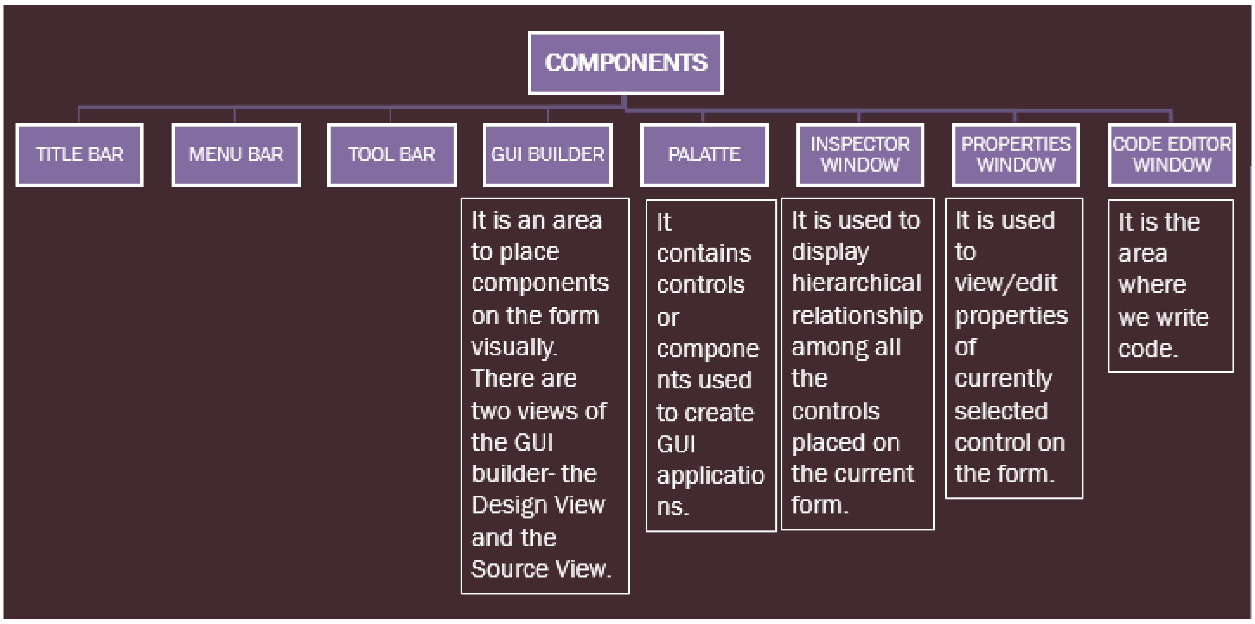
GETTING STARTED WITH IDE PROGRAMMING

NET BEANS IDE

- It is used to create java applications using the efficient GUI builder.
- IDE is an acronym for Integrated Development Environment which is a work environment that integrates all tools necessary for Application Development and makes them available as part of one environment.
- GUI is an acronym for Graphical User Interface which is an interface that allows us to interact with the various components through visual elements including pictures, graphical icons, symbols and visual indicators.

COMPONENTS OF NET BEANS IDE

The components of net beans IDE are illustrated as follows:



GUI BUILDER: It is an area to place components on the form visually. There are two views of the GUI builder- the Design View and the Source View.

PALATTE: It contains controls or components used to create GUI applications.

