Unit - 2

Exploring javafx Controlls

→ using image and image view, Toggle Button, Radio Button, checkbox, list view, combo Box, Textfield, Scroll Pane, tree view.

Introducing fx and transforms -

Adding Tooltips, disabling a Control,

Javafx menus - Menu Basics, And overview of menu Bar, menu and menu items,

Create a main menu, add mnemonics and accelerators to menu item, add images to the menu items, use radio menu item and check menu item create a Context menu, create a tool bar, menu demo program.

using image and image view:-

Image class: In Javafx we have the following two classes in order to load an image and display an image in a scene those are

1) Image
2) Image view

These two classes are packeged in "javafx.scene.image"

Image class: The image class is used to load an image either using an inputstream or a string URL. It has the following types of constructors.

1) Image()
2) Image(InputStream is)
3) Image(InputStream is, double width, double height, boolean preserveRatio, boolean smooth);
4) Image(String url);
5) Image(String url, boolean backgroundload);
6) Image(String url, double width, double height, boolean preserve ratio, boolean smooth, boolean backgroundload);

where 'is' is an InputStream class object and URL is a url of an image, width is a image width, height is an image height, preserve ratio

is a preserve aspect ratio of an image. A smoother is a true, when an algorithm is working to load an image and it is very fast. otherwise, it is very slow.

Image view:- An Image view is a class which will be used for displaying an image in a root node (layout) an image view is also a node. so, that we can add this to the other node just like a controls.

*It has the following constructions:

```
Image view();
Imageview(Image im);
Imageview(String url);
```

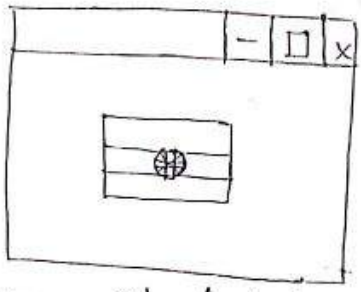
Note: if we are using an image view default constructor then we have to use setImage() method to set an Image to the Imageview. where 'im' refers an image class object.

Ex:- Programme on image

```
import javafx.application.*;
import javafx.scene.*;
import javafx.stage.*;
import javafx.scene.layout.*;
import javafx.scene.image.*;
import java.io.*;
public class JavafxImage extends Application
{
 public void start(stage ps) throws Exception
 {
 FileInputStream is = new InputStream("D:/Images/
                                       India.jpg");
 Image im = new Image(is);
 Imageview imv = new Imageview(im);
 Stackpane sp = new stackpane();
  sp.getchildren().add(imv);
  imv.setFitwidth(300);
  imv.setFitHeight(200);
 Scene s = new scene(SP, 400, 300);
     ps.setScene(s);
     ps.setTitle("Javafx Image");
     ps.show();
 }
```

```
public static void main(String[] args)
{
launch(args);
}
}
```

output:



using image with label:-

We can create an image with label using following label constructor type.

Label(String, Node);

If we are using Label String constructor then we can add a node using setGraphics() method. But in third type of constructor we are directly mentioning the node.

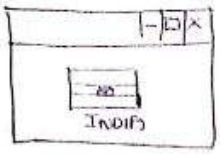
Ex:- Label l = new Label("India", imv);

where 'imv' is a object of image view class.

* we can specify the position of content relative to the label text using setContentDisplay(). It has the following position values

1) BOTTOM, 2) TOP, 3) RIGHT, 4) LEFT 5) CENTER 6) TEXTONLY 7) GRAPHICS ONLY

Ex:- l.setContentDisplay(ContentDisplay.TOP);



Ex:- programme

```
import javafx.application.*;
import javafx.scene.*;
import javafx.stage.*;
import javafx.scene.layout.*;
import javafx.scene.image.*;
import java.io.*;
public class LabelImage extends Application
{
public void start(Stage ps) throws Exception
{
```

```
FileInputStream is = new FileInputStream("D:\Images\India.jpg");
Image im = new Image(is);
ImageView imv = new ImageView(im);
StackPane sp = new StackPane();
 sp.getChildren().add(imv);
  imv.setFitWidth(300);
  imv.setFitHeight(200);
  Label l = new Label("India", imv);
  l.setContentDisplay(ContentDisplay.Top);
   Scene s = new Scene(sp, 400, 300);
     PS.setScene(s);
     PS.setTitle("Label Image");
     PS.show();
 }
public static void main(String[] args)
{
 launch(args);
}
}
```

using images with Button:-

We can create an image with a button using following type of constructor Button(String, Node); If we are using Button String then we can add a Node using setGraphics(). we can also specify position of content relative to the Button text using setContentDisplay() just like as label

Ex:-
```
Button b = new Button("India", imv);
    b.setContentDisplay(ContentDisplay.Top);
    b.setContentDisplay(ContentDisplay.GRAPHIC_ONLY);
```

Ex:- Programme:

```
public Class ButtonImage extends Application
{
public void Start(Stage PS) throws Exception
{
FileInputStream is = new FileInputStream("D:\Images\India.jpg");
Image im = new Image(is);
```

```
Imageview imv = new Imageview (im);
 Stackpane sp = new Stackpane();
  sp.getchildren().add(imv);
   imv.Setfitwidth(300);
   imv.Setfit Height(200);
  Button b=new Button("India", imv);
   b.setContentDisplay (contentDisplay.GRAPHIC_ONLY);
    Scene S=new Scene (SP, 400, 300);
      PS.Setscene(s);
      PS.Set Title("Button Image");
      PS.Show();
   }
Public Static void main(string[] args)
{
 launch (args);
}
}
```

Toggle Button:-

An useful variation of a button is called ToggleButton. A ToggleButton is simply like a pushed Button (button) but it acts differently because it has two states those are pressed (on) selected and released (off) deselected.

The ToggleButtons are encapsulated in ToggleButton class which is packaged in javafx.scene.Control.

The ToggleButton class always implements Toggle Interface because all two states buttons are implements Toggle Interface. It has the following three types of Constructors.

```
Toggle Button ();
Toggle Button (string str);
Toggle Button (string str, Node n);
```

when a user clicks on Toggle Button it was pressed rather than popping back up as a normal Button. when you click again on the same Toggle Button it will pop up (released)

A Toggle Button has two states selected and deselected. programatically to identify these two

states. we have the following methods

boolean isSelected()

It will return true when the ToggleButton is selected. It will return false when the ToggleButton is deselected.

Ex- write a javafx programme to demonstrate ToggleButton.

```
public Class JavafxToggle extends Application
{
public void start(stage ps)
{
Label msg = new Label("your selection: None");
msg.setFont (Font.font("Britannia Bold", FontWeight.
                                    EXTRABOLD,25));
Label l = new Label("select the subject");
l.setFont(Font.font("Britannia Bold", Font weight.BOLD,
                                    15));
ToggleButton tb1 = new ToggleButton ("Advanced Java");
Toggle Button tb2 = new Toggle Button("Compiler design");
Toggle Button tb3 = new Toggle Button("Python Progra-
                                    mming");
Toggle Button tb4 = new Toggle Button("Computer Net-
                                    works");
HBox hb = new HBox();
hb.getchildren().addAll(tb1, tb2, tb3, tb4);
hb.setspacing(10);
VBox root = new VBox();
root.getchildren().addAll(msg, l, hb);
root.setspacing(10);
root.SetAlignment(Pos.CENTER);
root.SetBackground(Background.EMPTY);  root.setMinsize(600,400);
Scene S = new Scene(root, 650, 450, Color.PINK);
ps.setscene(S);
ps.SetTitle("ToggleButton");
ps.show();
tb1.setonAction(new EventHandler<ActionEvent>()
{
public void handle(Actionevent ae)
{
if(tb1.isselected())
```

```
{
msg.setText("your selection:"+ tb1.getText());
}
else
{
msg.setText("your Selection : None");
}
tb2.setOnAction(new EventHandler<ActionEvent>()
{
Public void handle(ActionEvent ae)
{
if(tb2.isSelected())
{
msg.setText("your selection:"+ tb2.getText());
}
else
{
msg.setText("your selection: None");
}
tb3.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
if(tb3.isSelected())
{
msg.setText("your selection:"+ tb3.getText());
}
else
{
msg.setText("your selection: None");
}
tb4.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
if(tb4.isSelected())
{
msg.setText("your selection:"+ tb4.getText());
}
else
```

```
{
msg.setText("your selection : none");
}
}
public static void main(String[] args)
{
launch(args);
}
}
```

