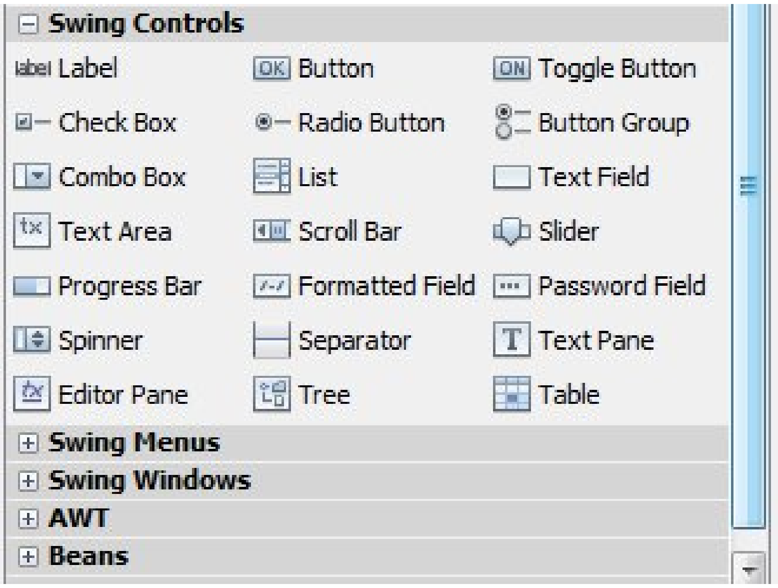
CHAPTER – 4 JAVA GUI PROGRAMMING

TYPES OF SWING COMPONENTS

1. Component
2. Container



COMPONENT

- Self contained graphic entity that can be customized and inserted into applications.
- e.g. JLabel
  JTextField
  JButton etc.

CONTAINER

- components that can hold other components.
- e.g. JFrame
  JDialog etc.

CONTAINERS

TOP LEVEL CONTAINER
CONTAINER

- can be displayed directly on desktop.

NON TOP LEVEL

- can be displayed within the context of another top level container.

HANDLING EVENTS

EVENT

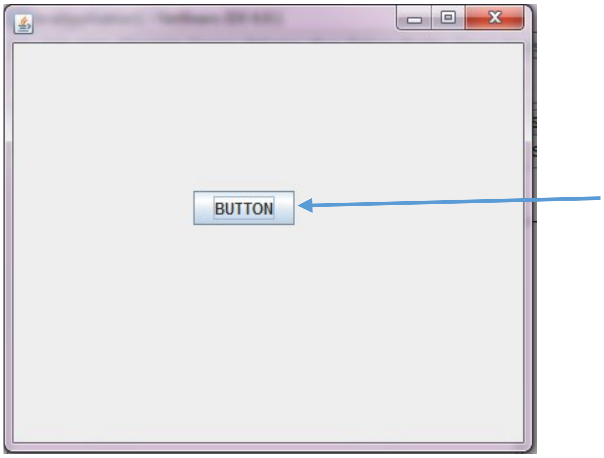
- It is occurrence of some activity either initiated by the user or internally by system.
- e.g. Clicking mouse button, pressing a key etc.

EVENT SOURCE

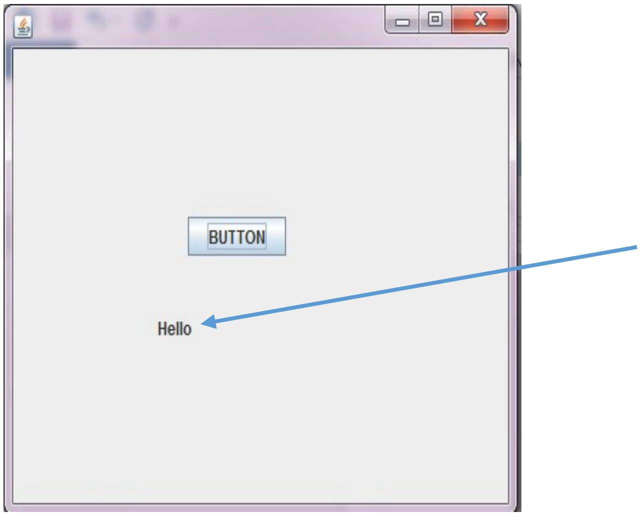
- It is the GUI component that generates the event, e.g. a button (when it is clicked).

EVENT HANDLER OR EVENT LISTENER

- It is implemented in form of code. It receives and handle events, e.g. when a button gets clicked, some value is computed and displayed.



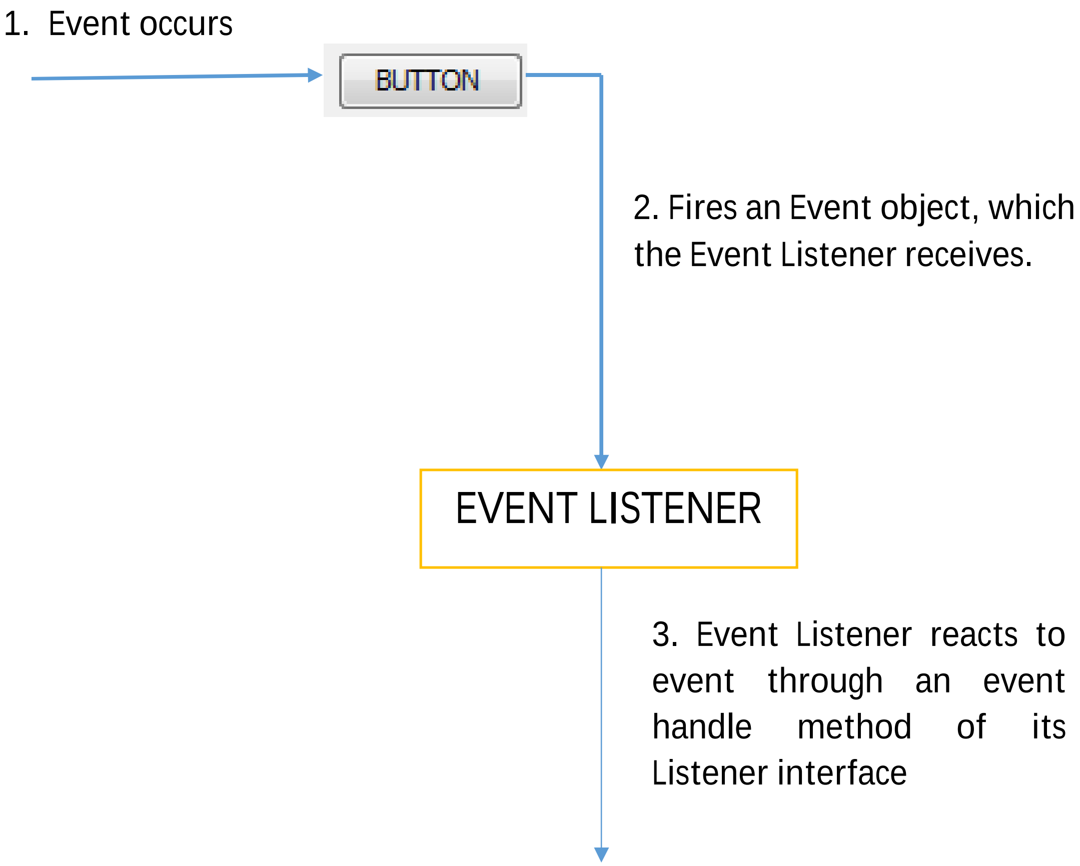
EVENT SOURCE
(BUTTON)



RESPONSE
GENERATED
AFTER CLICKING

EVENT OBJECT

- It gets created when event occurs.
- The component(*Event Source*) where the event has occurred, creates the Event Object and passes it to the *Event Listener*.
- It contains the following information:
  - Type of event occurred (whether user has pressed Enter key or mouse key)
  - Source of the event (name of the component where the event has occurred)



