) < 1 || Integer.parseInt(txtCredit.getText()) > 9)

                    JOptionPane.showMessageDialog(this, "Valid b/n 1-9", "Input", JOptionPane.ERROR_MESSAGE
                } else {
                    //setting course class properties
                    course.setCode(txtCode.getText());
                    course.setTitle(txtCourseTitle.getText());
                    course.setCredit(Integer.parseInt(txtCredit.getText()));
                    course.setLetterGrade(cmbLetterGrade.getSelectedItem().toString());
                    //Calculate grade point of each course
                    course.setGradePoint(Integer.parseInt(txtCredit.getText()), cmbLetterGrade.getSelectedI
                    //getting iformation from course class properties
                    tblModel.addRow(new Object[]{course.getCode(), course.getTitle(), course.getCredit(), c
                    txtTotalGP.setText("");
                    txtTotalCH.setText("");
                    txtSGPA.setText("");
                }
            } catch (NumberFormatException nex) {
                JOptionPane.showMessageDialog(this, nex.toString(), "Input Error", JOptionPane.ERROR_MESSAC
            } catch (HeadlessException e) {

            }
        } else {
```

Fig : the source code for btnSaveActionPerformed event handler

Common methods for controls

Method	Description
`getText()`	Returns the text in the control as a string. Used for text fields and other controls that contain text.
`setText(String)`	Sets the text in the control to the specified string. Used for text fields and other controls that contain text.
`requestFocusInWindow()`	Moves the focus to the control.
`setEditable(boolean)`	If the boolean value is true, the control is editable. Otherwise, it's not. Used for text fields and other controls that contain text.
`setEnabled(boolean)`	If the boolean value is true, the control is enabled so the user can interact with it. Otherwise, it's disabled.
`setFocusable(boolean)`	If the boolean value is true, the control can receive the focus. Otherwise, it can't.
`selectAll()`	Selects all the text in a control. Used for text fields and other controls that contain text.

`setVisible(boolean)`	Shows this component if the boolean value is true. Otherwise, this method hides the component.
`setLocationRelativeTo(component)`	Sets the location of this component relative to the specified component. If the component is null, this method centers the frame on the screen. Otherwise, it centers the frame on the specified component.

The code that's generated by NetBeans creates a new thread for the form, which is a single follow execution through the problem.

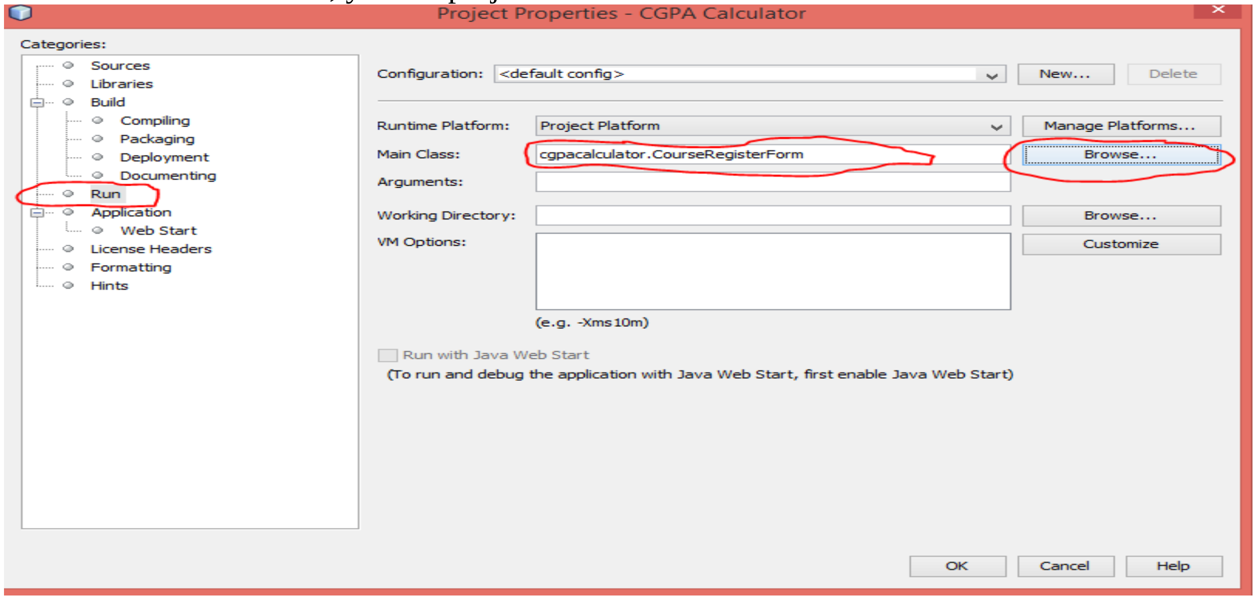
```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new CourseRegister().setVisible(true);
        }
    });
}
```

Fig : A main method for a form that's generated by NetBeans

Run the Application

Running the application is simply a matter of choosing Run -> Run Main Project within the NetBeans IDE. The first time you run this application, you will be prompted with a dialog asking to set `CourseRegister` as the main class for this project. Click the OK button, and when the program finishes compiling, you should see the application running in its own window. If you enable to create a main class when you create your project, change the main class for your project. To do that follow:

- Right click on the project => select properties => in the properties window select Run => Browse the class that contain the main method, you the project start from it.