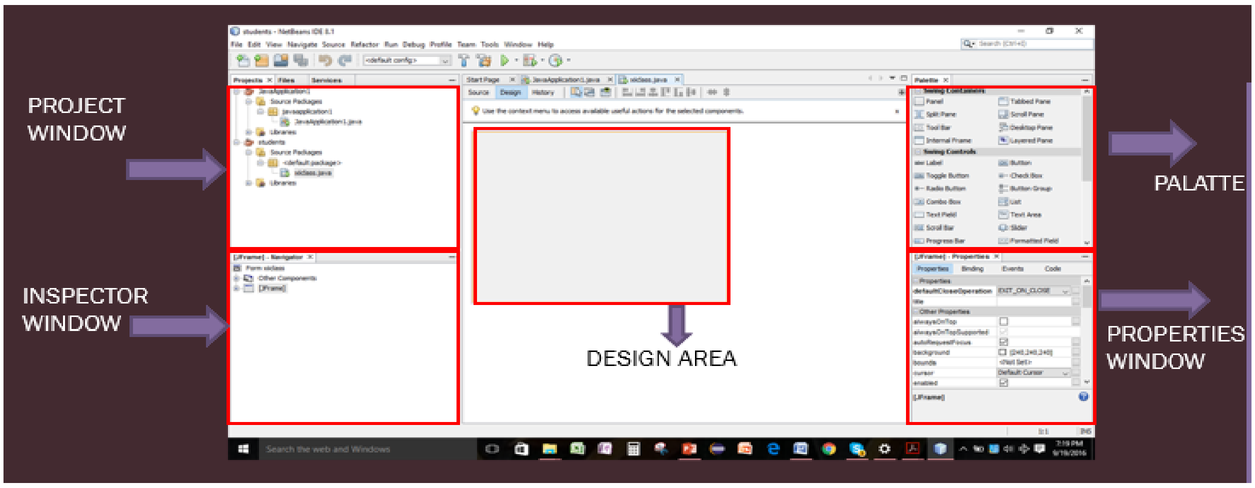
INSPECTOR WINDOW: It is used to display hierarchical relationship among all the controls placed on the current form.

PROPERTIES WINDOW: It is used to view/edit properties of currently selected control on the form.

CODE EDITOR WINDOW: It is the area where we write code.

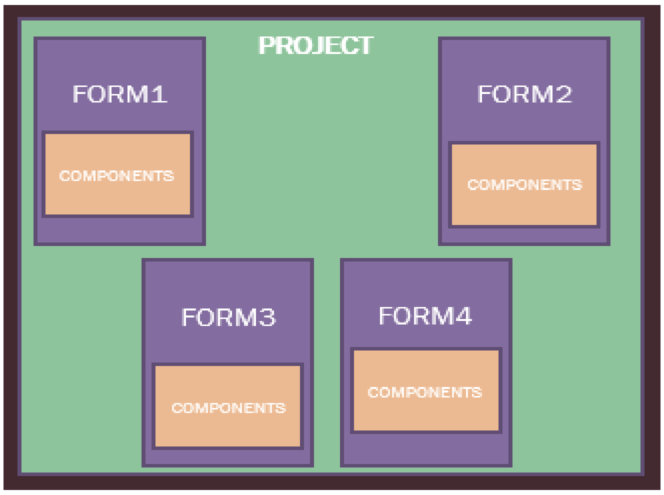
NET BEANS GUI INTERFACE

The GUI interface of Java Net Beans looks like this:



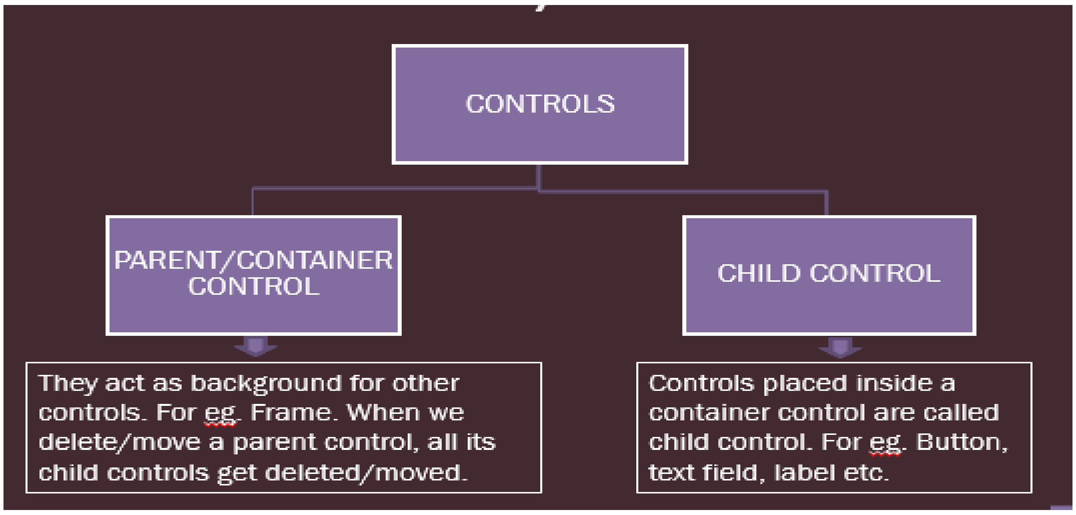
PROJECT, FORMS AND COMPONENTS

- ❑ Each application is treated as a Project in NetBeans.
- ❑ Each project can have one or multiple forms.
- ❑ Each form can have one or more components.