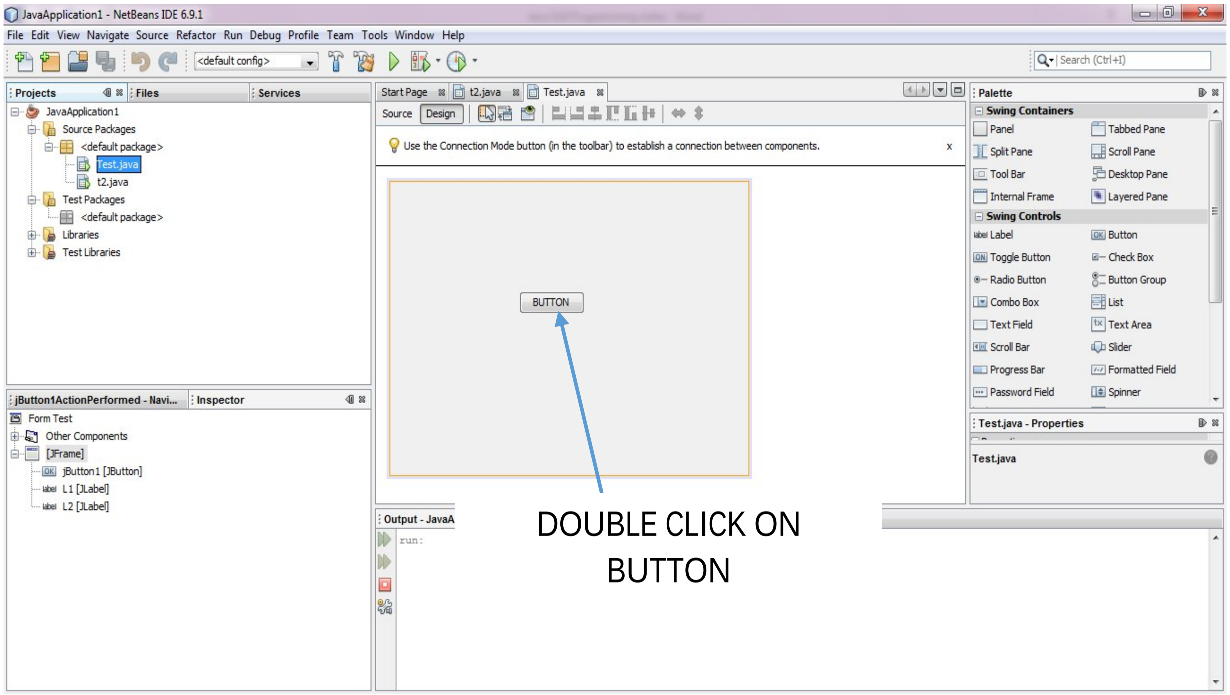
1. Event occurs

2. Fires an Event object, which the Event Listener receives.

3. Event Listener reacts to event through an event handle method of its Listener interface

1. BUTTONS

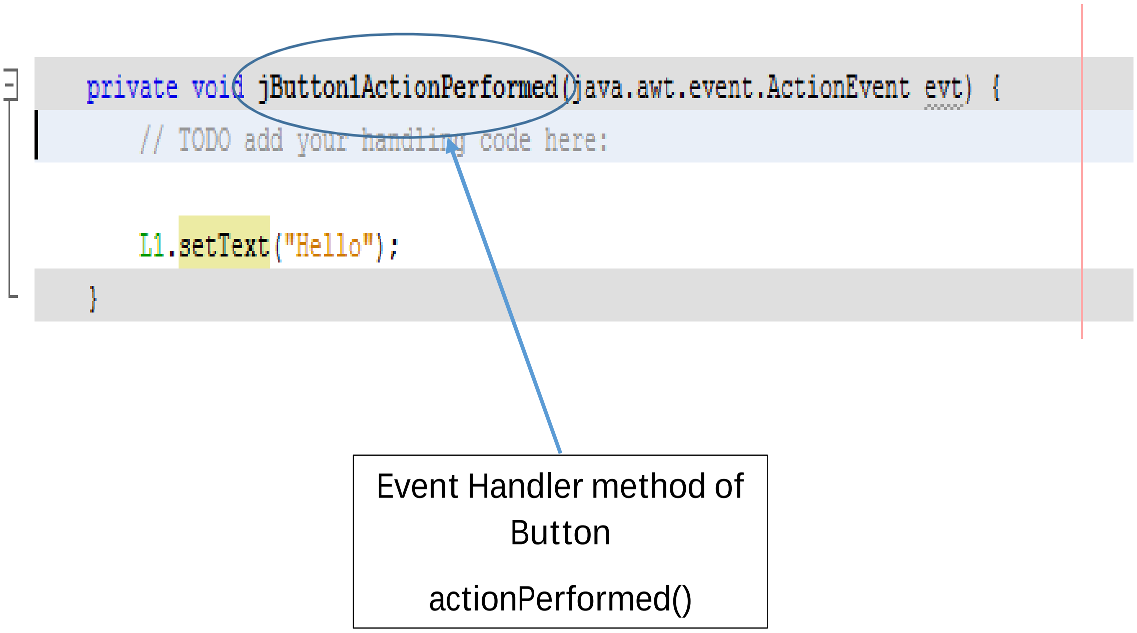
Button	Basic Facts
Swing API Class	JButton
Purpose	Displays a clickable push button
Most Common Event	Action Event
Event Listener	Action Listener
Event Handler Method	actionperformed()



Use the Connection Mode button (in the toolbar) to establish a connection between components.

```
run:
```

DOUBLE CLICK ON BUTTON



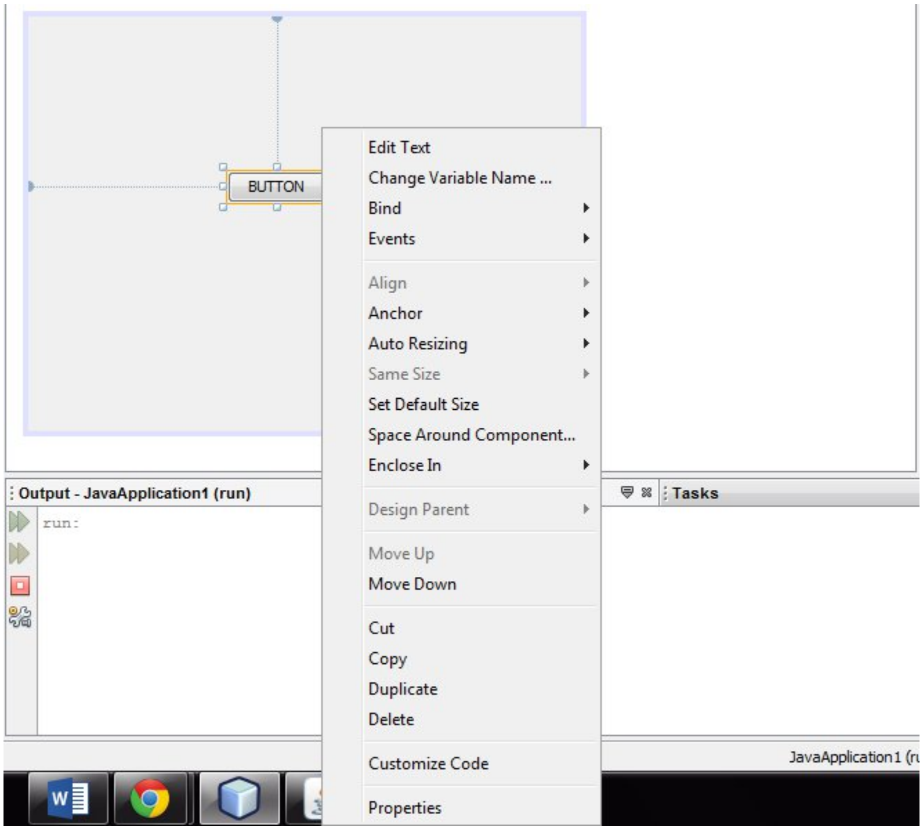
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    L1.setText("Hello");
}
```

Event Handler method of Button

actionPerformed()

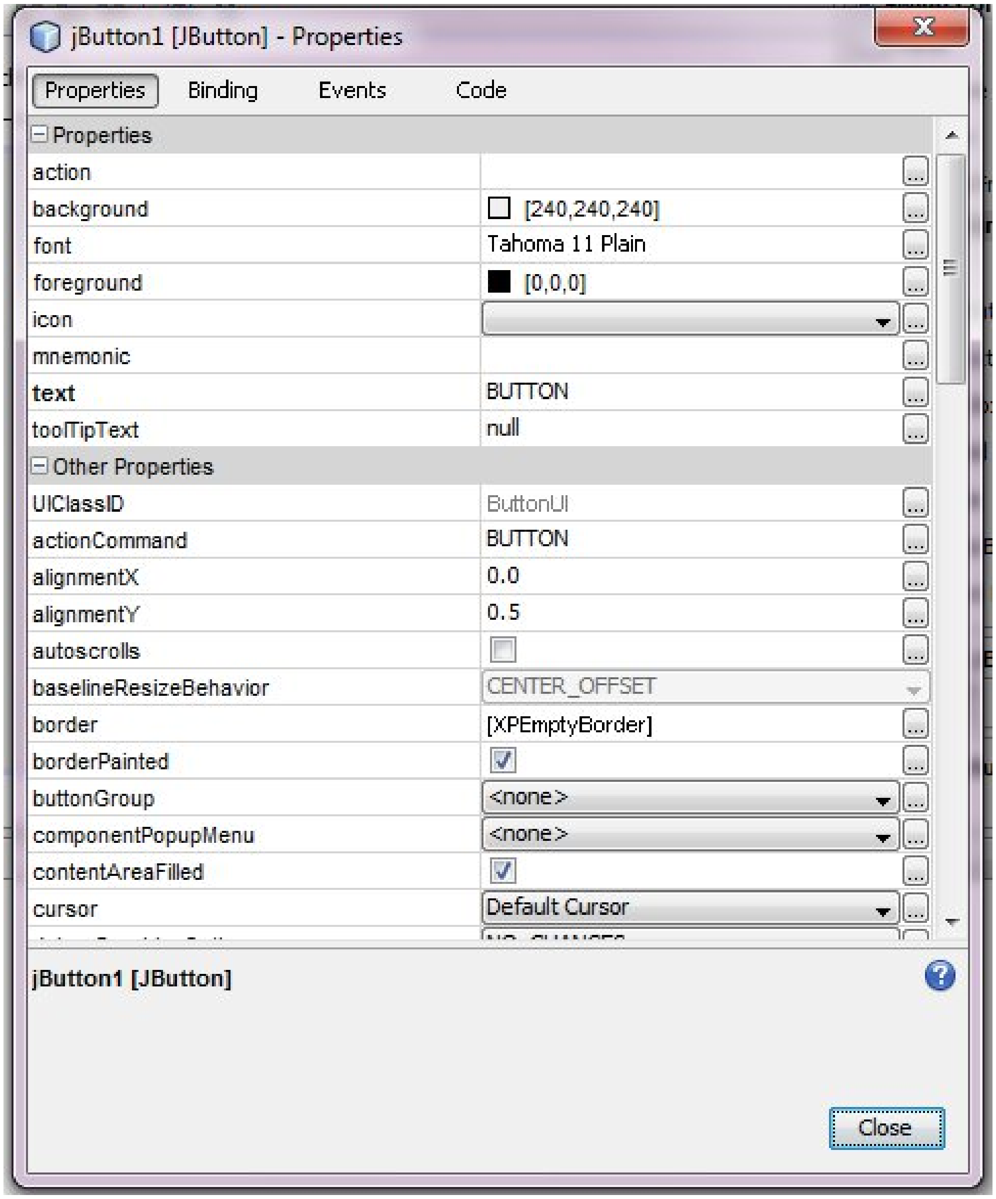
<u>PROPERTIES OF BUTTON</u>

PROPERTY	DESCRIPTION
background	Sets the background color of the button
font	Sets the font for the text of the button
foreground	Sets the foreground color, i.e. color of the text
text	Sets the text of the button
enabled	This property determines whether the button is active or not.



BUTTON
Edit Text
Change Variable Name ...
Bind
Events
Align
Anchor
Auto Resizing
Same Size
Set Default Size
Space Around Component...
Enclose In
Design Parent
Move Up
Move Down
Cut
Copy
Duplicate
Delete
Customize Code
Properties
Output - JavaApplication1 (run)
Tasks
JavaApplication1 (r