*we will add mutual Exclusive nature to the toggle Buttons to add this we have to configure all toggle buttons in a group. when we add toggle buttons to the group if any one of the button is selected in a group then any previous that selected button is automatically deselected.

we can create a button group using Toggle group class as follows

```
ToggleGroup tg = new ToggleGroup();
```

we can add Toggle buttons to the group using following method

```
void setToggleGroup(ToggleGroup tg);
```

for Example, if we have four Toggle Buttons like tb_1, tb_2, tb_3, and tb_4 then we can add all this four buttons to the group as follows.

```
tb1.setToggleGroup(tg);
tb2.setToggleGroup(tg);
tb3.setToggleGroup(tg);
tb4.setToggleGroup(tg);
```

(or)

we can add all toggle buttons to the group as follows.

```
tg.getToggles().addAll(tb1, tb2, tb3, tb4);
```

we can select a toggle button automatically that means without clicking on it we have the following method.

```
void setSelected(Boolean val)
```

Ex:- tb_1.setSelected(true);

The setselected method simply Selects the Button but it won't perform any action we have another method to perform selection and the respective button. Action event() method is void fire();

Ex:- tb1.fire();

Adding change event to the ToggleGroup:-

By adding change event to the Toggle group we can handle the change event by creating only one event handler to the entire group rather than to create an individual event handlers for each and every button in a group

* First we have to registered for the change event to handle change event we have to implement changelistener Interface which is packaged in "javafx.Beans.value.*"
* To register ChangeListener we have to use addListener() on the object return by selectedTogglepropertly().
*The ChangeListener defines only one method that is Changed().

Syntax:- void Changed(ObservableValue<? extends T> Changed, T oldvalue, T newvalue);

where observable value is always watched, the changes which are happens in a group the old value is a previous state Button state. The new value is a new Button state.

```
Ex:- ChangeListener<Toggle> Listener =new ChangeListener<Toggle>(
     {
     public void Changed(ObservableValue<? extends Toggle>
                         changed, Toggle old, Toggle new)
     {
     ToggleButton tb = (ToggleButton)new;
     msg.setText("your selection :"+ tb.getText());
     }
    };
    tg.selectedToggleproperty().addListener(Listener);
```

Radio Button:

The another useful variation of Button is called RadioButton. The RadioButton is a subclass of ToggleButton that means it is a specialized form of ToggleButton it is also having states like selected and deselected. It has the following constructor types

1) RadioButton()
2) RadioButton(String str);
3) RadioButton(String str, Node n);

* To add mutual exclusive nature to the RadioButton we have to configured all RadioButtons in a group. It is also implements Toggle interface. We can create a group using ToggleGroup class we can add RadioButtons to the group using setToggleGroup(). we can check whether the RadioButton is selected or not using isSelected(). Automatically we can set a Button to be select using setSelected().

* we can add change events by implementing ChangeListener interface it is just like as ToggleButton

Ex:- ChangeListener<Toggle> listener = new ChangeListener<Toggle>()
{
public void changed(ObservableValue<? extends Toggle>
changed, Toggle old, Toggle new)
{
RadioButton rb = (RadioButton) new;
msg.setText("Your Selection:" + rb.getText());
}
};

tg.selectedToggleProperty().addListener(listener);

Write a javafx program to demonstrate RadioButton and its EventHandlers.

```
import javafx.Beans.value.*;
Public class JavafxRadioButton extends Application
{
Public void Start(Stage ps)
```

```
{
Label l = new Label("Select the Transport Type");
l.setFont(font.font("Britannic Bold", FontWeight.BOLD,15));
Label msg = new Label("Transport selected is: NONE");
msg.setFont(font.Font("Britannic Bold", fontweight.EXTRA.BOLD,25));
  RadioButton rb1 = new RadioButton("Car");
  RadioButton rb2 = new RadioButton("Bus");
  RadioButton rb3 = new RadioButton("Train");
  RadioButton rb4 = new RadioButton("Aeroplane");
  ToggleGroup tg = new ToggleGroup();
   tg.getToggles().addAll(rb1, rb2, rb3, rb4);
   HBox hb = new HBox();
   hb.getchildren().addAll(rb1, rb2, rb3, rb4);
   hb.setspacing(10);
   hb.setAlignment(POS.CENTER);
   VBox root = new VBox();
   root.getChildren(l, hb, msg);
   root.setspacing(15);
   root.setAlignment(POS.CENTER);
   root.SetMinSize(350, 200);
ChangeListener<Toggle> listener = new ChangeListener<Toggle>()
{
public void Changed(ObservableValue<? extends Toggle> changed,
                    Toggle old, Toggle new)
 {
 RadioButton rb = (RadioButton) new;
 msg.setText("Transport selected is:" rb.getText());
 }
};
tg.selectedTypeToggle property().addListener(Listener);
 Scene S = new Scene(root, 350, 200, color.PINK);
  PS.setscene(S);
  PS.setTogTitle("RadioButton");
  PS.show();
}
Public Static void main(string[] args)
```

```
{
launch (args);
}
}
```

Adding Alternative method to handle events in RadioButtons:-

we are handling change events using ChangeListener interface the observable value simply watching which button state is changed. But sometimes we require which radio button is selected instead of handling change event. To do this will use getSelectedToggle() in the following programme. The following programme will create four radio buttons and one normal button. The normal button is used to conform the selected Transport type.

Ex:- Programme:

```
Public Class JavafxRadioButton extends Application
{
Public void start (Stage ps)
{
Label l = new Label("select the Transport Type");
l.setFont (Font.font("Britannic Bold", fontweight.BOLD,15));
Label msg = new Label(" NO Transport is confirmed");
msg.setFont(Font.font("Britannic Bold", fontweight.EXTRA.
                                        BOLD, 25));
    RadioButton rb1 = new RadioButton("car");
    RadioButton rb2 = new RadioButton("Bus");
    RadioButton rb3 = new RadioButton("Train");
    RadioButton rb4 = new RadioButton("Aeroplane");
     ToggleGroup tg = new ToggleGroup();
       tg.getToggles().addAll(rb1, rb2, rb3, rb4);
       HBox hb = new HBox();
       Button btn = new Button("Confirm the
       hb.getChildren().addAll(rb1, rb2, rb3, rb4);
       hb.setSpacing(10);
       hb.setAlignment(Pos.CENTER);
```

```
Button btn = new Button("Confirm the Transport");
Seperator sp = new Seperator();
Sep.setPrefWidth(350);
VBox root = new VBox();
root.getChildren().addAll(l, hb, Sep, btn, msg);
root.setSpacing(15);
root.setAlignment(POS.CENTER);
root.setMinSize(350, 200);
root.setBackground(Background.EMPTY);
btn.setOnAction(new EventHandler<ActionEvent>()
{
Public void handle(ActionEvent ae)
{
RadioButton rb = (RadioButton) tg.getSelectedToggle();
msg.setText(rb.getText()+" is Confirmed");
}
});
Scene S = new Scene(root, 350, 200, Color.PINK);
PS.setScene(PS);
PS.setTitle("Radio Button");
PS.show();
}
Public static void main(String[] args)
{
launch(args);
}
}
```

Check Box:-

CheckBox class is Encapsulated the functionality of checkBox as we know Every CheckBox as two states those are checked and unchecked. But in javafx the checkBox has three states those are Checked, unchecked and indeterminate. the indeterminate State specifies the checkBox has not been set

(ii) it is not relevant to the situation. The check box class has the following types of constructors,

checkBox(); → default constructor

CheckBox(String str); → one parameter constructor

where str refers string relevant to the CheckBox for Example, CheckBox cb = new CheckBox("cash");

o/p: ☐ cash

By default every checkBox has only two states that is checked and unchecked, if you want to enable the indeterminate state to the CheckBox, we have the following method

void setAllowIndeterminate(boolean enable);

where enable is a true then the CheckBox has three states otherwise it has two states.

If we want to check whether the checkBox has indeterminate state or not then we have to use following method.

