

واجهة المستخدم الرسومية

GRAPHICAL USER INTERFACE (GUI)

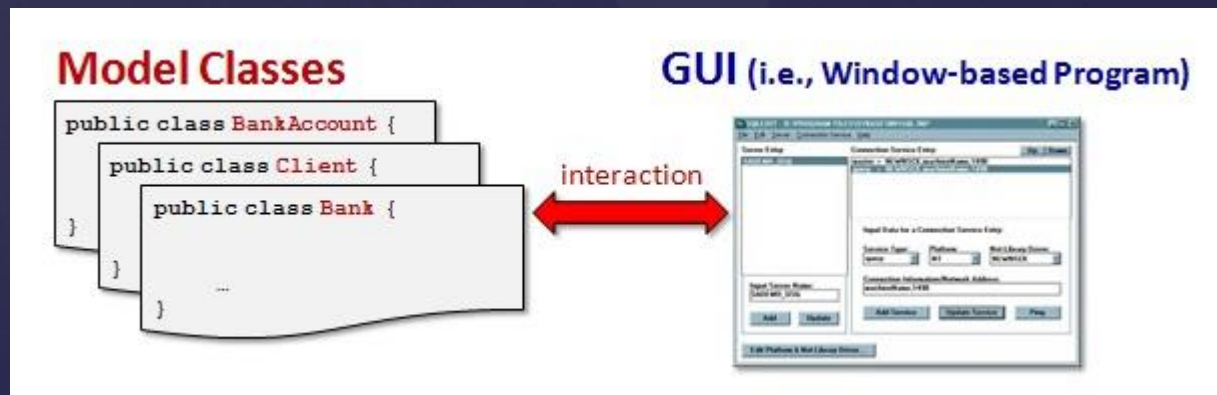
واجهة المستخدم الرسومية

GRAPHICAL USER INTERFACE (GUI)

يعتبر بناء واجهات المستخدم الرسومية من الاجزاء المهمة عند تطوير التطبيقات ، لما لها من أهمية في تيسير استخدام التطبيقات والتفاعل معها.

مثال:

التعامل مع الحساب المصرفي يتم من خلال الواجهة الرسومية والتي تقوم بأستقبال المدخلات وعرض النتائج من دون التدخل في طريقة المعالجة التي تمت للحصول على النتائج.



واجهة المستخدم الرسومية

GRAPHICAL USER INTERFACE (GUI)

يعتبر بناء واجهات المستخدم الرسومية من الاجزاء المهمة عند تطوير التطبيقات ، لما لها من أهمية في تيسير استخدام التطبيقات والتفاعل معها.

تطوير الواجهة الرسومية يتطلب منا التعامل مع ثلاث جوانب رئيسية:

- UI element

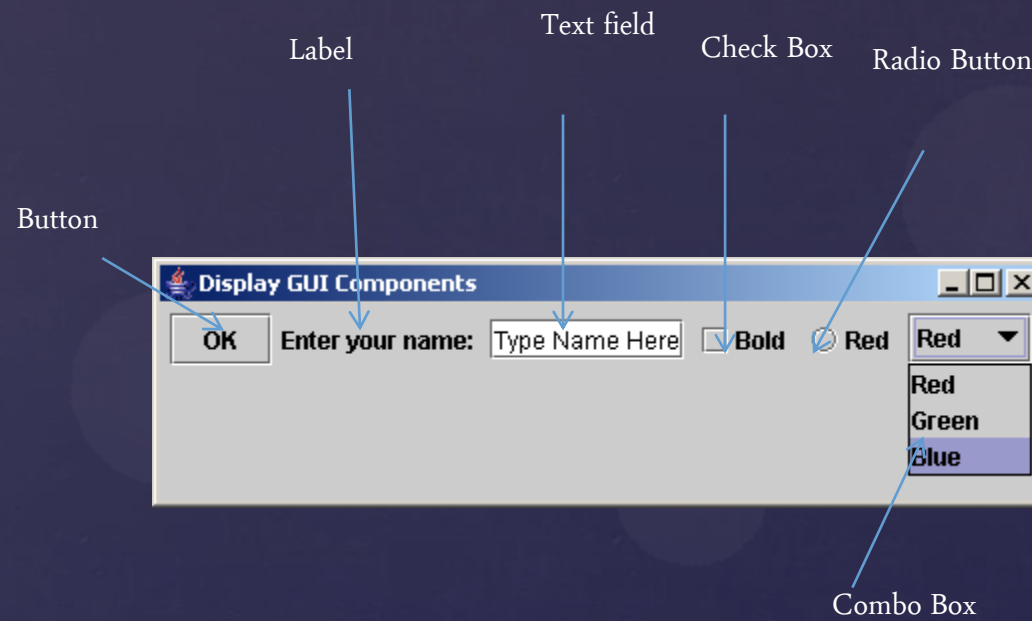
وهي العناصر الرسومية التي يتفاعل معها المستخدم مثل Button, CheckBox