```
run:
```



jButton1 [JButton] - Properties

⊟ Properties

action	
background	□ [240,240,240]
font	Tahoma 11 Plain
foreground	■ [0,0,0]
icon	
mnemonic	
text	BUTTON
toolTipText	null

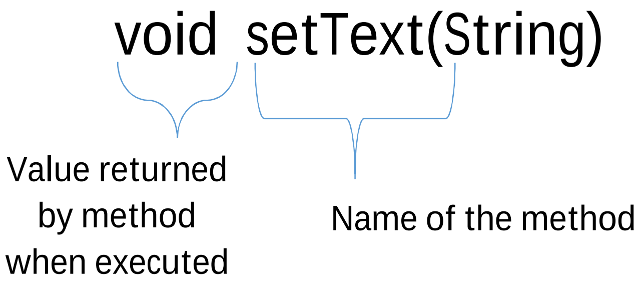
⊟ Other Properties

UIClassID	ButtonUI
actionCommand	BUTTON
alignmentX	0.0
alignmentY	0.5
autoscrolls	□
baselineResizeBehavior	CENTER_OFFSET
border	[XPEmptyBorder]
borderPainted	☑
buttonGroup	<none>
componentPopupMenu	<none>
contentAreaFilled	☑
cursor	Default Cursor

jButton1 [JButton]

METHODS OF BUTTON

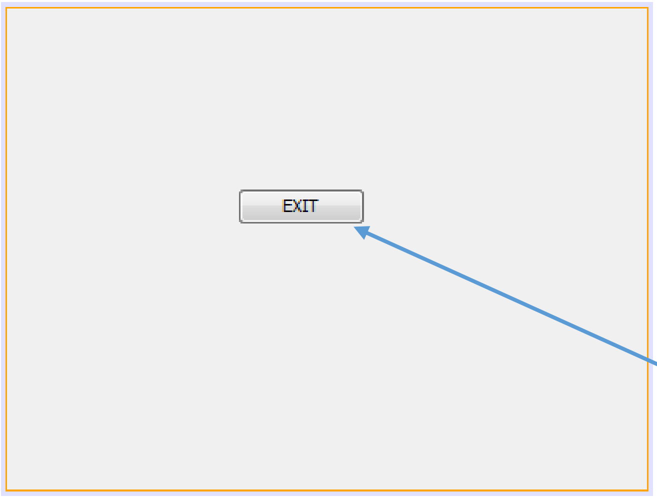
Method	Description
void setText(String)	Sets the text displayed by the button. <button-name>.setText(“Button1”)
String getText()	Returns the text displayed by the button String result=<button-name>.getText();


void setText(String)

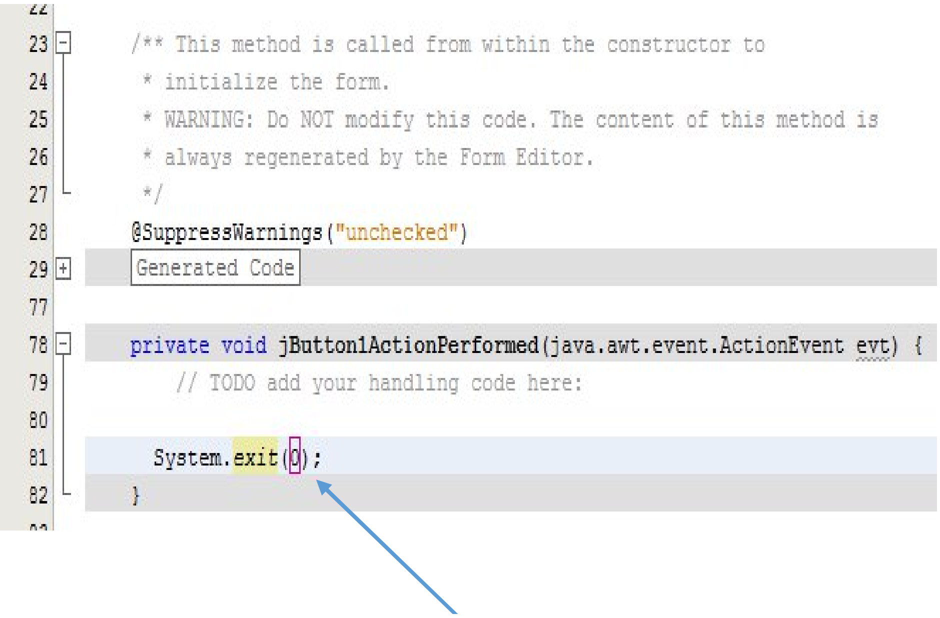
Value returned by method when executed

Name of the method

CREATING AN EXIT BUTTON



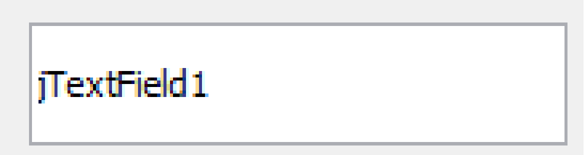
- Drag a Button from the Swing Controls box
- Rename it as “Exit”
- Double click on Button