Boolean isIndeterminate();

If it is true the CheckBox has indeterminate state otherwise there is no Indeterminate state. we can also check whether the CheckBox is checked or not to do this we have isSelected();.

If we want to set a checked CheckBox to the GUI we will use setSelected().

void setSelected(boolean);

But it will select a particular checkBox and it won't perform any action. we have another method to set checked checkBox and we can also perform the CheckBox related action. The method is as follows:

void fire();

By default if we want to set an indeterminate state to the checkBox we have the following method.

void setIndeterminate(boolean enable);

for Example. cb.setIndeterminate(true);

o/p: ☒ cash

Checked ☑

unchecked ☐

Ex: 10 Write a javafx programme to demonstrate CheckBox Control

```
Public class CheckBoxDemo extends Application
{
CheckBox cb1, cb2, cb3, cb4;
 Label msg, tlist, l;
 Public void Start(Stage PS)
 {
tlist = new Label("Transport list is : <None>");
msg = new Label("selected Transport is : None");
  l = new Label("Select the Transport Type");
CheckBox cb1 = new CheckBox("Car");
CheckBox cb2 = new CheckBox("Bus");
CheckBox cb3 = new CheckBox("Train");
CheckBox cb4 = new CheckBox("Aeroplane");
    HBox hb = new HBox();
   hb.getChildren().addAll(cb1, cb2, cb3, cb4);
   hb.SetSpacing(10);
   hb.setAlignment(POS.CENTER);
    VBox root = new VBox();
 root.getChildren.addAll(l, hb, msg, tlist);
   root.setSpacing(15);
   root.setAlignment(POS.CENTER);
   cb2.setAllowIndeterminate(true);
   cb1.setOnAction(new EventHandler<ActionEvent>()
    {
    Public void handle(ActionEvent ae)
    {
     if(cb1.isSelected())
    {
    msg.setText("selected Transport is :"+cb1.getText());
    }
   else
   {
   msg.setText("selected Transport is: None");
   }
   getAll();
  }
});
```

```
cb2.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
if(cb2.isIndeterminate())
{
msg.setText(cb2.getText()+" is indeterminate");
}
else if(cb2.isSelected())
{
msg.setText("selected Transport is:"+cb2.getText());
}
else
{
msg.setText("selected Transport is:None");
}
getAll();
}
});
cb3.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
if(cb3.isSelected())
{
msg.setText("selected Transport is:"+cb3.getText());
}
else
{
msg.setText("selected Transport is:None");
}
getAll();
}
});
cb4.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
```

```
{
if(cb4.isSelected())
{
msg.setText("selected Transport is:"+cb4.getText());
}
else
{
msg.setText("selected Transport is:"none");
}
getAll();
}
});
Scene S=new Scene(root, 500, 300, Color.PINK);
PS.setScene(S);
PS.setTitle("CheckBox");
PS.show();
}
public void getAll()
{
String list=" ";
if(cb1.isSelected())
  list+="Car";
if(cb2.isSelected())
  list+="Bus";
if(cb3.isSelected())
  list+="Train";
if(cb4.isSelected())
  list+="Aeroplane";
if(list.equals(" ")
  list="<None>";
tlist.setText("Transport list is:"+list);
}
public static void main(String[] args)
{
launch(args);
}
}
```

Select the Transport type

☐ Car ☐ Bus ☑ Train ☑ Aeroplane

Select Transport is:

Transport list is:

List view :- List view control is used to display the list of controls which allows us to select either single (or) multiple. It is encapsulated by "List view class". It has the following two types of constructors

```
List view();
List view(observable list<T>& list);
```

where 'T' specifies the data type values (or) entries. The "List view class" is a generic type which is as follows Class ListView<T>.

* The second type of constructor we have a parameter list which is type of "observable List" which is packaged in "javafx.Collection". we have a factory method called observableArrayList(). It is used to consider the list of entries. It has the following syntax.

State(E) observable List<E> observableArrayList(E... elements);

where 'E' is any data type. It is a static type so, that we can call this method using "fx Collections class" which is packaged in "javafx.Collection".

Ex:- Listview<String> lv = new Listview(list);

observable List<String> list = "fx Collection. observable Array List

Apple
Mango
Grapes

("Apple", "mango", "Grapes");

Note: we can assign entries to the list view as follows

```
List view lv = new Listview();
lv.getItems().addAll("Apple", "mango", "Grapes");
```

Listview has its own dimensions (width and Height) but if you want to set the preferable dimension to the Listview we can using following methods.

```
void setprefwidth(double w);
Void setpref Height (double h);
void setprefsize (double w, double h);
```

In a Listview we can select the item while selecting the items may be change. so, that it has

change event. The ChangeEvent is implemented by "changeListener" interface which is packaged in "javafx.beans.value".

To listen the change event we have to register with ChangeListener. Before that First we have to written the selection Model of our listview control. The selection model is written by the following method.

Syntax:

MultipleSelection SelectionModel<T> getSelectionModel();

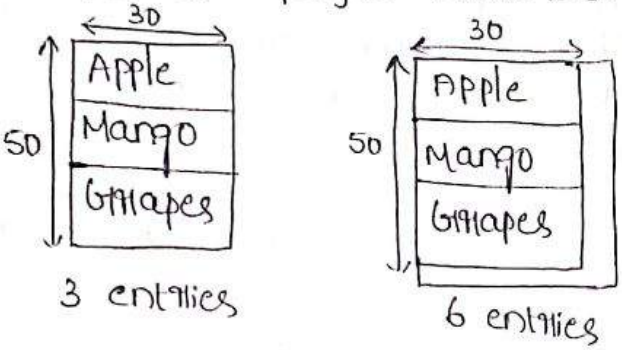
on the reference of getSelectionModel(), we can register with ChangeListener by the Selected item property method.

Syntax:- ReadOnlyObjectProperty<T> SelectedItemProperty();

Adding Scrollbars to the list view:-

The Listview has very good feature, because its automatically gets scrollbar when the list of entries are exceded a specific dimension.

Ex:-



Enable Multiple selection:- By default Listview has single selection mode by this we can select only one entry at a time. If we want to select multiple entries then we have to set multipleSelectionMode; it will happens by the following method.

void setSelectionMode(SelectionMode.mode);

where 'mode' is either SINGLE (or) MULTIPLE

we can select a list of items then if we want to display all selected items then we have to use following method.

ObservableList<T> getSelectedItems();

3(b) Write a javafx program to demonstrate Listview Control.

```
Public class ListViewDemo extends Application
{
 Label tlist, l, msg;
 Public void start (Stage ps)
 {
 tlist = new Label("Fruit list is : None");
   l = new Label("Select the fruit");
 msg = new Label("last selected fruit is : None");
 ListView lv = new ListView();
lv.getItems().addAll("Apple", "mango", "Grapes", "Banana",
                                    "Strawberry");
MultipleSelectionModel<String> lvmodel = lv.getSelectionModel()
lv.getSelectionModel().setSelectionMode(SelectionMode.
                                              MULTIPLE);
ChangeListener<String> listener = new ChangeListener<String>()
{
public void changed(ObservableValue change, String oldvalue
{                                          String newvalue)
msg.setText("last Selected Fruit is :"+new value);
String List = " ";
ObservableList<String> selected = lv.getSelectedModel().
                                    getSelectedItems();
for(String l: selected)
   list += "\n" +l;
tlist.setText("Fruit list:" + list);
}
};

lvmodel.SelectedItemProperty().addListener(listener);
 VBox root = new VBox();
  root.setSpacing(15);
  root.setAlignment(Pos.CENTER);
  root.getChildren().addAll(l, lv, msg, tlist);
  Scene S = new Scene(root, 500, 500);
  ps.setScene(S);
  Ps.setTitle("List view");
  ps.show();
 }
 Public static void main(String[] args)
 {
```

Select the fruit

Apple
mango
Grapes
Banana

```
        launch(args);
    }
}
```

ComboBox:-

The ComboBox is a control is a variation of ListView control but in ListView control we can selected either single or multiple entries or Items. In ComboBox we can select only one entry at a time, it was encapsulated by ComboBox class it has the following types of constructors

```
ComboBox();
ComboBox(observableList<T> list);
```