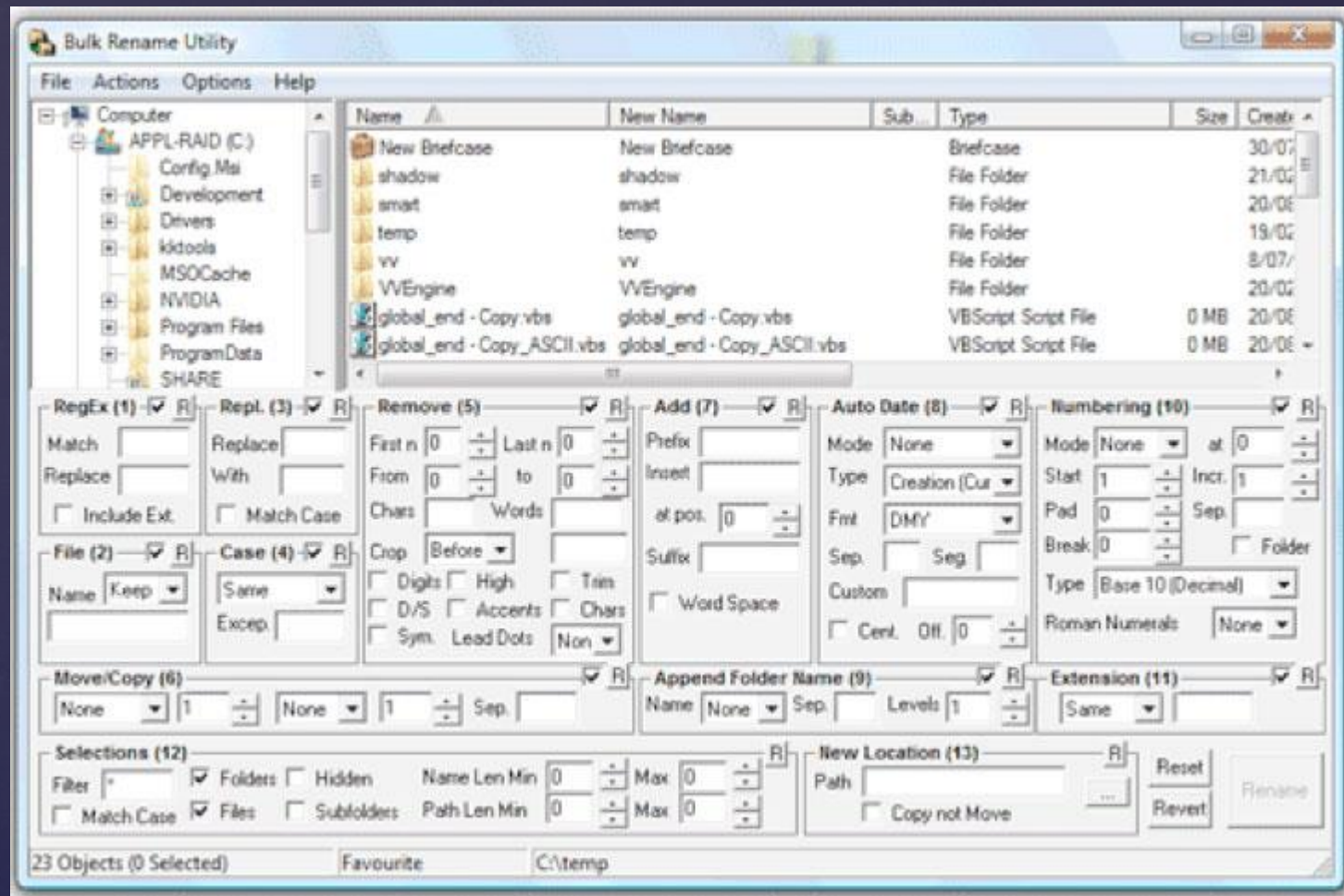
- Layouts