```
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
Generated Code

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

  System.exit(0);
}
```

Just type the line
And
Run your application.

TEXTFIELDS



jTextField1

- Display a field that allow the user to enter a single line of text.

PROPERTIES OF JTEXTFIELD

PROPERTY	DESCRIPTION
background	Sets the background color of the text field
font	Sets the font for the text of the text field
foreground	Sets the foreground color, i.e. color of the text
text	Sets the text of the text field
editable	If set to true, allow the user to edit the contents of the text field.

METHODS OF JTEXTFIELD

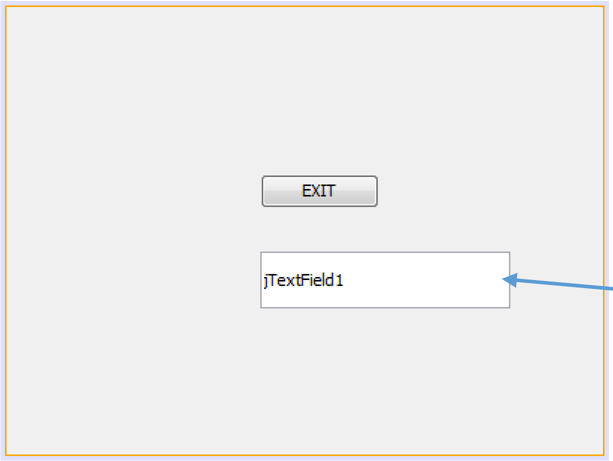
Method	Description
void setText()	Sets the text displayed by the textfield. <textfield-name>.setText("Textfield")
String getText()	Returns the text displayed by the textfield String result=< textfield –name>.getText();

void setEditable(boolean)	Sets whether the user can edit the text in the text field.
	e.g., NameTF.setEditable(true);
	or
	NameTf.setEditable(false);

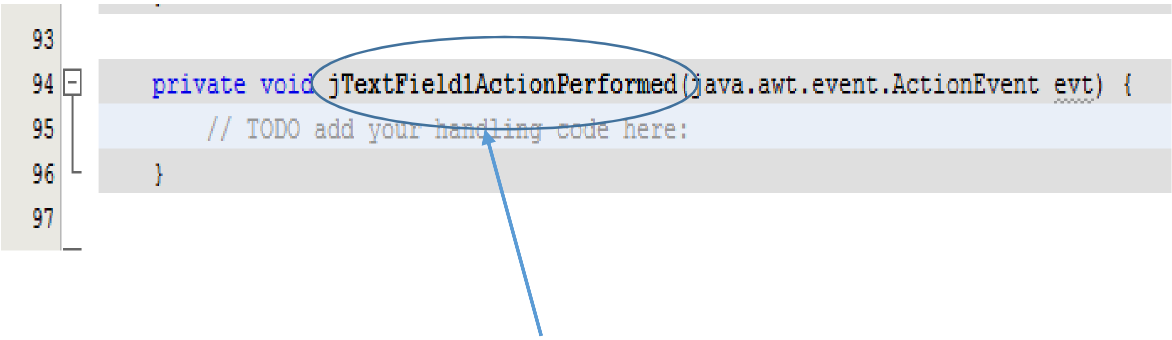
boolean isEditable()	Returns true/false whether the user can edit the text in the text field or not.

JTEXTFIELD EVENTS

TextField	Basic Facts
Swing API Class	JTextField
Purpose	Displays editable text
Most Common Event	Action Event
Event Listener	Action Listener
Event Handler Method	actionperformed()



Double click on the textfield.



```
    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

Event Listener of TextField.

PASSWORD FIELDS

- Used for accept passwords.
- The character that is displayed in place of text being entered is known as echo character.

PROPERTIES OF JPASSWORDFIELD

PROPERTY	DESCRIPTION
background	Sets the background color of the password field
font	Sets the font for the text of the password field
foreground	Sets the foreground color, i.e. color of the text
text	Sets the text of the password field
echoChar	This property determines which character will replace the text in display.

METHODS OF JPASSWORDFIELD

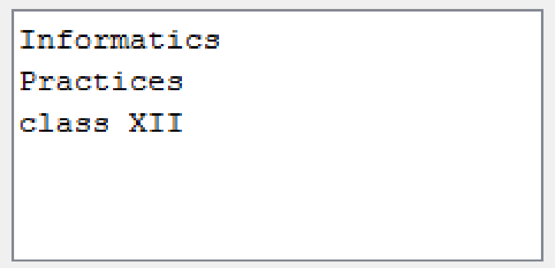
Method	Description
void setEchoChar(char)	Sets the echo character
Char getEchoChar()	Returns the echo character
Char[] getPassword()	Returns the text displayed by the password field.

JPASSWORD EVENTS

Password Field	Basic Facts
Swing API Class	JPasswordField
Purpose	Displays encrypted text
Most Common Event	Action Event
Event Listener	Action Listener
Event Handler Method	actionperformed()

TEXT AREA

- It is a multi-line text component to display text or allow the user to enter text.



```
Informatics
Practices
class XII
```

PROPERTIES OF JTEXTAREA

PROPERTY	DESCRIPTION
background	Sets the background color of the text area
font	Sets the font for the text of the text area
foreground	Sets the foreground color, i.e. color of the text
text	Sets the text of the text area
enabled	Sets whether the text area is enabled.
editable	Sets whether the text area is editable or not.

METHODS OF JTEXTAREA

Method	Description
void setText()	Sets the text displayed by the textarea. <textarea-name>.setText(“TextArea”)
String getText()	Returns the text displayed by the textarea String result=< textarea –name>.getText();

void setEditable(boolean)	Sets whether the user can edit the text in the text area.
	e.g., NameTA.setEditable(true);
	or
	NameTA.setEditable(false);

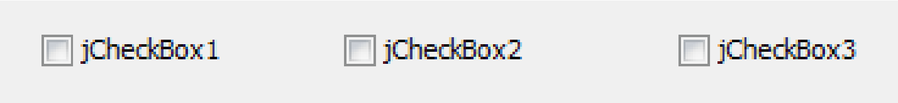
boolean isEditable()	Returns true/false whether the user can edit the text in the text field or not.
void append(String)	Adds the specified text to the end of the text area.

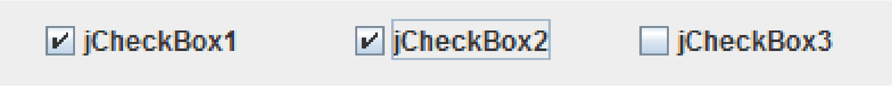
JTEXTAREA EVENTS

Text Area	Basic Facts
Swing API Class	JTextArea
Purpose	Displays multi-line text
Most Common Event	Action Event
Event Listener	Action Listener
Event Handler Method	actionperformed()

CHECK BOXES

- It indicates whether a particular condition is ON or OFF.
- Check boxes works independently of each other, a user can select any number of checkboxes at the same time.
- It is a control with a rectangular area that can be checked or unchecked. Checked rectangle means check box is selected.





PROPERTIES OF JCHECKBOX

PROPERTY	DESCRIPTION
background	Sets the background color of the check box.
font	Sets the font for the text of the check box.
foreground	Sets the foreground color, i.e. color of the text
text	Sets the text of the check box.
selected	If set to true, the check box appears selected.

METHODS OF JCHECKBOX

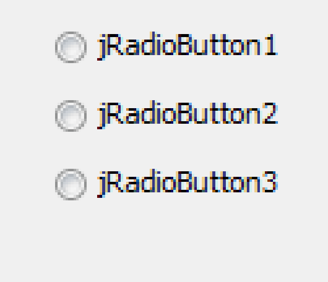
Method	Description
String getText()	Returns the text displayed by the checkbox. String result=< checkbox –name>.getText();
void setText()	Sets the text displayed by the checkbox. <checkbox-name>.setText(“Checkbox”)
void setEnabled(boolean b)	Enables the check box if true is passed otherwise disable the check box.
boolean isEnabled()	Returns the state whether the check box is enabled. An enabled check box can respond to user input and generate events. Check box are enabled initially by default.
void setVisible(boolean b)	Makes the checkbox visible if true is passed otherwise hides the checkbox.
boolean isVisible()	Returns true if the check box is visible, false otherwise.
boolean isSelected()	Returns the state of the check box. Returns true if the check box is selected, false if it’s not.

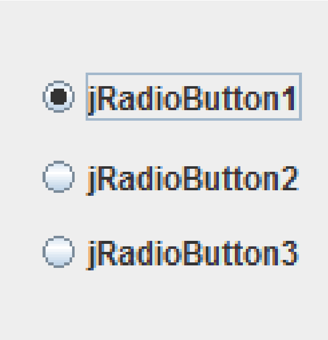
JCHECKBOX EVENTS

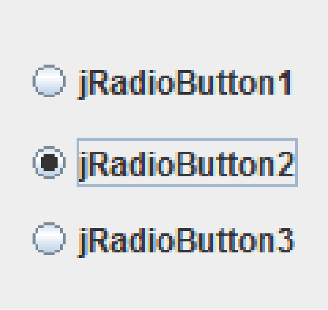
Text Area	Basic Facts
Swing API Class	JCheckBox
Purpose	Displays on/off or yes/no type choice option.
Most Common Event	Item Event
Event Listener	ItemListener
Event Handler Method	itemStateChanged()

RADIO BUTTONS

- Present a set of two or more choice to the user.
- Unlike checkboxes, radio buttons should always work as part of a group; selecting one radio button immediately clears all the other buttons in the group.