A ComboBox class is a generic class as follows

Class ComboBox<T>

where 'T' is a any data type.

we can add Items to the ComboBox using either observableList (or) using getItems() method. In observableList class we have a factory method called observableArrayList(). It is a static method so, that we can call this method using fxCollection class these are all packaged in "javafx.collection.*".

```
ObservableList<T> list = fxCollection.observableArrayList("
                                "abc@gmail.com",
                                "cse@gmail.com",
                                "ece@gmail.com",
                                "IT@gmail.com"),
ComboBox<String> cb = new ComboBox(list);
```

Note: we can also add entries to the ComboBox using getItems() as follows

```
ComboBox cb = new ComboBox();
cb.getItems().addAll("abc@gmail.com",
                     "cse@gmail.com",
                     "ece@gmail.com",
                     "IT@gmail.com");
```

we can also edit the ComboBox value using following method

```
void SetEditable(boolean);
Cb.SetEditable(true);
```

we can directly get the SelectedItem using following method.

```
T getvalue();
Cb.getvalue();
```

we can also set new value to the ComboBox we will use setValue() method.

```
void setvalue(T);
Cb.setvalue("Bpharmacy@gmail.com");
```

* The ComboBox is also generate ChangeEvent. If we want to handle ChangeEvent we have to Implement ChangeListener Interface.

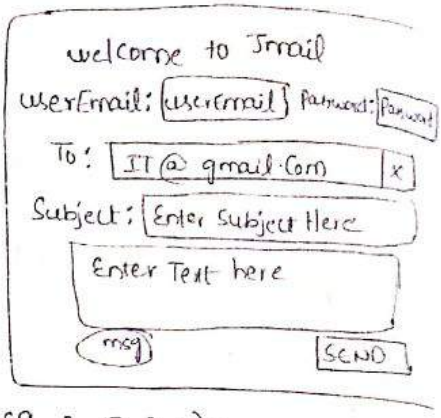
we can call the ChangeListener() Interface using addListener(). The addListener() will be called on the return value of valuePropertY(); method.

3c) write a javafx programme to demonstrate ComboBox.

```
Public Class ComboBoxDemo extends Application
{
public void start(stage ps)
{
Text t = new Text();
t.setText("welcome to Jmail");
  label user = new Label("user Email:");
Textfield utf = new Textfield();
  utf.setPromptText("userEmail:");
  Label pass = new Label("Password:");
  Passwordfield ptf = new passwordfield();
   ptf.setPromptText("password:");
  Label to = new Label("To:");
ComboBox Cb = new ComboBox();
   Cb.getItems().addAll("cse@gmail.com",
                        "ece@gmail.com",
                        "EEE@gmail.com");
   Cb.setPreferableSize(310, 20);
   Cb.setEditable(true);
   Label Subject = new Label("subject:");
   Textfield Stf = new Textfield();
   Subject.setPromptText("subject:");
     TextArea ta = new TextArea();
```

```
ta.setPromptText("Enter text:");
Button send = new Button("SEND");
Label msg = new Label();
GridPane gp = new GridPane();
    gp.add(t,0,0,4,1);
GridPane.setHAlignment(t, HPos.CENTER);
    gp.add(user,0,1);
    gp.add(utf,1,1);
    gp.add(pass,2,1);
    gp.add(ptf,3,1);
    gp.add(subject,to,0,2);
    gp.add(cb,1,2,3,1);
    gp.add(subject,0,3);
    gp.add(stf,1,3,3,1);
    gp.add(ta,0,4,4,1);
    gp.add(send,3,5);
    gp.add(msg,0,5,3,1);
GridPane.setHAlignment(send, HPos.RIGHT);
gp.setAlignment(Pos.CENTER);
cb.valuePropertY().addListener(new ChangeListener<String>()
{
Public void Changed(ObservableValue ov, String old, String
                                                    newvalue)
{
address = new value;
});
Send.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
if(cb.getvalue()!=null)
{
msg.setText("your message was successfully sent to:"
                                                + address);
    msg.setTextFill(Color.GREEN);
        utf.clear();
        ptf.clear();
        cb.setValue(null);
        stf.clear();
        ta.clear();
    }
    else
```

output:



```
{
msg.setText("you have not selected a recipant");
msg.setTextFill(Color.RED);
}
}
});
Scene SC = new Scene(gp, 500, 200);
  PS.setScene(sc);
  PS.setTitle("ComboBox");
  PS.Show();
}
public static void main(String[] args)
{
launch(args);
}
}
```

Text field: JavaFx provides various text controls, the most useful text control is Textfield. The Textfield control is used to enter a line of text. It was encapsulated by Textfield class. It has the following two types of constructors.

```
TextField();
TextField(String str);
```

we can create a textfield control as follows

```
TextField tf = new TextField();
```

Here we are using default constructor so that we have to use setText(). In order to set a Text into a Textfield.

```
void setText(String str);
```

Ex:- `tf.setText("CSE");`

CSE

we can get a Text from Textfield using getText()

Ex:- `String str = tf.getText();`

we can also increase the size of Textfield using setPreferrable Columns Count()"

we can also set a prompting message in a blank Textfield &. It always prompts to the user whenever he try to enter the Text into a blank Textfield.

```
tf2.setPromptText("Enter Text here");
```

write a javafx programme to demonstrate Textfield Control.

```
public class TextfieldDemo extends Application
{
  public void start(Stage ps)
  {
  Label source = new Label("source");
  TextField tf1 = new Textfield();
tf1.setPromptText();
  Label destination = new Label("Destination");
  Textfield tf2 = new Textfield();
  tf2.setPromptText();
    Label msg = new Label();
    Button mv = new Button("MOVE");
    Button cl = new Button("Clear");
      GridPane gp = new GridPane();
        gp.add(source, 0, 0);
        gp.add(tf1, 0, 1);
        gp.add(des, 0, 2);
        gp.add(tf2, 0, 3);
        gp.add(mv, 1, 2, 2);
        gp.add(cl, 1, 3, 2);
        gp.add(msg, 0, 5, 4, 1);
    mv.setOnAction(new EventHandler<ActionEvent>()
    {
    public void handle(ActionEvent ae)
    {
    if (!tf1.getText().isEmpty())
    {
    tf2.setText(tf1.getText());
    tf1.clear();
    }
  else if(!tf2.getText().isEmpty())
    {
    tf1.setText(tf2.getText());
    tf2.clear();
    }
  else
  {
```

```
msg.setText("Please provide data");
msg.setTextFill(Color.RED);
}
}
});
cl.setOnAction(new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
tf1.clear();
tf2.clear();
msg.setText("");
}
});
Scene S = new Scene(gp, 500,200);
ps.setScene(s);
ps.setTitle("TextField");
ps.show();
}
public static void main(String[] args)
{
launch(args);
}
}
```

Scroll pane:

A scroll pane is used to scroll the content of a control (or) node in a scene. Generally it will be used whenever a content of control does not fit in within the dimensions of a scene (or) window. A scrollpane is encapsulated by a scrollpane class it was packaged in javafx.scene.control.*.