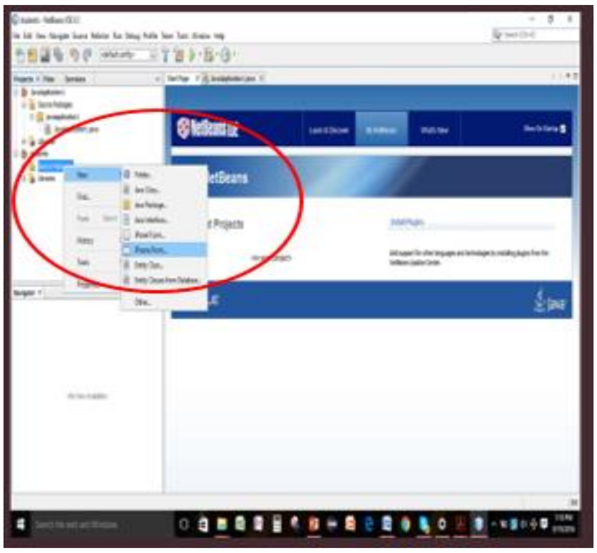
COMPONENTS/CONTROLS

There are two types of controls: Parent control and child control.


CONTROLS
PARENT/CONTAINER CONTROL
CHILD CONTROL
They act as background for other controls. For eg. Frame. When we delete/move a parent control, all its child controls get deleted/moved.
Controls placed inside a container control are called child control. For eg. Button, text field, label etc.

JFRAME FORM

Forms are used to accept data (input) from users and respond to actions like clicking on a button.



[Click to view video on Introduction to Java Netbeans]

OPTION PANE

Option Pane is used when we want to request information from the user, display information to the user or a combination of both. It requires the following import statement at the top of the program.

import javax.swing.JOptionPane;

OR

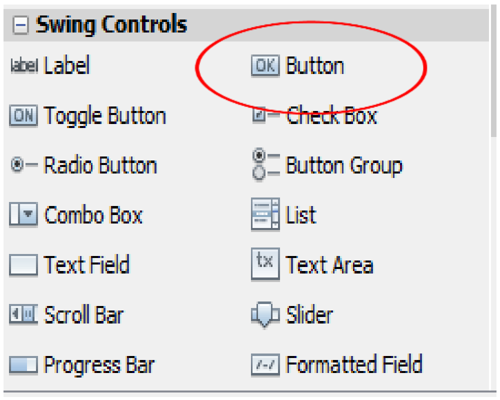
import javax.swing.*;

METHOD	DESCRIPTION	EXAMPLE
showMessageDialog ()	Shows a one-button, modal dialog box that gives the user some information.	JOptionPane.showMessageDialog(this, “Lets learn Java”);
showConfirmDialog ()	Shows a three-button modal dialog that asks the user a question. User can respond by pressing any of the suitable buttons.	Confirm= JOptionPane.showConfirmDialog(null, “quit?")
showInputDialog ()	Shows a modal dialog that prompts the user for input. It prompts the user with a text box in which the user can enter the relevant input.	name= JOptionPane.showInputDialog(this, “ENTER ROLL NUMBER:");

BUTTON CONTROL

A button is a component that the user presses or pushes to trigger a specific action. When the user clicks on the button at runtime, the code associated with the click action gets executed.