Configuration: <default config>

Runtime Platform: Project Platform

Main Class: cgpacalculator.CourseRegisterForm

Arguments:

Working Directory:

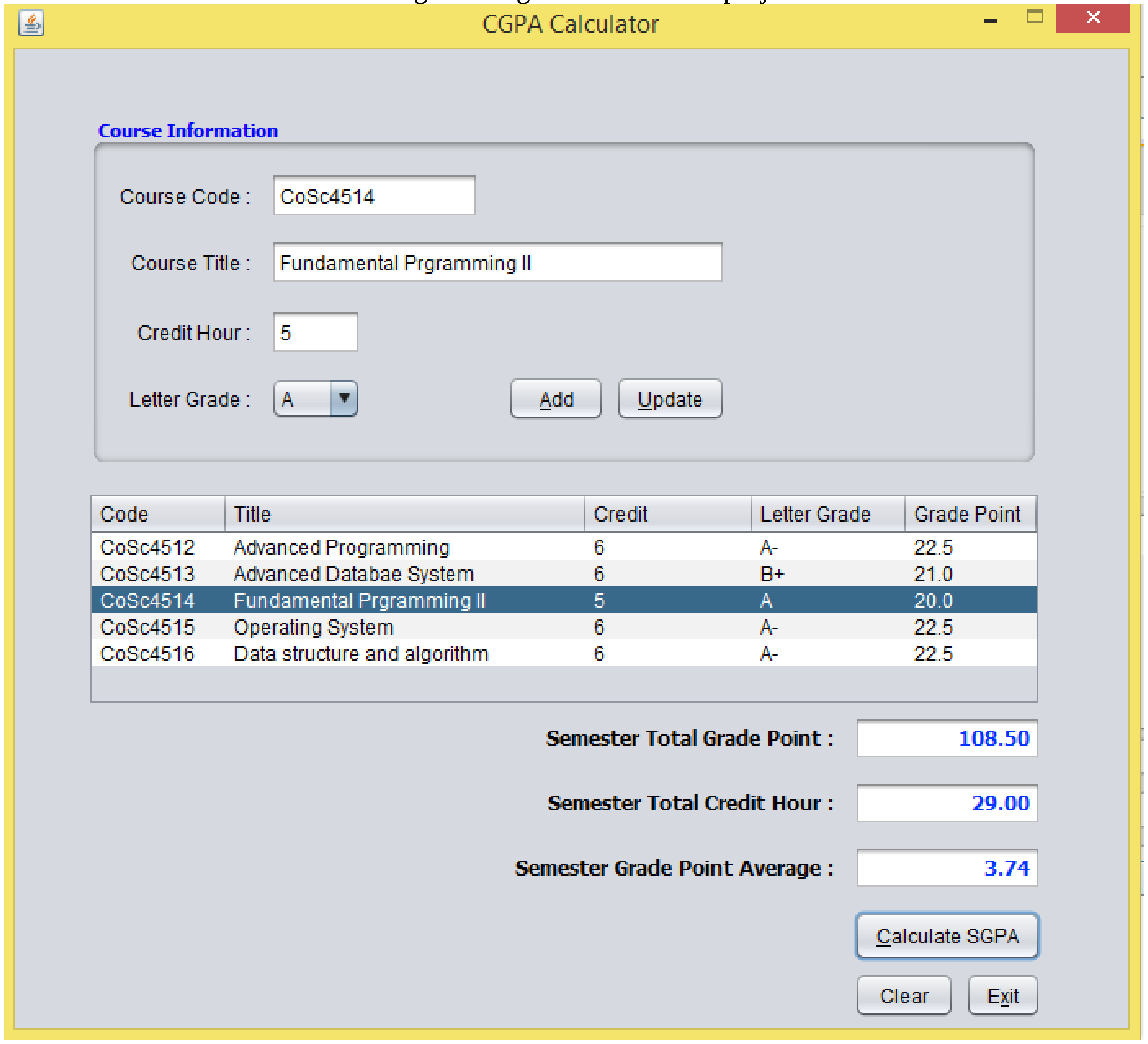
VM Options:

(e.g. -Xms10m)

Run with Java Web Start

(To run and debug the application with Java Web Start, first enable Java Web Start)

Fig : Setting main class for a project



Course Code : CoSc4514

Course Title : Fundamental Prgramming II

Credit Hour : 5

Letter Grade : A

Code	Title	Credit	Letter Grade	Grade Point
CoSc4512	Advanced Programming	6	A-	22.5
CoSc4513	Advanced Databae System	6	B+	21.0
CoSc4514	Fundamental Prgramming II	5	A	20.0
CoSc4515	Operating System	6	A-	22.5
CoSc4516	Data structure and algorithm	6	A-	22.5

Semester Total Grade Point : 108.50

Semester Total Credit Hour : 29.00

Semester Grade Point Average : 3.74

Fig : The CGPA Calculator application that calculate semester Grade Point Average