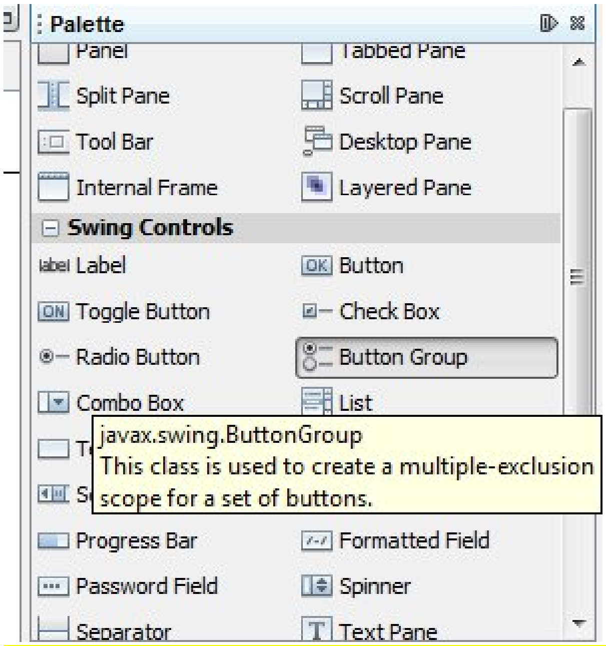






STEPS TO ADD RADIO BUTTONS

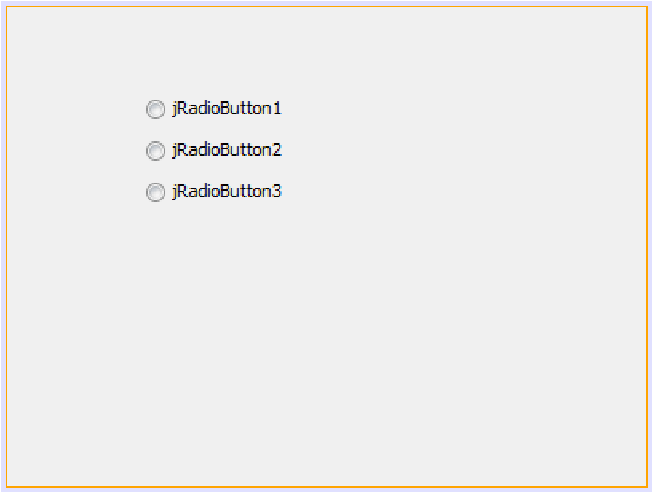
1. Add ButtonGroup to your application by first clicking at Button Group control on palette and then dragging into design space.



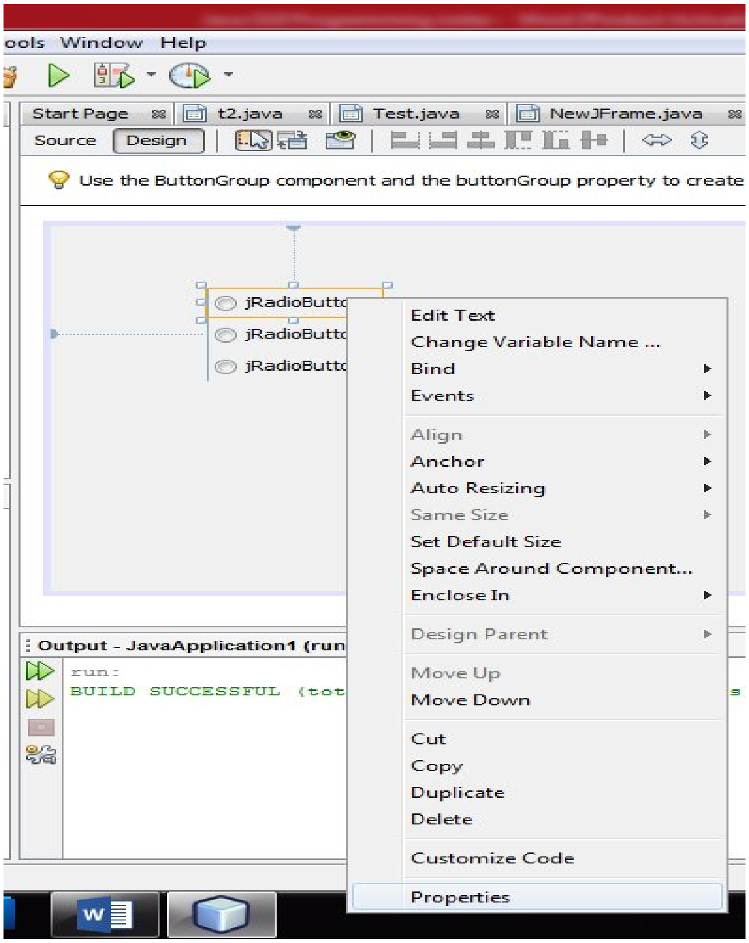
javax.swing.ButtonGroup
This class is used to create a multiple-exclusion scope for a set of buttons.

Button Group is an invisible control, i.e. it will not be visible when you drag and drop it on design space.

2. Then drag desired number of radiobuttons on design space.

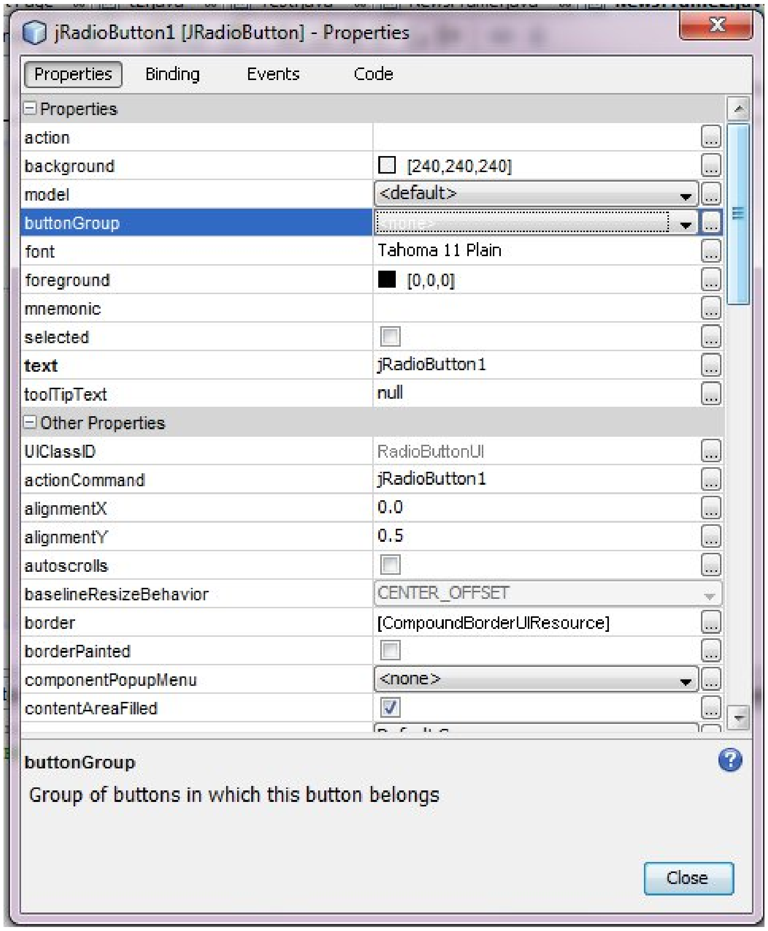


3. Right click on each radio button and set the buttongroup property.



Use the ButtonGroup component and the buttonGroup property to create

```
run:
BUILD SUCCESSFUL (tot
```