It has the following two types of constructors

```
Scrollpane();
Scrollpane(Node n);
```

If we are using default constructor type then we can set the content to the scrollpane using setContent().

```
void SetContent(Node n);
```

The scroll pane has a good feature that is it will

automatically add scroll bars. It has its own dimensions, we can also change its viewport width and height. The viewport is a visible area of a scroll pane we can use the following two methods in order to change viewport dimensions.

```
void setPrefViewportWidth(double w);
void setPrefViewportHeight(double h);
```

In scroll pane we can also pan the content of a control by dragging mouse on the viewport area by default this future is in off state we can enable these using following method.

```
void setpannable(boolean enable);
```

we can also set the scroll bar positions using following methods by default the scroll bar positions are zeros.

```
void setHvalue(double h);
void setVvalue(double v);
```

The scroll bar positions may be increased or decreased (maximize or minimize) using following methods

```
void setHmax(double m);
void setVmin(double min);
```

we can also set a scroll bar policies that is you can always add a scroll bar, you never add a scroll bar, and you can add a scroll bar as needed. so, that we have the following three policies.

```
ALWAYS
NEVER
AS_NEEDED
```

we have the following method to set the scroll bar policy to the scroll pane.

```
void setHbarpolicy(scrollpane.scrollBarpolicy value);
void setvbarpolicy(scrollpane.scrollBarpolicy value);
```

Ex:- 49 write a javafx programme to demonstrate scroll pane.

```
Public Class ScrollpaneDemo extends Application
{
public void start(Stage ps)
{
Button b= new Button("RESET SCROLL BAR");
```

```
Image im = new Image(getclass().getResourceAsStream
                                ("aits.jpg"));
Imageview imv = new Imageview(im);
 imv.SetFitForWidth(300);
 imv.SetFitForHeight(400);
 Scrollpane sp = new Scrollpane();
  Sp.setContent(imv);
  Sp.setPrefviewportwidth(300);
  Sp.setPrefviewportHeight(400);
  Sp.setpannable(true);
  Sp.setHbarpolicy(ScrollPane.ScrollBarpolicy.NEVER);
  Sp.setvbarpolicy(Scrollpane.ScrollBarpolicy.AS_NEED);
   VBox root = new VBOX();
    root.getchildren.addAll(b, Sp);
 b.setonACTION(new EventHandler<ACTIONEvent>()
{
public void handle(ACTIONevent ae)
{
 Sp.setHvalue(0);
 Sp.setVValue(0);
}
});
Scene S = new Scene(root, 300, 250);
 PS.setScene(S);
 PS.setTitle("SCROLL PANE");
  PS.Show();
}
Public static void main(String[] args)
 {
  launch(args);
}
}
```

Tree View:-

The tree view is used to create conceptual simple tree based data structure in this tree we have different nodes those are root node the root node may have one or more childrens those

are terminal nodes or leaf nodes and branch nodes.
* The branch node will again create a subtree in a larger tree. we can create a tree view using "Treeview" class.

It has the following constructors

```
Treeview();
Treeview(TreeItem<T> root);
```

It is a generic type. so that the format of Generic class as follows.

```
Class Treeview<T>
```

we can create the items which are childrens of a tree. using "TreeItem" class it is also a Generic type.

```
Class TreeItem<T>
```

It is used to create nodes in a tree that may be any type of node we can add this nodes either using add (or) addAll().

* In Treeview we have selection event before handle this events first we have to know the selection model using getSelectionModel() then we have to use the selectedItemProperty() to select the property of particular item. on the return value will call addlistener() in order to handle the selection events.

To handle Events we will implement changed() which is in Changelistener Interface. These handle events are similar to listview's control.

* To get a value of particular selected Item we will use getvalue() and we can also get a parent and child nodes using getparent(), getchildren(). If you are using a Treeview default construct then we can set the root node to the treeview using setRoot().

* In TreeItem class we have the following constructors

```
TreeItem();
TreeItem(T value);
TreeItem(T value, Node n);
```

4b) write a javafx programme to demonstrate Treeview.

```
public class TreeviewDemo extends Application
{
int i;
Image[] im = new Image[10];
```

```
Imageview imv = new Imageview();
public void start(stage ps)
{
label msg = new Label("No Preview", imv);
msg.setContentDisplay(ContentDisplay.TOP);
TreeItem root = new TreeItem("Images");
TreeItem f = new TreeItem("Flowers");
f.getChildren().add(new TreeItem("flower0");
f.getchildren().add(new TreeItem("flower1");
f.getChildren().add(new TreeItem("flower2");
TreeItem v = new TreeItem("Flowers");
                          vehicle
v.getchildren().add(new TreeItem("vehicle 0");
v.getChildren().add(new TreeItem("vehicle 1");
v.getChildren().add(new TreeItem("vehicle 2");
root.getchildren().addAll(f,v);
Treeview tv = new Treeview();
 tv.setRoot(root);
Stackpane left = new stackpane(tv);
stackpane Right = new stackpane(msg);
 Splitpane sp = new splitpane();
 Sp.getItems().addAll(left, Right);
  Sp.setDividesPositions(0.2f, 0.8f);
 for(i=0; i<7; i++)
 {
 im[i] = new Image(getClass().getResourceAsStream
                                    (i+".jpg"));
 }
multipleSelectionModel<TreeItem<String>> tvModel =
                                tv.getSelectionModel();
ChangeListener<TreeItem<String>> listener = new Change
{                               Listener<TreeItem<String>>()
public void Changed(observable list Change, TreeItem<String>
                        old, TreeItem<String> new value)
{
String path = newvalue.getvalue();
TreeItem<String> tmp = new value.getparent();
while(tmp != null)
{
```

```
Path = tmp.getvalue()+"-->"+path;
} tmp = tmp.getparent();
if (newvalue.getvalue()!=null)
{
Switch(newvalue.getvalue())
{
Case "IMAGIES":
    imv.setImage(null);
    msg.setGraphic(imv);
  msg.setText("selected is:"+newvalue.getvalue()+"Complete
                                    Path is:"+path);
  break;
Case "FLOWER":
    imv.SetImage(null);
    msg.setGraphic(imv);
  msg.setText("selected is:"+newvalue.getvalue()+"Complete
  break;                            path is:"+path);
Case "FLOWER 0":
    imv.setImage(im[0]);
    msg.setGraphic(imv);
  msg.setText("selected is:"+newvalue.getvalue()+"Complete
   break;                           path is:"+path);
Case "FLOWER 1":
    imv.setImage(im[1]);
    msg.setGraphic(imv);
  msg.setText("selected is:"+newvalue.getvalue()+"Complete
    break;                          Path is:"+path);
Case "FLOWER 2":
    imv.setImage(im[2]);
    msg.setGraphic(imv);
  msg.setText("selected is:"+new value.getvalue()+"Complet
                                    path is:"+path);
Case "VEHICLE":
    imv.setImage(null);
    msg.setGraphic(imv);
  msg.setText("selected is:"+new value.getvalue()+"
                                Complete path is:"+path);
Case "VEHICLE 0":
    imv.SetImage(im[3]);
```

```
        msg.setGraphic(imv);
        msg.setText("selected is:"+new value.getvalue()+"Complete
                                                path is:"+path);
        break;
        case "VEHICLE1":
            imv.setImage(im[1]);
            msg.setGraphics(imv);
        msg.setText("selected is:"+new value.getvalue()+"Complete
                                                path is:"+path);
        break;
        case "VEHICLE 2":
            imv.setImage(im[I]);
            msg.setGraphics(imv);
        msg.setText("selected is:"+new value.getvalue()+"complete
                                                path is:"+path);
        break;
    }
  }
 }
});

    model.selectedItemProperty().addListener(listener);
    Scene S = new Scene(800,400);
        PS.setTitle("TREEview");
        PS.setScene(S);
        PS.show();
    }
public static void main(String[] args)
{
launch(args);
}
}
```

Introducing Effects and Transforms:

The Effects and Transforms provides a sufisticated GUI's and also provides the modern look for the Controls. effects

Effects: The Effects are encapsulated by Effects class it was packaged in "Javafx.scene.effects"

we have different types of effects in a javafx some of them are listed below.

1) Bloom:- It is used to highlight the part of the control.

2) Box Blur:- It is used to create Blur type

3) Drop shadow: It is used to create a shadow behind the control.

4) Inner shadow: It is used to create an inner shadow in a control.

5) Glow:- It is used to set the different levels of glow the level range is 0.0 to 1.0.

6) Lightening:- It is used to create a light shadow in a control.

7) Reflection:- It is used to create Reflection of a control.

we can apply all above effects on a specific control. if you want to set these effects we have to use following method.

```
Void setEffects(Effects obj);
```

To demonstrate Effects in Javafx it is as easy to implement glow and inner shadow effects.

Glow:- It is encapsulated by Glow class it has the following Constructor

```
Glow();
Glow(double level);
```

If we are using default Constructor then we can set the level using setLevel()

```
void setLevel(double level);
```

we can add (or) remove these effects on controls to add we can use setEffects() to remove effects we can also use SetEffects() but the passing parameter is null.

Inner shadow:- It was Encapsulated by InnerShadow class it has the following Constructors

1) InnerShadow();

2) InnerShadow(double radius, Color color);

If we are using default type then we can set radius and color using following methods. SetRadius() and SetColor(). and we can also set height and width of inner shadow using setHeight() and setWidth().