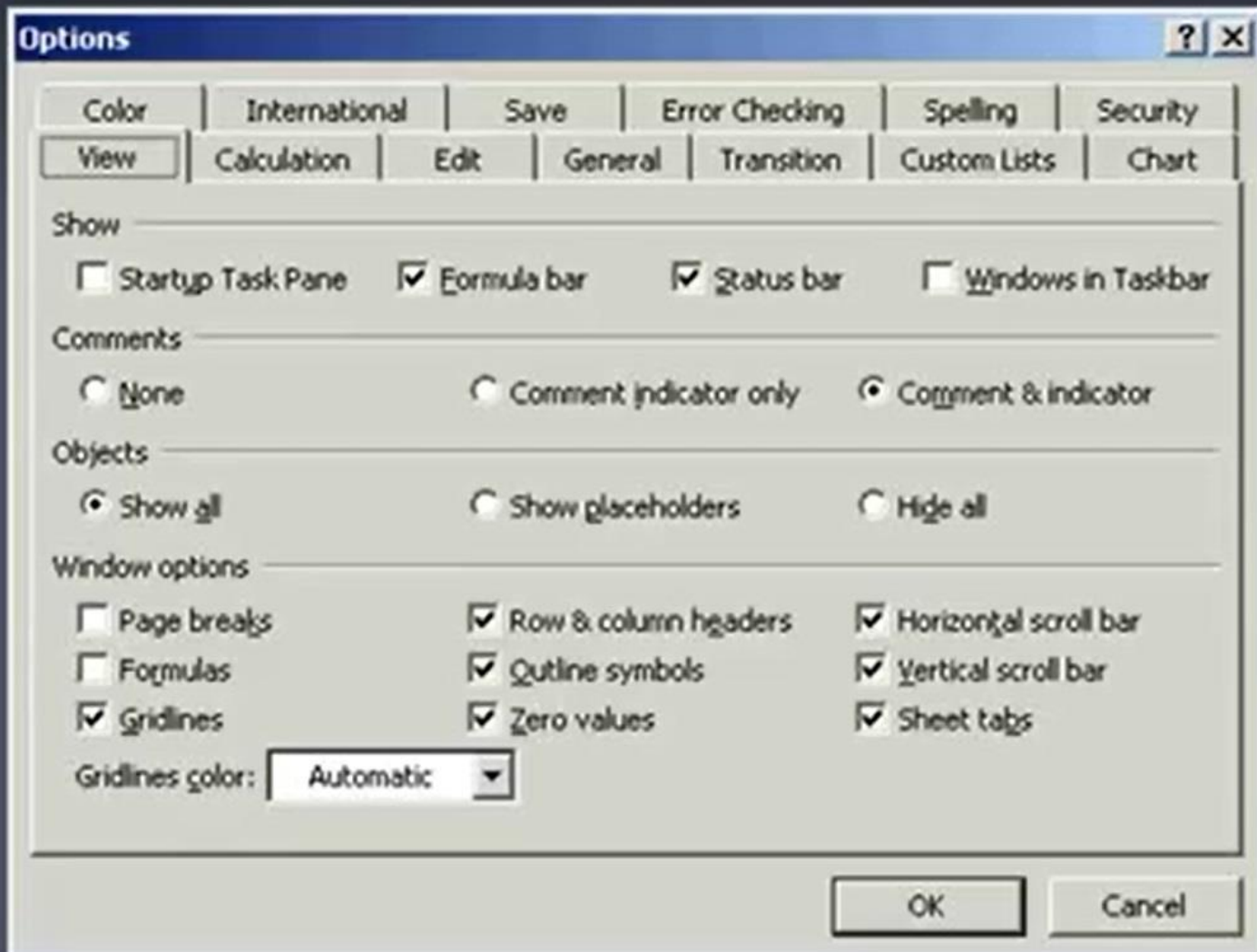
وهي عملية تنسيق وترتيب العناصر الرسومية.