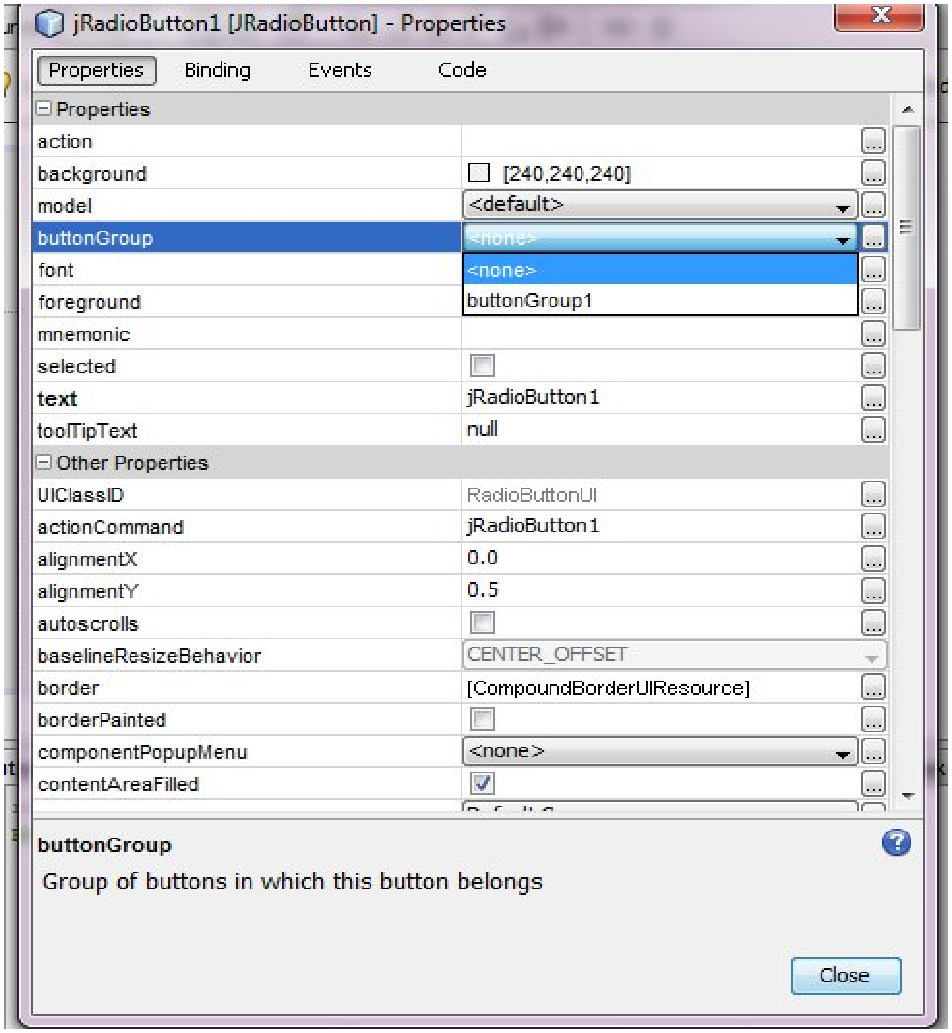


Properties	
action	
background	□ [240,240,240]
model	<default>
buttonGroup	<none>
font	Tahoma 11 Plain
foreground	■ [0,0,0]
mnemonic	
selected	☐
text	jRadioButton1
toolTipText	null
Other Properties	
UIClassID	RadioButtonUI
actionCommand	jRadioButton1
alignmentX	0.0
alignmentY	0.5
autoscrolls	☐
baselineResizeBehavior	CENTER_OFFSET
border	[CompoundBorderUIResource]
borderPainted	☐
componentPopupMenu	<none>
contentAreaFilled	☑

buttonGroup

Group of buttons in which this button belongs

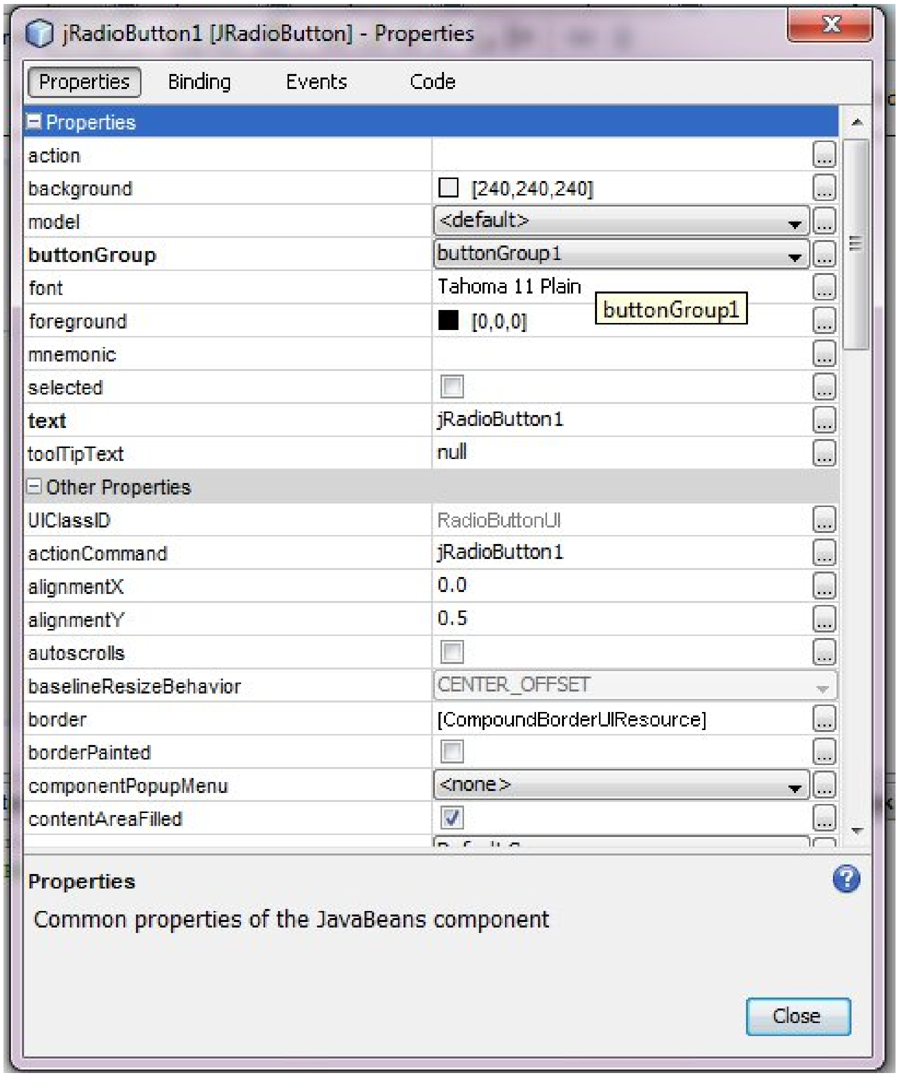
4. Select the appropriate value.



Properties	
action	
background	□ [240,240,240]
model	<default>
buttonGroup	<none>
font	
foreground	
mnemonic	
selected	□
text	jRadioButton1
toolTipText	null
Other Properties	
UIClassID	RadioButtonUI
actionCommand	jRadioButton1
alignmentX	0.0
alignmentY	0.5
autoscrolls	□
baselineResizeBehavior	CENTER_OFFSET
border	[CompoundBorderUIResource]
borderPainted	□
componentPopupMenu	<none>
contentAreaFilled	☑

buttonGroup

Group of buttons in which this button belongs



Properties	
action	
background	□ [240,240,240]
model	<default>
buttonGroup	buttonGroup1
font	Tahoma 11 Plain
foreground	■ [0,0,0]
mnemonic	
selected	□
text	jRadioButton1
toolTipText	null
Other Properties	
UIClassID	RadioButtonUI
actionCommand	jRadioButton1
alignmentX	0.0
alignmentY	0.5
autoscrolls	□
baselineResizeBehavior	CENTER_OFFSET
border	[CompoundBorderUIResource]
borderPainted	□
componentPopupMenu	<none>
contentAreaFilled	☑

Properties

Common properties of the JavaBeans component

5. Set the buttongroup property of other radio buttons.

PROPERTIES OF JRADIOBUTTON

PROPERTY	DESCRIPTION
background	Sets the background color of the radio button
font	Sets the font for the text of the radio button
foreground	Sets the foreground color, i.e. color of the text
text	Sets the text of the radio button.
selected	If set to true, the radio button appears selected.

METHODS OF JRADIOBUTTON

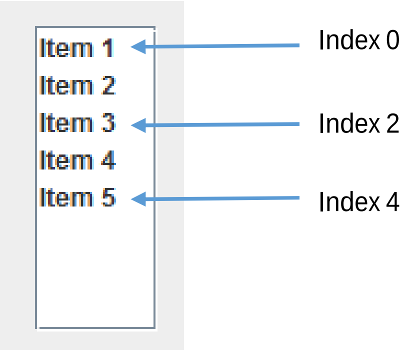
Method	Description
String getText()	Returns the text displayed by the radio button. String result=< radio button –name>.getText();
void setText()	Sets the text displayed by the radio button. < radio button -name>.setText(“RadioButton”);
void setEnabled(boolean b)	Enables the radio button if true is passed otherwise disable the check box.
boolean isEnabled()	Returns the state whether the radio button is enabled. An enabled radio button can respond to user input and generate events. Check box are enabled initially by default.
void setVisible(boolean b)	Makes the radio button visible if true is passed otherwise hides the radio button.
boolean isVisible()	Returns true if the radio button is visible, false otherwise.
boolean isSelected()	Returns the state of the Radio button. Returns true if the Radio button is selected, false if it’s not.

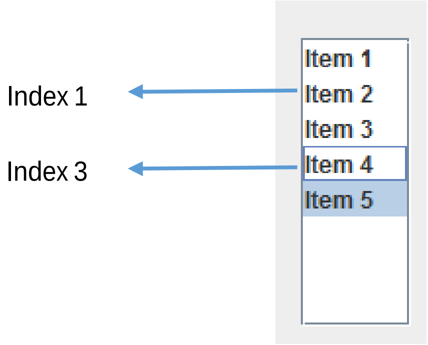
JRADIOBUTTON EVENTS

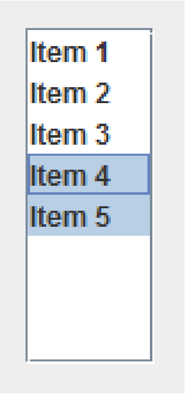
Radio Button	Basic Facts
Swing API Class	JRadioButton
Purpose	Displays a set of mutually exclusive options.
Most Common Event	Item Event
Event Listener	ItemListener
Event Handler Method	itemStateChanged()

LISTS

- It is a box-shaped control containing a list of attributes from which single or multiple selections can be made.







