# Lesson 12

# *Graphical User Interface*

By the end of this lesson you will learn:

- What is Graphical User Interface?
- Components of a GUI
- Java Library Classes for making a GUI
- Customizing the color and font of a GUI component
- Different EventListeners
- Steps to design a GUI
- Designing a GUI

---

It is obvious that in real life there is a very little usage of console based applications rather than applications that has graphical components to work with. In this lesson, we will be learning about how to design a graphical user interface.

## What is Graphical User Interface?

A user interface is the medium that a user uses while interacting with a system. The programs we have done so far are called Console Applications as console is the interface that the user is interacting with. User is giving input in the console and also getting the output in the console.

A user interface that uses different graphical components for users to interact with the system is known as a Graphical User Interface. Example:

- The application you are using to read this document is a GUI based application and anything that you can click here is a GUI component.
- The text editor you use while coding is also a GUI based application and even when you are typing, you are typing in a GUI component.

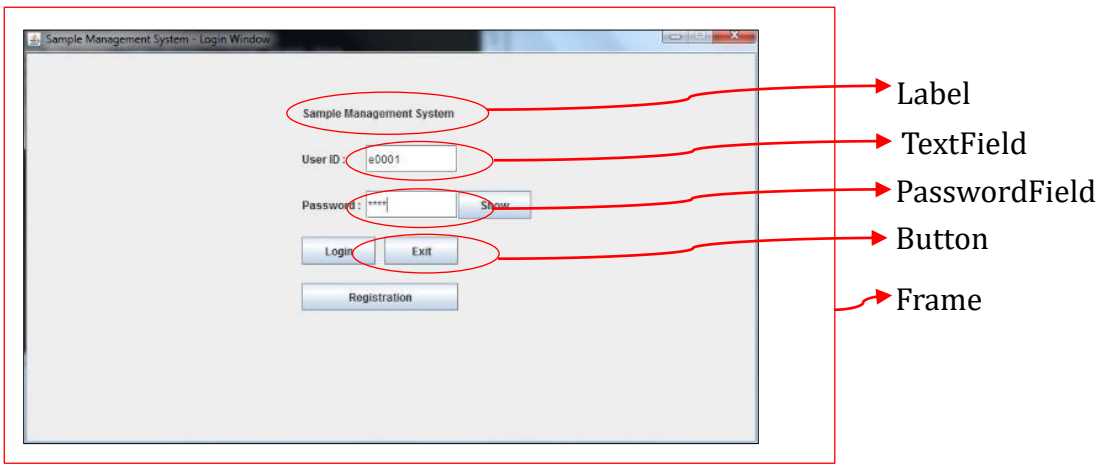
A GUI makes a user interface more attractive and easier to work with. We can even customize GUI components to make it better. If GUI was not here, using a computer would have become boring and irritating.

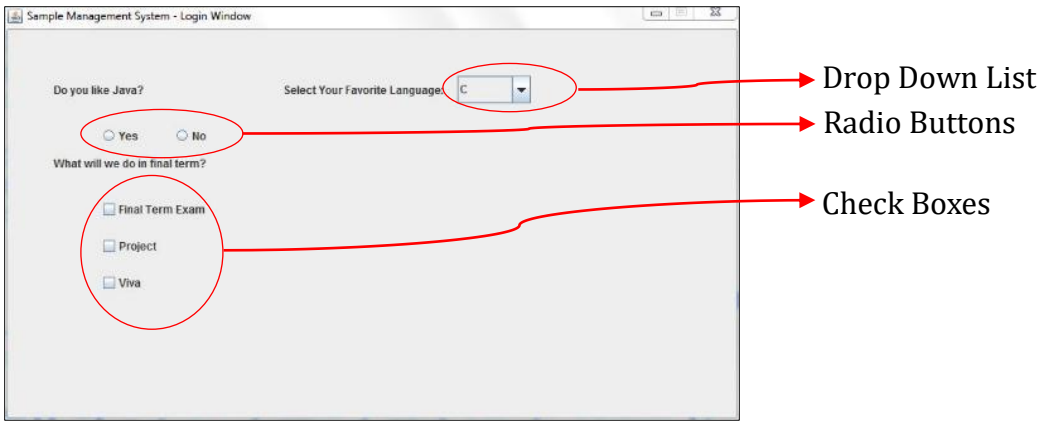
## Components of GUI

There are numerous GUI components. These components can be used to develop a GUI as per requirements. Some basic and frequently used GUI components are:

| Name of Component | General Purpose |
|---|---|
| Label | View plain text |
| TextField | Type anything as Text |
| PasswordField | Type anything as Password |
| Button | Trigger any event or to navigate |
| Radio Button | Select any Option |
| Check Box | Select Multiple Options |
| Button Group | Grouping Multiple Radio Buttons or Check Boxes |
| Drop Down List | View items as a list and select one from the list |
| Text Area | Type anything as Text in a large area |
| Image | View an Image |
| Table | View Data in a Table |
| Panel | Create a panel |
| Frame | Design a GUI |
| Notification/Message Box | Display any notification or Message. |

The following images show some of the components of a Graphical User Interface:



## Java Library Classes for Making a GUI

All the above mentioned GUI components can be created using java library classes. These classes are inside javax.swing package. The following table shows the java library classes for each pf the above mentioned GUI components:

| Name of Component | Java Library Class Name |
|---|---|
| Label | JLabel |
| TextField | JTextField |
| PasswordField | JPasswordField |
| Button | JButton |
| Radio Button | JRadioButton |
| Check Box | JCheckBox |
| Button Group | ButtonGroup |
| Drop Down List | JComboBox |
| Text Area | JTextArea |
| Image | ImageIcon |
| Table | JTable |
| Panel | JPanel |
| Frame | JFrame |
| Notification/Message Box | JOptionPane |

For example: to create a GUI, we need to write a class that inherits the java library class JFrame.

## Customizing Color and Font

Up until now, we have discussed about the components of a GUI. By default, they do not have any color. To make a GUI more attractive and eye catching color and font of GUI components can be customized/changed by using the following library classes:

- Color
- Font

Both of these classes are in java.awt package. These classes have constructors to create color and font according to user preference. These user defined colors and fonts can be used for GUI components by calling library methods of their respective classes. The following table summarizes some of the necessary constructors and methods of the Color and the Font class:

| Constructors and Methods | Description |
|---|---|
| Color(int r, int g, int b) | Creates a color having the RGB value as per the parameters |
| setBackground(Color c) | Changes the background color as per the Color object passed in the parameter |
| setForeground(Color c) | Changes the font color as per the Color object passed in the parameter |
| Font(String name, int style, int size) | Creates a font having the name, style and size as per the parameter values |
| setFont(Font f) | Changes the font as per the Font object passed in the parameter |

## Different EventListeners

Now that we know about the things required to design a GUI, we need to know how we can make a GUI work. All the GUI components can trigger events if they are associated with an EventListener for respective events. These event listeners invoke respective methods whenever an event occurs for a component. The following four are some of the most frequently used event listeners:

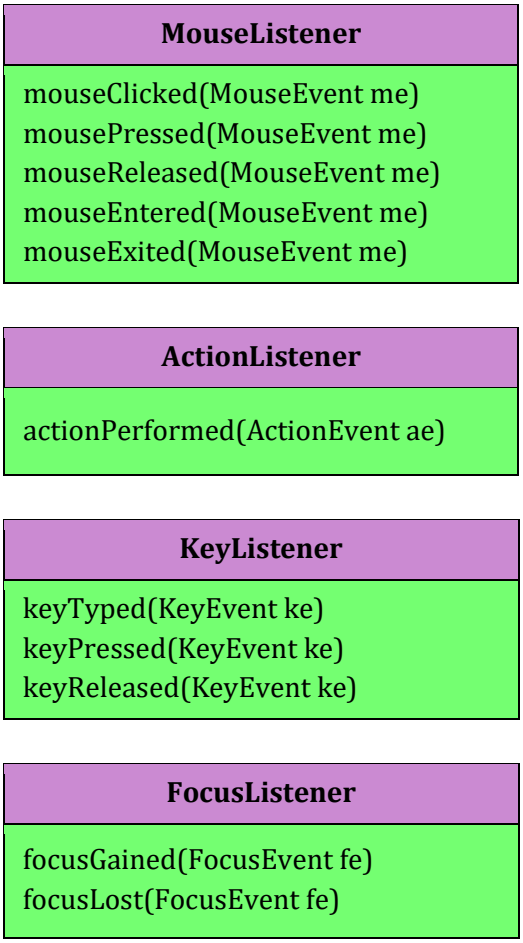
- ActonListener
- MouseListener
- KeyListener
- FocusListener

All of these listeners are java library interfaces. They are inside the java.awt.event package. Each type of listener handles their own type of event.

| EventListener | Event |
|---|---|
| ActionListener | ActionEvent |
| MouseListener | MouseEvent |
| KeyListener | KeyEvent |
| FocusListener | FocusEvent |

A frame that wants to perform some operation or task whenever any event occurs, needs to implement the respective EventListeners. As these event listeners are interfaces, our class has to override the all the abstract methods of the implemented interfaces.

The followings are the abstract methods that these interfaces have:




## Steps to Design a GUI

As stated earlier, we need to write a class to develop a GUI. The object of our class will be the GUI. It will be easier to develop a GUI, if we do the followings first:

- Draw a sketch of the GUI in a paper.
- Make a list of the components of the GUI.
- Make a list of color and fonts that the components will have (optional).
- Make a list of events that might happen in the GUI.

Now, we can write our class. The followings steps should be followed while writing the class:

Step 0: Import necessary library classes.

Step 1: Our class needs to extend the JFrame class and implement necessary listeners.

Step 2: The components used in the GUI will be the attributes of our class.

Step 3: The colors and fonts used in the GUI should also be in the attributes (optional).

Step 4: Write a constructor for the class. The body of the constructor can be divided into several parts:

(a) Initialize the Frame Header, the size of the frame.

(b) Create the panel object and set its layout as null.

(c) Initialize custom Color and Font (Optional).

(d) For each of the components (except for ImageIcon, JTable, ButtonGroup and JComboBox) follow the pattern of "Three Statements".

[1] Create an object for the component.

[2] Initialize its size and position.

[3] Add the component in the panel.

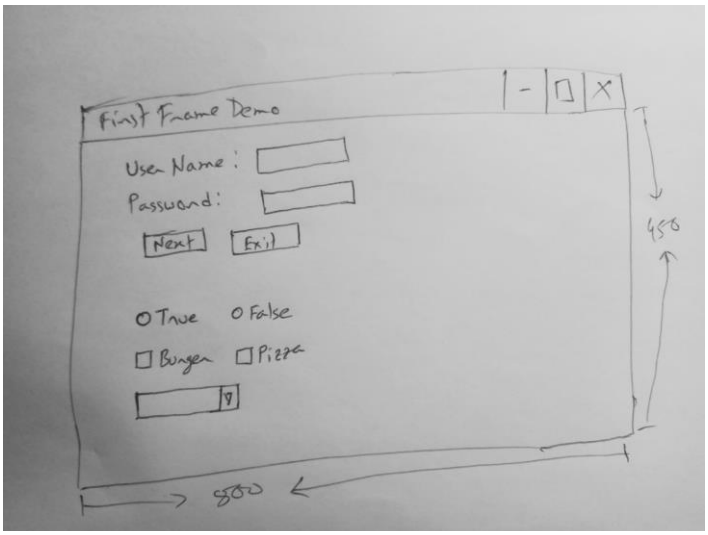
(e) Change color and font for the components and add necessary EventListeners (optional).

(f) Add the panel in the frame.