PROPERTY	DESCRIPTION
BACKGROUND	Sets background color
FOREGROUND	Sets foreground color
FONT	Sets font of text on button
TEXT	Sets the text displayed on button



METHOD S	DESCRIPTION	EXAMPLE

setEnabled ()	Enables/disables the button	JButton1.setEnabled(false); jbutton1.setEnabled(true);
setVisible ()	Visible/invisible the button	JButton1.setVisible(false); jbutton1.setVisible(true);

Click to view video on simple GUI application

METHOD: EXITING JAVA APPLICATION

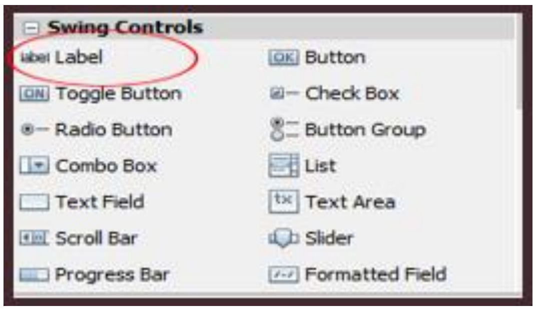
To exit a java application, we use the method:

```
System.exit(0);
```

LABEL CONTROL

Label provides text instructions or information. It displays a single line of read-only text, an image or both text and image.

PROPERTY	DESCRIPTION
BACKGROUND	Sets background color
FOREGROUND	Sets foreground color
FONT	Sets font of text on button
TEXT	Sets the text displayed on button



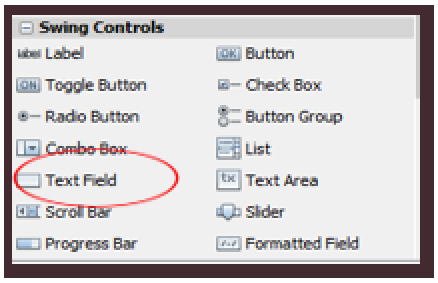
METHODS	DESCRIPTION	EXAMPLE
isEnabled()	Returns true if label is enabled else false	Boolean b= jLabel1.isEnabled();

setVisible()	Visible/invisible the label	jLabel1.setVisible(false); jLabel1.setVisible(true);
setText()	Sets the text on label at run time	JLable1.setText("Enter name");

TEXT FIELD CONTROL

Text Field allows editing/displaying of a single line of text. For eg. Name, Phone number etc.

PROPERTY	DESCRIPTION
BACKGROUND	Sets background color
FOREGROUND	Sets foreground color
FONT	Sets font of text on button
TEXT	Sets the text displayed on button



METHODS	DESCRIPTION	EXAMPLE
isEnabled()	Returns true if text field is enabled else false	boolean b= jTextField1.isEnabled();
setVisible()	Visible/invisible the button	jTextField1.setVisible(false);
setText()	Sets the text on text field at run time	jTextField1.setText("WELCOME");
getText()	Gets the text from the text field	string str = jTextField1.getText();
isEditable()	Returns true if the component is editable else false.	boolean b= jTextField1.isEditable();

setEditable()	true if text in the text field is editable else false.	JTextField1.setEditable(true);

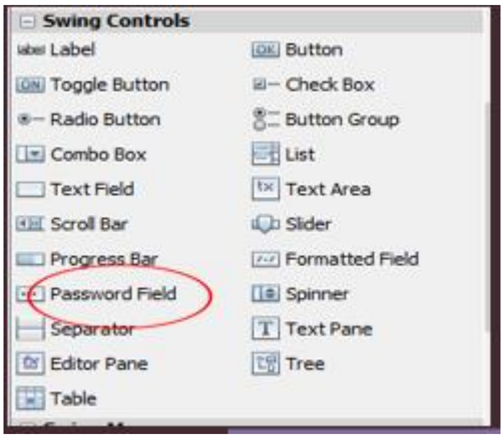
setEditable() vs setEnabled() vs setVisible()

- **setEditable()**: TextField are editable by default ie it's contents can be changed at run time. The code setEditable(false) makes the TextField uneditable. It is still selectable but the user cannot change the TextField's contents directly.
- **setEnabled()**: The code setEnabled(true) implies that TextField can trigger a reaction at run time. The code setEnabled(false), disables this TextField. It is not selectable and the user cannot change the TextField's contents directly.
- **setVisible():** setVisible(true) implies that the component is visible and setVisible(false) implies that the component is hidden.