Transforms:- we have the following transforms classes it are encapsulated by Transforms class and it is packaged in "Javafx.scene.Transforms"

The transform classes are

1) Rotate
2) Scale
3) Shape, Shear
4) Translate

It is easy to demonstrate transforms using Rotate and Scale. The Rotate class is used to Rotate the control with specific angle at specific positions we can set the angle using setAngle() and we have the following types of constructors

```
Rotate()
Rotate(double Angle, double x, double y);
```

we can also set x and y values using

```
setTranslateX();
SetTranslateY();
```

we have another class to Rotate particular control A class is RotateTransition. In this class we have the following methods

1) SetAxis();-

It is used to set the Rotating Axis. for Example

```
Rt.setAXIS(Rotate.X-AXIS);
                  ↓
                  Z
```

2) SetByAngle():- It is used to set the Rotating Angle

Ex:- Rt.setByAngle(360);

3) SetCycleCount():- It is used to set the Cycle Counts

Ex:- Rt.SetCycleCount(Animation.INDEFINITE);

import "Javafx.Animation".

4) SetInterpolation():- It is used to set either linear or discrete interpolation.

Ex:- Rt.setInterpolation(Interpolation.LINEAR);

5) SetAutoReverse():- It is used to setAutoReverse

Rt.setAutoReverse(boolean);

Ex:- Rt.SetAutoReverse(true);

6) SetDuration():- It is used to set the speed of Rotation. It is packaged in

"javafx.util package".

Ex: rt.setDuration(Duration.millis(1000));

1) SetNode(): It is used to set a node to the Rotation

Ex: rt.setNode(c); // circle object

2) setplay(): It is used to play a rotate transform

Ex: rt.play();

3) setStop(): It is used to stop the rotate transition

Ex: rt.stop();

The scale class is used to increase or decrease the size of a node at specific positions x and y. we have the following constructors

```
Scale();
Scale(double x, double y);
```

If we are using default constructor then we can used setX() and setY() to set x and y values

```
void setX(double x);
void setY(double y);
```

4c. write a javafx program to demonstrate rotate, scale, glow, and innerShadow

```
public class weekye extends Application
{
public void start(Stage ps)
{
Image im = new Image(getClass().getResourceAsStream
                    ("image.png"));
ImageView imv = new ImageView(im);
RotateTransition rt = new RotateTransition();
    Scale s = new Scale();
    Glow g = new Glow();
  InnerShadow is = new InnerShadow();
    Button b1 = new Button("Rotate");
    Button b2 = new Button("scale");
    Button b3 = new Button("Glow");
    Button b4 = new Button("Shadow");
    Button b5 = new Button("Reset");
    GridPane root = new GridPane();
     root.add(imv, 0, 0, 4, 1);
     root.add(b1, 0, 1);
     root.add(b2, 1, 1);
```

```
root.add( b3,2,1);
root.add( b4,3,1);
root.add(b5, 0,5,4,1);
root.setHalignment (b5, HPos.CENTER);
root.setVgap(15);
Scene SC = new Scene (root, 300, 300);
  PS.setScene(SC);
  PS.setTitle ("Effects & Transforms");
  PS.show();
b1.setOnAction(new EventHandler <ActionEvent> ()
{
public void handle (ActionEvent ae)
{
rt.setAxis(Rotate.Z_AXIS);
rt.setByAngle (360);
rt.setCycleCount(Animation.INDEFINITE);
rt.setInterpolator(Interpolator.LINER);
rt.setDuration(Duration.millis(1000));
rt.setNode(imv);
rt.play();
}
};
b2.setOnAction(new EventHandler <ActionEvent> ()
{
public void handle(ActionEvent ae)
{
rt.setAxis(Rotate.Z_AXIS);
rt.setByAngle (360);
rt.setCycleCount(Animation.INDEFINITE);
rt.setInterpolator(Interpolator.LINER);
rt.setDuration( Duration.millis(1000));
rt.setNode(
  s.setX(0.5);
  s.setY(0.5);
}
};
b3.setOnAction(new EventHandler <ActionEvent> ()
{
public void handle(ActionEvent ae)
{
```

```
        g.setLevel(0.9);
        inv.setEffects(g);
    }
};
by.setOnAction(new EventHandler<ActionEvent>()
{
Public void handle(ActionEvent ae)
{
 is.SetColor(Color.RED);
 is.SetRadius(.5);
 is.setHeight(25);
 is.setWidth(20);
 is.setChoke(0.9);
 inv.setEffect(is);
}
};
bs.setOnAction(new EventHandler<ActionEvent>()
{
 Public void handle(ActionEvent ae)
 {
  s.setX(1.0);
  s.setY(1.0);
  g.setLevel(0.0);
 inv.setEffect(null);
  rt.stop();
}
};
}
Public static void main(String[] args)
{
launch(args);
}
}
```

Adding Tooltip:-

A tooltip is a short message which can be displayed whenever the mouse pointer move on to the control. It is encapsulated by a tooltip class It has the following constructions

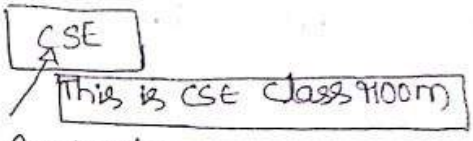
```
Tooltip();
Tooltip(String str);
```

If we want to set Tooltip to a particular Control then we have to call setTooltip()

Syn:- void setTooltip(Tooltip tooltipobj);

Ex: Tooltip tp = new Tooltip("this is CSE classroom");

Button CSE = new Button("CSE")

o/p:- 

Disabling a Control:-

we can disable a control using setDisable()

void setDisable(boolean val);

If we want to set a disable state by a control they we can call the setDisable() using control obj.

for Example Button CSE = new Button("CSE");

CSE.setDisable(true);  → Disable button

If we want to enable the button which is already disable then we can pass the boolean variable as false.

CSE.setDisable(false);  → Enable button

Javafx menus:-

The menus are used to provide more simple and suffiscated GUI. These are provide more functionality's to the application. So, that in every application menus are playing key roles. Javafx also providing different types of menu's. we have the following menus:

1) menuBar:- It is a main menu for every application

2) Standard Menu:- These are the menu's to be added a menuBar.

3) ContextMenu:- These menus is displayedwhen you press the mouse right click Button on a particular control. we have the following menu Basics

1) Menu:- we can create a menu using Menuclass These are all standard Menus.

2) MenuItem:- These are the items to be added to a particular menu. we can create MenuItems using MenuItem class.

3) Radio menuItem:- It is used to create the radiomenu

Items, so that we can select any one of the Item from the different radio menu items.

4) CheckMenuItems:- It is used to create checked Menu Items so that we can select either one or more check MenuItems at a time.

5) SeperatorMenuItem:- It is used to seperate the menu Items.

6) ContextMenu:- It is used to create the Context Menus.

An overview of menuBar, menu and menuItems:-

MenuBar:- Menubar is used to create a container for the menus. it is a main menu type for every application. It can be encapsulated by MenuBar Class. It has only one construction which is a default construction. We can add the menu's using either add or addAll(). Before using add or addAll(). first we have to call getmenu's method getMenus().

Syntax:- ObservableList <Menus> getMenus().

Here we can use a different version of add()

```
void add(int index, menu m);
```

where index is the position of the menu.

```
void add(Menu m);
void addAll(menu m1, m2 ...);
```

we can also remove menus using remove() on the return value by getMenus()

we can also remove all menus using removeAll(). we can also obtain the size of menuBar using size().