PROPERTIES OF JLIST

PROPERTY	DESCRIPTION
model	Used to add the elements/items in the list.
selectionMode	Describes the mode for selecting list items. It defines three modes: 1. SINGLE (single item can be selected) 2. SINGLE_INTERVAL (single range of items can be selected) 3. MULTIPLE_INTERVAL(multiple ranges of items can be selected)
selectedIndex	Returns the index of the specified item. Default is -1 i.e. initially no item is selected.
selectedIndices	Returns the indices of selected item in form of an array. Default is -1 i.e. initially no item is selected.

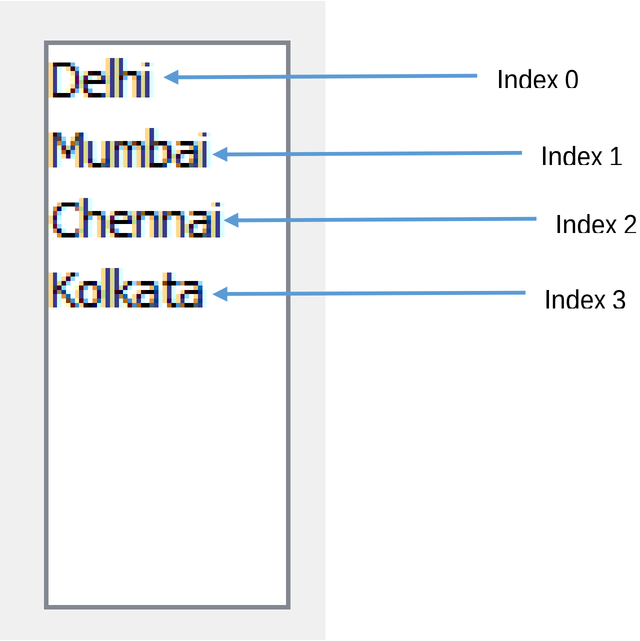
METHODS OF JLIST

Method	Description
int getSelectedIndex()	Returns the index of selected item,when only a single item is selected in the list.
int[ ] getSelectedIndices()	Returns the array of all of the selected indices, in increasing order.
object getSelectedValue()	Returns the selected value when only a single item is selected in the list.
object[ ] getSelectedValues()	Returns an array of all the selected value, in increasing order.

JLIST EVENTS

List	Basic Facts
Swing API Class	JList
Purpose	Displays a list of data
Most Common Event	ListSelection
Event Listener	ListSelectionListener
Event Handler Method	valueChanged()

HANDLING SINGLE SELECTION



CityList

Two methods used in single selection:

1. int getSelectedIndex() – which returns the index of selected index.
2. object getSelectedValue() – which returns the value of selected item.

To obtain the index of selected item:

```
int ind = CityList.getSelectedIndex();
```

To obtain the selected item:

```
String selcity = (String)CityList.getSelectedValue();
```

Since the return type of above method is object, we need to explicit convert the return type to string.

HANDLING MULTIPLE SELECTIONS

Two methods used in single selection:

1. int [ ] getSelectedIndices() – which returns the indices of selected items in form of int array.
2. Object [ ] getSelectedValues() – which returns the values of selected items in form of object.

To obtain the indices of selected items:

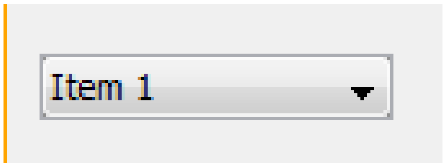
```
int ind [ ] = CityList.getSelectedIndices();
```

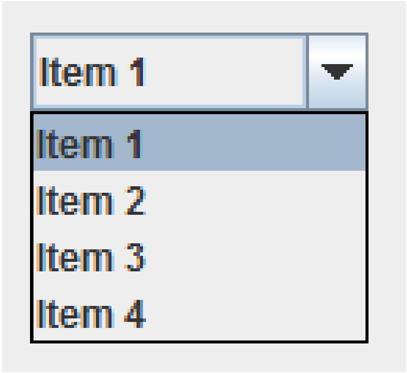
To obtain the selected item:

```
Object [ ] cities = CityList.getSelectedValues();
String acity = (String) cities [ I ];
```

COMBOBOXES

- It appears as a text field along with a drop-down list arrow from which the user can choose a value.





- It allow the user to select only one item at a time.

PROPERTIES OF COMBOBOX

PROPERTY	DESCRIPTION
model	Used to add the elements/items in the combo box.
selectedIndex	Returns the index of the specified item. Default is -1 i.e. initially no item is selected.
selectedItem	Returns the selected item.

METHODS OF COMBOBOX

1. int getSelectedIndex() – Returns the index of the selected item.
2. object getSelectedItem() – Returns the selected item.

COMBOBOX EVENTS

ComboBox	Basic Facts
Swing API Class	JComboBox
Purpose	Displays a list that allow single selection
Most Common Event	Action Event, Item Event
Event Listener	ActionListener, ItemListener

Event Handler Method	actionperformed()
	stateChanged()

CHAPTER 5 & 7
INTRODUCING CLASSES, OBJECTS AND INHERITANCE

OBJECT

- An object, in real world, is anything that is visible or tangible.

- Each object has a unique identity, some characteristic and behavior.

- For example: an orange. Its characteristics are: it is spherical shaped, its color is orange etc. Its behavior is: it is citrus in nature and it tastes sweet and sour.

- Another example: a chair
  Its characteristic is: It has four leg, a back, may or may not have arms.
  Its behavior is: it let you sit on it.

An object is an identifiable entity with some characteristic and behaviour.

Q: What can be the characteristic and behaviour of a car?

Characteristic:

Behaviour:

SOFTWARE OBJECT

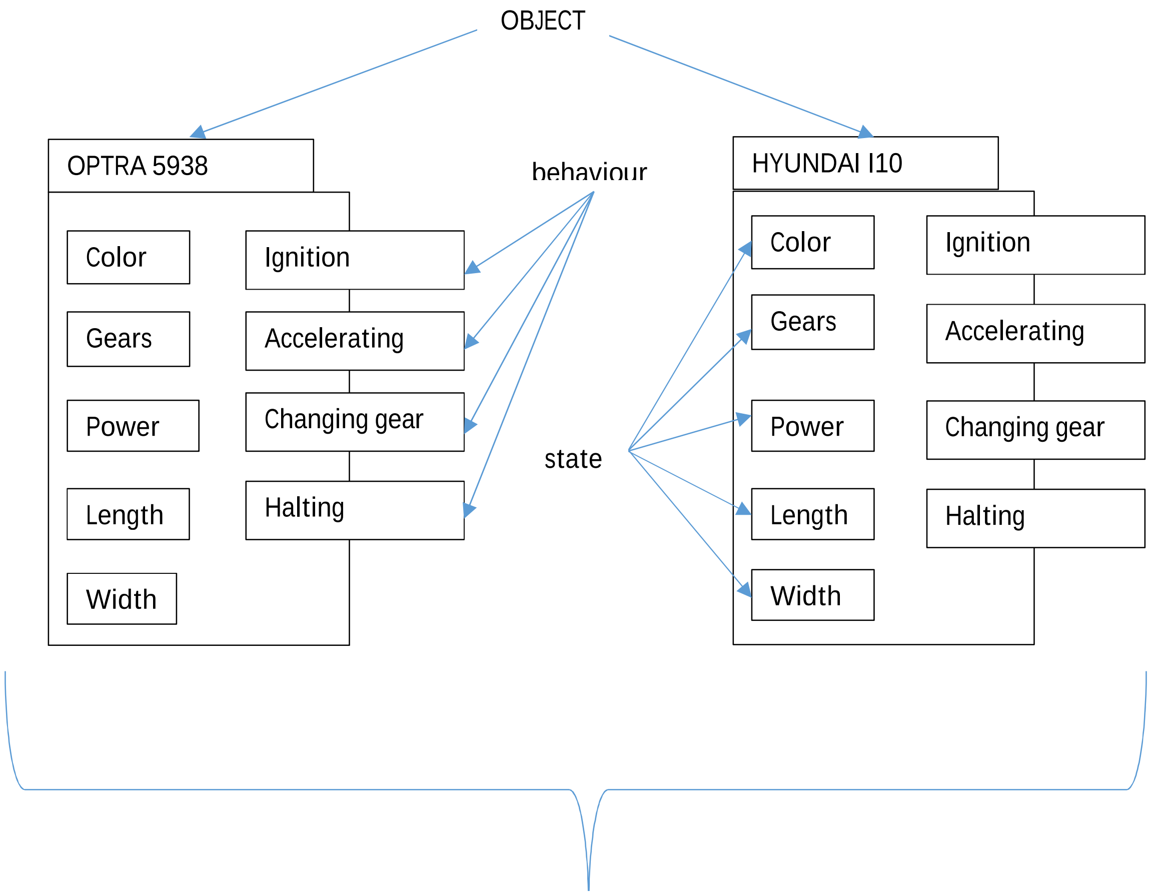
- In terms of software the state (characteristic) of an object is maintained through variables or data items.