- Events

وهي الاحداث الناشئة عن تفاعل المستخدم مع العناصر الرسومية مثل « حدث الضغط على زر الفأرة »



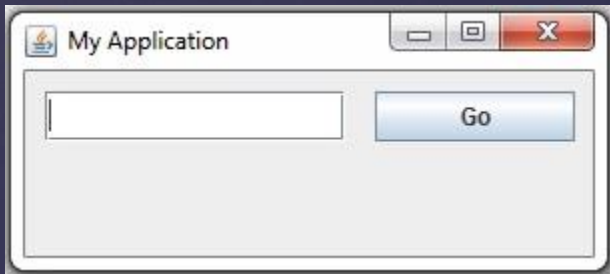




المكونات الرسومية

تتكون واجهة المستخدم الرسومية من العديد من المكونات components وهي عبارة عن كائنات objects يستطيع المستخدم التعامل معها من خلال لوحة المفاتيح أو الفأرة. الجدول التالي يبين بعض المكونات الرسومية.

الرمز	الوصف	أسم المكون الرسومي
	هي نقطة البداية في الواجهة الرسومية والتي يتم فيها وضع المكونات الرسومية	JFrame
label	مكان يوضع فيه نص أو صورة ولا يمكن تغييره أو الكتابة عليه.	JLabel
	يستخدم لآستقبال المدخلات من المستخدم وطباعة الناتج فيه.	TextField
	يستخدم في توجيه التطبيق لتنفيذ عمليات معينة عند الضغط عليه.	JButton
	شكل رسومي يتغير في حالة يكون قد تم اختياره	JCheckBox
	قائمة من العناصر يمكن للمستخدم الاختيار منها.	JcomboBox
	قائمة من العناصر تستخدم للاختيار منها	JList
	عبارة عن مكان يستخدم لاحتواء مجموعة من العناصر الرسومية	JPanel

البرنامج التالي يقوم بإنشاء **JFrame** بها **TextField** و **Button**.

```
import javax.swing.JFrame;
import javax.swing.JTextField;

public class MyApplication extends JFrame {

    public MyApplication(String title) {
        super(title); // Set title of window
        getContentPane().setLayout(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE); // allow window to close
        setSize(300, 130);
        // Add the text field
        JTextField newItemField = new JTextField();
        newItemField.setLocation(10,10);
        newItemField.setSize(150,25);
        getContentPane().add(newItemField);

        // Add the ADD button
        JButton addButton = new JButton("Go");
        addButton.setLocation(175, 10);
        addButton.setSize(100,25);
        getContentPane().add(addButton); // Set size of window
    }

    public static void main(String[] args) {
        MyApplication frame;

        frame = new MyApplication("My Application"); // Create window
        frame.setVisible(true); // Show window
    }
}
```