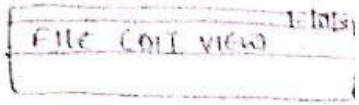
Ex:- MenuBar mb = new MenuBar();

Menu:- The menu's are encapsulated by Menu class. it is populated with menu items a menu is derived from MenuItem Class so that we can add another Menu to the Menu that means SubMenus. We can add Menu or MenuItems to the Menu. The Menuclass has the following constructors.

```
Menu();
Menu(String M);
Menu(String m, Node n);
```

Ex:- Menu file = new Menu("FILE");

```
Menu edit = new Menu("EDIT");
Menu view = new Menu("VIEW");
Mb.getMenus().add(File, edit, view);
```



we can add MenuItems using either add or addAll( ) on the return value of getItems( )

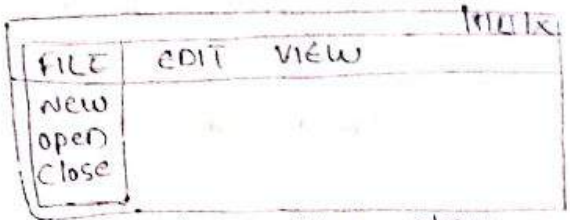
syn:- ObservableList<MenuItem> getItems( );

MenuItem:-

It is used to create MenuItems it is encapsulated by MenuItem class. it has the following types of constructors

```
MenuItem();
MenuItem(String m);
MenuItem(String m, Node n);
```

Ex:-
```
MenuItem New = new MenuItem("New");
MenuItem Open = new MenuItem("Open");
MenuItem Close = new MenuItem("Close");
file.getItems().addAll(New, Open, Close);
```



* we can perform the actions when you click a particular MenuItem the respective actions will be handle as follows:

Here we can use setOnActionMethod( ) Just like actions performed by the buttons.

void setOnAction(new EventHandler<ActionEvent>( ));

Ex:-
```
setOnAction(new EventHandler<ActionEvent>())
New
{
public void handle(ActionEvent ae)
{
String str = (MenuItem)ae.getTarget()).getText();
System.out.println("you have selected :"+str+"option");
}
});
```

Create a Main Menu:-

```
Public class JavaFxMenu extends Application
{
 Public void start (Stage PS)
 {
 MenuBar Mb = new MenuBar( );
 Menu file = new Menu ("FILE");
 Menu edit = new Menu ("EDIT");
 Menu view = new Menu("VIEW");
MenuItem New = new MenuItem ("NEW");
MenuItem open =new MenuItem("open");
MenuItem close =new MenuItem("close")
          exit                    EXIT
 MenuItem Cut =new MenuItem ("CUT");
 MenuItem Copy = new MenuItem ("Copy");
 MenuItem Paste = new MenuItem("PASTE");
   mb.getMenus().addAll( file ,edit, view);
   file.getItems().addAll(New, open, exit);
   edit.getItems().addAll(Cut, Copy, Paste);
EventHandler <ActionEvent> event = new EventHandler <Action
                                                Event>( )
{
 Public void handle(ActionEvent ae)
 {
String str = ((menuItem) ae.getText()).getText();
msg.setText("you have selected +"str" + option");
 }
};
New.setOnAction(event);
open.SetonAction(event);
exit.Seton Action (event);
Cut.SetonAction(event);
Copy.SetonACtion(event);
Paste.SetonAction(event);
 BorderPane bp = new BorderPane();
        bp.setTop(mb);
        bp.setCenter(Mb);
      Scene S = new Scene (bp, 500, 300);
        PS.setScene(S);
        PS.SetTitle("MAIN MENU");
```

```
ps.show();
}
public static void main(String[] args)
{
launch(args);
}
}
```

Add Mnemonics & Accelerators to menu Item:-

In Javafx we can add keyboard shortcuts to the Menu and MenuItems we have the following two keyboard forms one is accelerator and second one is Mnemonics.

Accelerator:- The Accelerator is a keyboard shortcut which is key combinations in javafx. for Example If we want to open. Generally we can use open MenuItem but when you set the accelerator to the open MenuItem we can open by using the shortcut key ctrl+O. Generally, we have Ctrl, Alt, shift etc. In Javafx we have a special value. "shortcuts" in the place of Ctrl for windows operating system. we can set the accelerator to the MenuItem as follows.

void setAccelerator(keyCombination.keyCombination(String))

Here the String is a shortcut keyCombination for the MenuItems. A keyCombination class is packaged in "javafx.scene.input". It has the keyCombination factory method using this Method we can assign the shortcut key combinations. for example If you want to assign a shortcut key for the Save MenuItem then we can do as follows.

```
Save.setAccelerator(keyCombination.keyCombination("ctrl+s"));
```

Mnemonics:-

The Mnemonics will be set to the Menu and MenuItems. Generally, it is preceded an underscore before the starting letter of particular Menu or MenuItem then we can activate this Menu or Menu Items by pressing Alt+ the respective Menu or MenuItem starting letter. Generally, it is in off state then we can enable this using following method.

```
void setMnemonicParsing(boolean);
```

Ex:- Menu file = new Menu("_FILE");
file.setMnemonicParsing(true);

Add images to the MenuItems:-

Generally, we can load an image using Imageclass and we can display those Images using Imageview class, then we can add this Imageview to the MenuItems using three types of constructors. Construction are as follows.

Menu(String str, Node Graphic);

MenuItem(String str, Node Graphic);

If we are using second type of Construction then we can set an image using setGraphic().

Image im = new Image(getClass().getResourceAsStream("Print.png"));
ImageView imv = new ImageView(im);
MenuItem print = new MenuItem("PRINT", imv);
(or)
MenuItem print = new MenuItem("PRINT");
Print.setGraphic(imv);

Use Radio MenuItems and Check MenuItems:-

Radio MenuItems:-

It is used to create RadioMenuItem for a particular Menu to do this we have RadioMenuItem class. It has the following three types of Constructors

RadioMenuItem()
RadioMenuItem(String)
RadioMenuItem(String, Node)

RadioMenuItem red = new RadioMenuItem("RED");

If we want to create a selected RadioMenuItem then we can use the following Method.

void setSelected(boolean);

Now, we have to create Toggle group using ToggleGroup class then we can add All RadioMenuItems to the ToggleGroup using setToggleGroup() as follows

ToggleGroup tg = new ToggleGroup();
red.setToggleGroup(tg);

We can perform Actions related to the RadioMenuItems

by using SelectedToggle property(). we can handle this by implementing changelistener().

```
Ex:- tg.selectedToggleProperty().new ChangeListener<Toggle>()
     {
     public void Changed(ObservableValue ob, Toggle old,
                         Toggle newvalue)
     {
     if(newvalue != null)
     {
     RadioMenuItem r1 = (RadioMenuItem) newvalue;
     String str1 = r1.getText();
     msg.setText("you have selected is"+str1+" Color option");
     }
   }
});
```

CheckMenuItem:-

It is encapsulated by CheckMenuItem class. It is used to create CheckMenuItems in javafx. It has the following three types of constructors.

```
CheckMenuItem();
CheckMenuItem(String)
CheckMenuItem(String str1, Node n);
```

we can set the selected CheckMenuItem using setSelected().

```
Ex:- CheckMenuItem high = new CheckMenuItem("High");
     high.setSelected(true);
```

✓High

we can perform actions by calling setOnAction().

```
Ex:- high.setOnAction(new EventHandler<ActionEvent>()
     {
     public void handle(ActionEvent ae)
     {
   String str1 = ((CheckmenuItem)ae.getToggle()).getText();
   msg.setText("you have selected +"str1"+ Color option");
   }
  }
});
```

Create Context Menu:-

In Javafx applications we have added Menus to the MenuBar we can also create a different type of Menus. These Menu's will show a pop up Menu

whenever we click Right mouse button on a Control, These type of Menus are called ContextMenu.

* It has a direct super class that is popupControl and indirect super class popupwindow. The indirect super class is packaged in "javafx.stage.popupwindow". Class is packaged in "javafx.stage.popupwindow".

* It is Encapsulated by ContextMenu class. we have the following Constructors.

```
ContextMenu();
ContextMenu(MenuItem ... MenuItems);
```

Ex:- `ContextMenu cm = new ContextMenu(red, green, blue);`

we can set context Menu to a Control we have set ContextMenu()

```
void setContextMenu(ContextMenu obj);
```

For Example, if we have a Control like Circle when you click right mouse button on the Circle. immediately it will show a Context Menu which contains red, green, blue MenuItems.

Ex:- If circle object is 'c' then we can set a ContextMenu as follows.

```
c.setContextMenu(cm);
```

where red, green, blue are RadioMenuItems.

* we can call the ContextMenu Events by Calling SetOnContextMenuRequested(). The related event is Context Menu Event. which is packaged in "javafx.scene.input".