PASSWORD CONTROL

It is used to enter confidential input like passwords which are single line. It suppresses the display of input and each character entered can be replaced by an echo character. By default, the echo character is the asterisk, *.

PROPERTY	DESCRIPTION
BACKGROUND	Sets background color
FOREGROUND	Sets foreground color
FONT	Sets font of text on button
TEXT	Sets the text displayed on button
echoChar	Sets the character

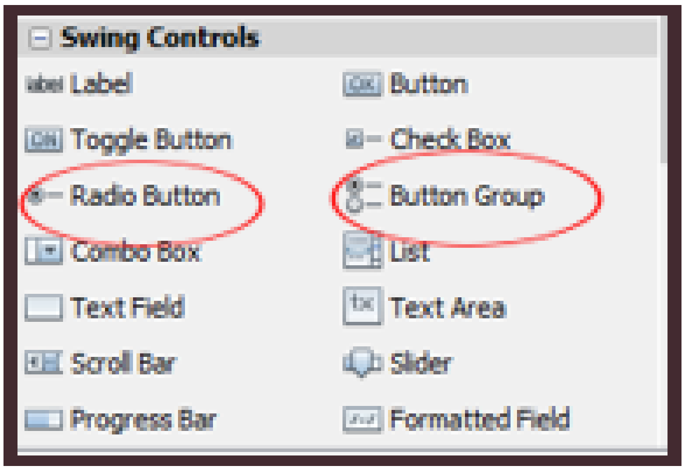


	that will be displayed instead of text.

RADIO BUTTON CONTROL

Radio button is used when user has to select one option out of many mutually exclusive options given. For eg. Gender(male or female), Stream(Science, Commerce, Humanities)

PROPERTY	DESCRIPTION
buttonGroup	Specifies the name of the group of button to which the jRadioButton belongs.
Selected	Sets the button as selected, if set to true, default is false.

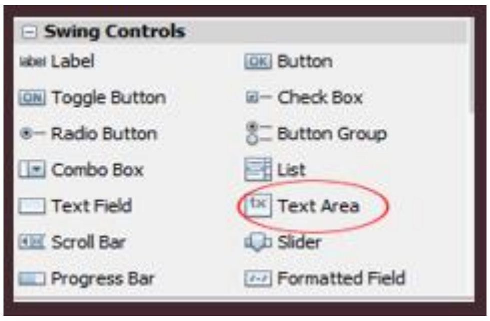


METHODS	DESCRIPTION	EXAMPLE
isSelected()	Returns true if radio button is checked else false	boolean b= jRadiobutton1.isSelected();
setSelected()	Checks(true) or unchecks the radio button.	jRadiobutton1.setSelected(false);
setText()	Sets the text on radio button at run time	jRadiobutton1.setText("WELCOME");

TEXT AREA CONTROL

Text area allows editing/displaying of a multi-line text. It automatically adds vertical or horizontal scroll bars as and when required during run time. For eg. Comments, address etc.

PROPERTY	DESCRIPTION
lineWrap	Indicates whether line of text should wrap in case it exceeds allocated width.(Default is false)
wrapStyleWord	Sends word to next line in case lineWrap is true else it results in breaking of a word, when lines are wrapped.
rows/columns	Sets number of rows/columns preferred for display.
text	Sets the text displayed on button



METHODS	DESCRIPTION	EXAMPLE
isEnabled()	Returns true if text field is enabled else false	boolean b= jTextarea1.isEnabled();
setText()	Sets the text on text field at run time	jTextarea1.setText("WELCOME");
getText()	Gets the text from the text field	string str = jTextarea1.getText();
isEditable()	Returns true if the component is editable else false.	boolean b= jTextarea1.isEditable();
append()	Adds text at the end	JTextarea1.append("we are studying JAVA");

CHECK BOX CONTROL

Check box is used when multiple options are given to the user and the user can select zero or more out of the given options. Examples of such options are Hobbies (a user may have zero or more hobbies), Magazines to subscribe for (a user may subscribe to zero or more of the given magazines) etc.