Step 5: Override the abstract methods of the implemented event listener interfaces.
Step 6: Write a main method in a separate class to display the GUI.

## Designing a GUI

Let us assume that we going to develop the following GUI:




Now, let us make a list of the components for this GUI. There are 2 Labels, 1 Text Field, 1 Password Field, 2 Buttons, 2 Radio Buttons, 2 Check Boxes and a 1 Drop Down List. The

color of the "**Next**" button is Green and the **"Exit"** button is Red. One of the Label is Blue in color. The font color of both of the buttons is White. The background color of the GUI and the font of the Labels are customized. When we click on the **Next** button, a new GUI will appear with all the values from the components of this GUI and when the **Exit** Button is clicked the program will terminate.

Before starting to code, let us get familiarized with some constructors and methods that we will be using.

| Constructor | Description |
|---|---|
| public JLabel(String text) | It creates an object of JLabel. The String text passed in the parameter is the text of the label. |
| public JTextField( ) | It creates an object of JTextField. |
| public JPasswordField( ) | It creates an object of JPasswordField. |
| public JButton(String text) | It creates an object of JButton. The String text passed in the parameter is the text of the Button. |
| public JRadioButton(String text) | It creates an object of JRadioButton. The String text passed in the parameter is the text of the RadioButton. |
| public JCheckBox(String text) | It creates an object of JCheckBox. The String text passed in the parameter is the text of the CheckBox. |
| public JComboBox(String [ ]items) | It creates an object of JComboBox. The value of each index of the String array items passed in the parameter, represents each value of the Drop Down List. |
| public JPanel( ) | It creates an object of JPanel. |
| public JFrame(String text) | It creates an object of JFrame. We will not be using this one directly. |

| Method | Description |
|---|---|
| void setSize(int length, int height) | It is called to initialize the size of the frame. Length and height specifies the length and height of the Frame respectively. |
| void setDefaultCloseOperation(int value) | It is called to make the "**X**" button to close the program. |
| void setLayout(LayoutManager lm) | It is called to specify the layout to follow while adding a component into a panel. |
| void setBounds(int upLeftX, int upLeftY, int length, int height) | It is called to initialize the position and size of a GUI component. The parameter values specifies the x coordinate of the upper left corner of the component, the y coordinate of the upper left corner of the component, the length of the component and the height of the component respectively. |

| | |
|---|---|
| Component add(Component c) | It is called to add a component to a container. |
| String getActionCommand( ) | It is called to get the text of the component for which an action event has occurred. |
| void setText(String text) | It is called to initialize/change the text of a component. |
| String getText( ) | It is called to get the text of any component. |
| boolean isSelected( ) | It is called to check whether a RadioButton or a CheckBox is selected or not. |
| Object getSelectedItem( ) | It is called to get the item which has been selected from a ComboBox. This method does not return the text of the item. To get the text of the item, we need to call another method *toString( )* with this method. |
| void setOpaque(boolean value) | In order to change the background color of a JLabel, we need to call this method and pass the value ***true*** in parameter, in addition with the *setBackground( )* method. |
| void addActionListener(ActionListener al) | This method is called to add an Action Listener to a component. |
| void addMouseListener(MouseListener ml) | This method is called to add a Mouse Listener to a component. |

As stated earlier, we need to write a class to develop a GUI and the object of that class will be the GUI. Let the name of our class be ***FirstFrame***. Let's start our code:

Step 0

```
import java.lang.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

Step 1

```
public class FirstFrame extends JFrame implements ActionListener{
//ActionListener Interface is implemented because we will perform some actions on the two buttons
```

Step 2

```
    private JLabel nameLabel, passLabel;
    private JTextField nameTF;
    private JPasswordField passPF;
    private JButton nextBtn, exitBtn;
    private JRadioButton r1, r2;
    private ButtonGroup bg;
    private JCheckBox c1, c2;
    private JComboBox combo;
    private JPanel panel;
```

Step 3