- And the behaviour is maintained through functions generally called methods.

CLASS

A class represents a set of objects that have a common structure and common behavior.

For example there is a class Car.



CLASS – CAR

CLASSES

- A class is a blueprint for creating different objects, which defines its properties and behavior.
- A class is defined through keyword class.

- For instance – following lines define a class namely Fruit with two fields namely grams and cals_per_gram and a method namely total_calories ().

```
class Fruit {                          INSTANCE VARIABLE OF CLASS FRUIT
      int grams;
      int cals_per_grams;
                                          METHOD/FUNCTION OF CLASS FRUIT

      int total_calories( ) {
            return(grams * cals_per_gram);
      }
}
```

Object creation from Fruit class:

```
Class-name object = new Class-name ();
      E.g.  Fruit apple = new Fruit ( );
```

Accessing the variables of class:

```
<object-name>.variable = value;
      E.g.    apple.grams = 20;
              apple.cals_per_grams= 200;
```

Accessing/invoking method of class:

```
<object-name>.method ( );
E.g.  apple . total_calories ( );
```

PROBLEMS

Q1: Consider the following Java class:

```
    Public class Date {

      int day;
      int month;
      int year;

     void display ( ){

                System.out.println("Date");

               }

      }
```

(i) How many instance variables does it have?
(ii) How many methods does it have?
(iii) Create an object of class Date.
(iv) Write statements to store any value in all the instance variables.
(v) Write a statement to call/ invoke the display () method.

Q2: Create a class Car and define its instance variable and some methods.

CLASS AS USER DEFINED DATA TYPE

- The data types are based on fundamental or primitive data types, are known as Composite Datatypes. Since these data types are created by users, these are known as User-Defined Datatypes.

- A class is a good example of composite datatype.

```
class TypeDemo {

   byte a;
   int b;                PRIMITIVE DATA TYPES
   float c;
   char d;

   void getdata() {
          .........
         }
}
```

Once a class is declared, variable of this class type can be declared and created e.g.,

```
TypeDemo obj1 = new TypeDemo ( );

   CLASS                          OBJECT
```

- Variable of a class type are known as objects.
- obj1 in the above line is the object of class TypeDemo.

INHERTIANCE

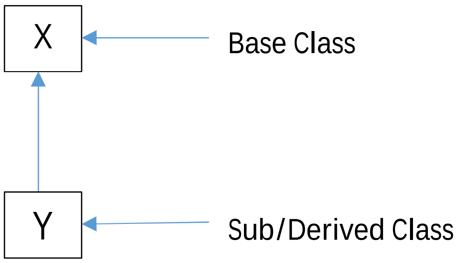
- The capability of one class to derive properties from another class.
- The class inheritance lets you derive new classes (called derived class) from old classes (base class), with the derived class inheriting the properties, including the methods of the old class.

ADVANTAGE OF INHERITANCE

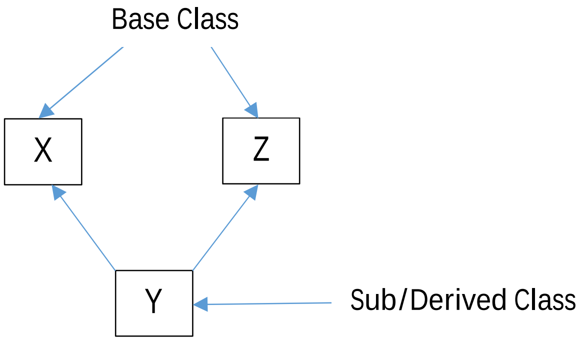
- Code reusability.
- Extend the capability of existing code.

TYPES OF INHERITANCE

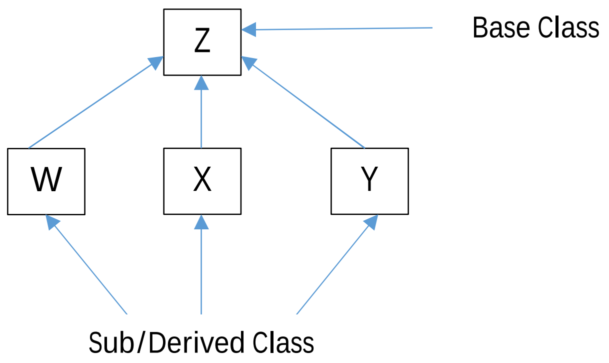
- Single Inheritance
  When a subclass inherits only from one base class.



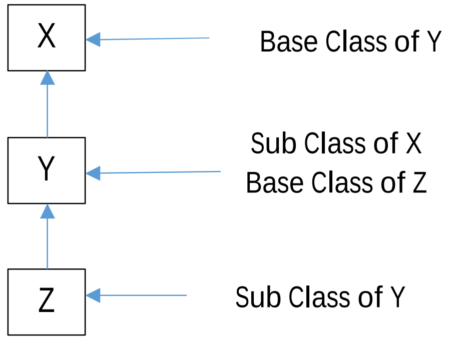
- Multiple Inheritance
  When a subclass inherits from multiple base classes, it is known as multiple inheritance.



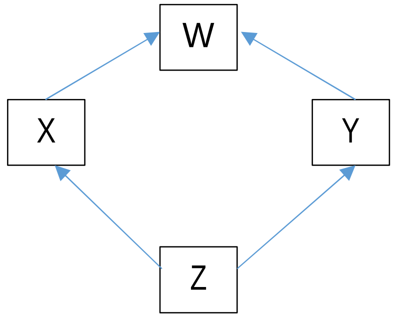
- Hierarchical Inheritance
  When many subclasses inherit from a single base class.



- Multilevel inheritance
  When a subclass inherits from class that itself inherits from another class.



- Hybrid inheritance
  When a subclass inherits from multiple base classes and all of its base classes inherits from a single base class.



DERIVED/SUB AND BASE/SUPER CLASSES

- A derived class (or a subclass) has to identify the class from which it is derived i.e., its base class (or superclass).
- The Syntax of defining a derived/ sub class is:

```
class <sub class-name> extends <super-class-name> {
            // Members of sub class
}
```

e.g.

```
class One {

int a;
char b;

void printerone( ) {

      System.out.println (“Informatics Practices“ );

   }

}
```

Derived/Sub Class

Base/Super Class

```
class Two extends One {

   int x, y;

   void printertwo( ) {

      System.out.println (“Again Informatics Practices“ );

           }

    }
```

- Class Two is Sub/Derived class of Class One which is the base or super class.
- While extending classes, instance variables and methods of super class also become part of new class.

It means,

int a;
char b;

INSTANCE VARIABLES OF CLASS ONE

and

void printerone()

METHOD OF CLASS ONE

now are the parts of class two, as this class is extended from class one. Also the class Two have its own instance variable (int x, y;) and method void printertwo( ).

FUNCTION OVERLOADING

- Java allows functions/methods to have the same name if it can distinguish them by their number and type of argument.
- For example, the following two functions are different for a Java class:

  float divide (int a , int b) { ....}
  int divide (float a , float b) { ....}

  That is, divide( ) taking two int arguments is different from divide( ) taking two float arguments.

  This is known as function overloading.

ABSTRACT CLASSES

- An Abstract Class is the one that simply represents a concept and whose objects can't be created. It is created through the use of keyword abstract.
- Syntax of using abstract keyword :

```
abstract class <class-name> {
              ..................
                }
```

e.g.

Consider the following code fragment:

```
abstract class Shape {

                String name;
                double area;

                void display ( ) {.................}

                    }
class Circle extends Shape {

             ..........

              }


class Rectangle extends Shape {

             ..........

             }
```

In above example, we definitely need to create objects of classes Circle and Rectangle but no object of Shape class should get declared as it represents merely a concept.

PROBLEMS

1. When creating a subclass, what keyword is used to include a superclass?
2. Does a subclass include the members of its superclass?
3. A class Method inherits from a class concept. Write syntax to define class Method.
4. State True or False.
   (i) A subclass inherits both member variables and member methods of superclass.
   (ii) A class created with keyword abstract can have at the most one object.
5. Define Inheritance. Discuss Various types of inheritance.