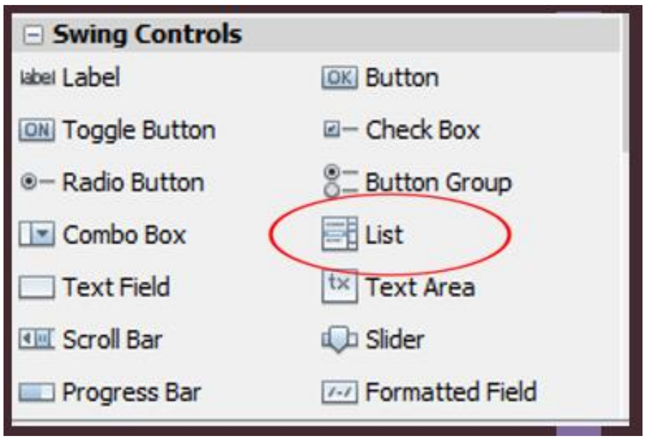
PROPERTY	DESCRIPTION
buttonGroup	Specifies the name of the group of button to which the check box belongs.
Selected	Sets the check box as selected, if set to true, default is false.

METHODS	DESCRIPTION	EXAMPLE
isSelected()	Returns true if check box is checked else false	boolean b= jCheckbox1.isSelected();
setSelected()	Checks(true) or unchecks the check box.	jCheckbox1.setSelected(false);
setText()	Sets the text on check box at run time	jCheckbox1.setText(“WELCOME”);

NOTE: The ‘\n’ character is used to move the control to the next line. It is used so that every sentence appears on the separate line in the text area.

LIST CONTROL

- A list is a scrollable set of items, used to get one or more options out of several given options which may or may not be mutually exclusive.
- Lists are preferred over checkboxes when there are large number of options.
- In such case using Check Boxes may take up a lot of space on the form and it may also be inconvenient for the user to select the desired options.

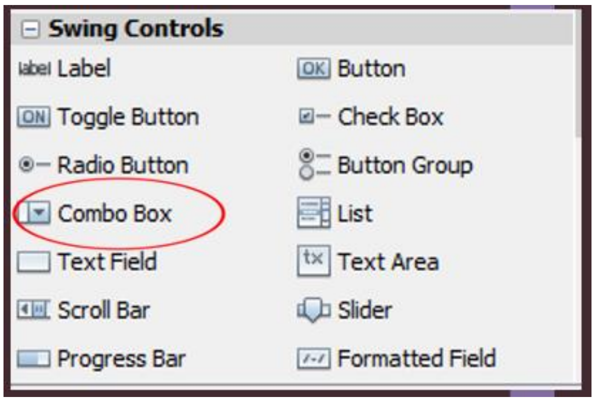


PROPERTY	DESCRIPTION
model	Contains the values to be displayed in the list.
selectedIndex	Contains the index value of selected option of the control.
SelectionMode	Describes the mode for selecting values. - SINGLE (List box allows single selection only) - SINGLE_INTERVAL (List box allows single continuous selection of options using shift key of keyboard) - MULTIPLE_INTERVAL (List box allows multiple selections of options using ctrl key of keyboard)

METHODS	DESCRIPTION	EXAMPLE
getSelectedValue()	Returns the selected value when only a single item is selected, if multiple items are selected then returns first selected value. Returns null in case no item selected.	jList1.getSelectedValue();
isSelectedIndex()	Returns true if specified index is selected.	Boolean b= jList1.isSelectedindex();

COMBO BOX CONTROL

- A combo box is a drop down box of items, used to get one option out of several given mutually exclusive options.
- Combo box are preferred over radio button when there are large number of options.
- In such case using radio buttons may take up a lot of space on the form and it may also be inconvenient for the user to select the desired options.



PROPERTY	DESCRIPTION
model	Contains the values to be displayed.
selectedIndex/selectedItem	Contains the index value/selected item of selected option of the control.

METHODS	DESCRIPTION	EXAMPLE
getSelectedItem()	Returns the selected value.	jCombobox1.getSelectedItem();
getSelectedIndex()	Returns the selected index	Boolean b= jCombobox1.getSelectedindex();
setModel()	Sets the data model that the combo box uses to get its list of elements.	jCombobox1.setModel(ComboBoxModel aModel);