```
    private Color customColor;
    private Font customFont;
```

| | |
|---|---|
| Step 4 | public FirstFrame( )<br>{ |
| (a) | super(“First Frame Demo”);<br>this.setSize(800, 450);<br>this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); |
| (b) | panel = new JPanel( );<br>panel.setLayout(null); |
| (c) | customColor = new Color(180, 200, 150);<br>customFont = new Font(“Consolas”, Font.BOLD, 16); |
| (d) | //Now, we will follow the three line pattern for each of the components |
| [1]<br>[2]<br>[3] | nameLabel = new JLabel(“User Name: ”);<br>nameLabel.setBounds(50, 50, 100, 30);<br>panel.add(nameLabel); |
| [1]<br>[2]<br>[3] | passLabel = new JLabel(“Password: ”);<br>passLabel.setBounds(50, 100, 100, 30);<br>panel.add(passLabel); |
| [1]<br>[2]<br>[3] | nameTF = new JTextField( );<br>nameTF.setBounds(170, 50, 100, 30);<br>panel.add(nameTF); |
| [1]<br>[2]<br>[3] | passPF = new JPasswordField( );<br>passPF.setBounds(170, 100, 100, 30);<br>panel.add(passPF); |
| [1]<br>[2]<br>[3] | nextBtn = new JButton(“Next”);<br>nextBtn.setBounds(90, 170, 80, 30);<br>panel.add(nextBtn); |
| [1]<br>[2]<br>[3] | exitBtn = new JButton(“Exit”);<br>exitBtn.setBounds(180, 170, 80, 30);<br>panel.add(exitBtn); |
| [1]<br>[2]<br>[3] | r1 = new JRadioButton(“True”);<br>r1.setBounds(250, 250, 80, 30);<br>panel.add(r1); |
| [1]<br>[2]<br>[3] | r2 = new JRadioButton(“False”);<br>r2.setBounds(350, 250, 80, 30);<br>panel.add(r2); |
| [1]<br>[2]<br>[3] | c1 = new JCheckBox(“Burger”);<br>c1.setBounds(250, 300, 80, 30);<br>panel.add(c1); |
| [1]<br>[2]<br>[3] | c2 = new JCheckBox(“Pizza”);<br>c2.setBounds(350, 300, 80, 30);<br>panel.add(c2); |

```
[0]     //There is one additional line for JComboBox
        String items[ ] = new String [ ]{"C", "C++", "Java", "Python"};
[1]     combo = new JComboBox(items);
[2]     combo.setBounds(250, 350, 100, 30);
[3]     panel.add(combo);

        //As we want only one RadioButton to be selectable
[1]     bg = new ButtonGroup( );
[2]     bg.add(r1);
[3]     bg.add(r2);

        nextBtn.setBackground(Color.GREEN);
        exitBtn.setBackground(Color.RED);

        nextBtn.setForeground(Color.WHITE);
        exitBtn.setForeground(Color.WHITE);

(e)     panel.setBackground(customColor);
        nameLabel.setFont(customFont);
        passLabel.setFont(customFont);

        nextBtn.addActionListener(this);
        exitBtn.addActionListener(this);

(f)     this.add(panel);
    }
    public void actionPerformed(ActionEvent ae)
    {
        String command = ae.getActionCommand( );

        if(command.equals(nextBtn.getText( )))
        {
            //to display a notification/message box
            JOptionPane.showMessageDialog(this, "This is a Message Box");
        }
        else if(command.equals(exitBtn.getText( )))
Step    {
5           //to close the application
            System.exit(0);
        }
        else
        {
            //Write necessary codes
        }

    }
}
```

| | |
|---|---|
| Step 6 | public class FrameDemo<br>{<br>public static void main(String args[ ])<br>{<br>First Frame ff = new FirstFarme( );<br>ff.setVisible(true);<br>}<br>} |

## Practice

- ✓ Design a GUI with a TextField and a Button. Whenever you click on the button, the text inside the TextField is displayed in a message box.