* Inside the handle method we must implement show() which is used to display a ContextMenu at a specified locations.

```
void show(Node c, double upperx x, double upperx y)
```

we can get x and y values by using a Context Menu Event Methods those are

```
getScreenX();
getScreenY();
```

Ex:-
```
root.SetOnContextMenuRequested(new EventHandler<Context
                                MenuEvent>()
{
public void handle(ContextMenuEvent ae)
{
cm.show(c, ae.getscreenx(), ae.getscreeny());
}
});
```

Create ToolBar:-

A ToolBar is a Component which can serve as both an alternative and adjustment to the Menu.

The ToolBar has a list of Buttons which can be used to perform various program options. The ToolBar has horizontal orientation defaultly. we can also create a vertical ToolBar using setOrientation method.

syntax:- void setOrientation (Orientation how);

we have two parameter types here those are

Orientation.HORIZONTAL

Orientation.VERTICAL

All ToolBar instances are created by ToolBar class It has the following types of constructors.

ToolBar()

ToolBar(Node N1, N2...);

we can add Nodes to the ToolBar using Add() on the return value of getItems()

In general the ToolBar has image based buttons so, that Every button has an image and we can also use only buttons. we can set tooltip for a button using SetTooltip() and we can also set a Contentdisplay using setContentDisplay().

Ex:- Button b1 = new Button("Refresh");

Button b2 = new Button("search");

ToolBar tb = new ToolBar(b1, b2);

we can also set the button ACTION event handlers by using setONACTION() and we can handle those events by implementing EventHandler interface.

4d) MenuDemo program:-

```
Public Class MenuDemo extends Application
{
public void start(stage ps)
{
Image im1 = new Image(getclass().getResourceAS
                      stream("print.png"));
Image im2 = new Image(getclass().getResourceAS
                      stream("Refresh.png"));
Image im3 = new Image(getclass().getResourceAS
                      stream("search.png"));
```

```
Imageview imv1 = new Imageview (im1);
Image.view imv2 = new Imageview (im2);
Imageview imv3 = new Imageview (im3);
  imv1.setfitwidth(25);
  imv1.setFitHeight(25);
  imv2.setFitwidth(30);
  imv2.setFitHeight(30);
  imv3.setfitwidth(30);
  imv3.setFitHeight(30);
MenuBar mb = new MenuBar();
  Menu m1 = new Menu("_File");
  m1.setMnemonicParsing(true);
  MenuItem New = new MenuItem("New");
  MenuItem open = new MenuItem("OPEN");
  MenuItem save = new MenuItem("SAVE");
  MenuItem SaveAS = new MenuItem("SAVE AS");
  MenuItem Print = new MenuItem("Print", imv1);
  MenuItem exit = new MenuItem("EXIT");
m1.getItems().addAll(New, open, Save, SaveAS, Print, exit);
  Menu m2 = new Menu("_Edit");
m2.setMnemonicParsing(true);
  MenuItem cut = new MenuItem("cut");
  MenuItem copy = new MenuItem("copy");
  MenuItem paste = new MenuItem("paste");
  MenuItem delete = new MenuItem("delete");
SeperatorMenuItem sp = new SeperatorMenuItem();
MenuItem select = new MenuItem("Select All");
m2.getItems().addAll(cut, copy, Paste, delete, sp, select);
  Menu m3 = new Menu("_view");
m3.setMnemonicParsing(true);
  MenuItem find = new MenuItem("Find");
  MenuItem Replace = new MenuItem("Replace");
  MenuItem goto = new MenuItem("goto");
M3.getItems().addAll(Find, Replace, goto);
  Menu m4 = new Menu("_options");
  m4.setMnemonicParsing(true);
  Menu m41 = new Menu("_colors");
  Menu m42 = new Menu("_priority");
```

```
m4.getItems().addAll(m41, m42);
RadioMenuItem Red = new RadioMenuItem("RED");
              red          Radio
RadioMenuItem green = new RadioMenuItem("green");
RadioMenuItem blue = new RadioMenuItem("blue");
m41.getItems().addAll(red, green, blue);
ToggleGroup tg = new ToggleGroup();
 red.setToggleGroup(tg);
 green.setToggleGroup(tg);
 blue.setToggleGroup(tg);
 green.setSelected(true);
CheckMenuItem high = new CheckMenuItem("high");
     "        medium = new      "      ("medium");
CheckMenuItem low = new CheckMenuItem("low");
m42.getItems().addAll(high, medium, low);
 menu m5 = new Menu("_Help");
Me m5.setMnemonicParsing(true);
 MenuItem help = new MenuItem("Help Content");
 MenuItem about = new MenuItem("About");
  m5.getItems().addAll(help, about);
mb.getMenus().addAll(m1, m2, m3, m4, m5);
New.setAccelerator(KeyCombination.keyCombination("Ctrl+N");
Open.setAccelerator(KeyCombination.keyCombination("Ctrl+O");
Save.setAccelerator(KeyCombination.keyCombination("Ctrl+S");
SaveAS.setAccelerator(KeyCombination.keyCombination("Ctrl+SS");
Print.setAccelerator(KeyCombination.keyCombination("Ctrl+P");
Exit.setAccelerator(KeyCombination.keyCombination("Alt+X");
Label msg = new Label("No option selected");
msg.setFont(Font.font("Britannic Bold," FontWeight.EXTRA
                                        BOLD, 15));
ContextMenu cm = new ContextMenu(red, green, blue);
 Circle c = new Circle(25, Color.PINK);
 Gridpane gp = new Gridpane();
      gp.add(msg, 0, 0);
      gp.add(c, 0, 1);
   Gridpane.setHalignment(msg, HPos.CENTER);
   Gridpane.setHalignment(c, HPos.CENTER);
    gp.setAlignment(Pos.CENTER);
    gp.setVgap(15);
  Button b1 = new Button("Refresh", rnv2);
```

```
Button b2 = new Button("search", imv3);
b1.setContentDisplay(ContentDisplay.GRAPHIC_ONLY);
b2.setContentDisplay(ContentDisplay.GRAPHIC_ONLY);
b1.setTooltip(new Tooltip("Refresh"));
b2.setTooltip(new Tooltip("search"));
HBox hb = new HBox(b1, b2);
ToolBar tb = new ToolBar(hb);
BorderPane root = new BorderPane();
        root.setTop(mb);
        root.setCenter(gp);
        root.setBottom(hb);
    Scene s = new Scene(root, 500, 200, color.PINK);
     ps.setTitle("JAVAFX MENU");
     ps.setScene(s);
     ps.show();
  }
/**Public static void main(String[] args)
  {
  launch(args)
  }
} **/
EventHandler<ActionEvent> event = new EventHandler<ActionEvent>()
{
public void handle(ActionEvent ae)
{
String name = ((MenuItem) ae.getText()).getText();
 if (name.equals("Exit"))
  platform.exit();
msg.setText("you have selected "+name+" option");
}
};
new.setOnAction(event);
open.setOnAction(event);
Save.setOnAction(event);
Print.setOnAction(event);
exit.setOnAction(event);
Copy.setOnAction(event);
Paste.setOnAction(event);
delete.setOnAction(event);
Selectall.setOnAction(event);
find.setOnAction(event);
```

```
    replace.setOnAction(event);
    goto.setOnAction(event);
   helpContent.setOnAction(event);
     about.setOnAction(event);
      high.setOnAction(event);
      medium.setOnAction(event);
      low.setOnAction(event);
ChangeListener<Toggle> listener1 = new ChangeListener<Toggle>()
{
public void Changed(observableValue ,ob, Toggle old, Toggle newvalue)
{
 if (newvalue!=null)
 {
RadioMenuItem r =(RadioMenuItem) newvalue;
 String clr1 = r.getText(); -> msg.setText("you have selected "+
   c.setFill(Color.web(clr1));                    clr1+"coloroption");
  }
 }
};
tg.selectedTogglePropertly(listener1); .addListener(listener1);
EventHandler<ContextMenuEvent> cmevent = new EventHandler
                                          <ContextMenuEvent>()
 {
 public void handle(ContextMenuEvent ae)
 {
cm.show(c, ae.getscreenx(), ae.getscreeny());
 }
};
root.setOnContextMenuRequested((cmevent);
}
}
};
public void main (String[] args)
{   static
launch(args);
}
}
```