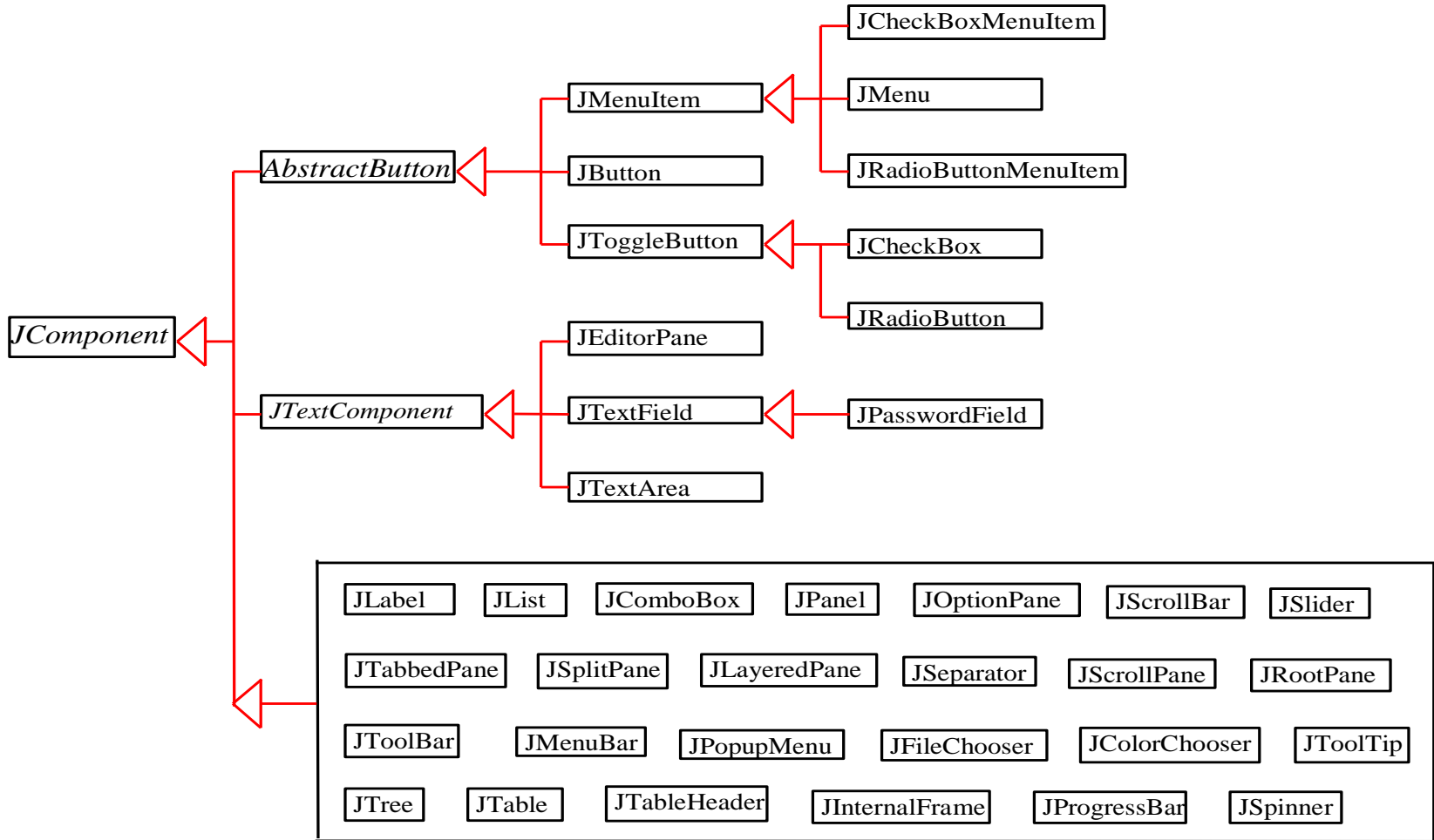

Swing Package

المكونات الرسومية الموجودة في الجدول السابق موجودة داخل الحزمة الرسومية `Javax.swing` وهي قد تم كتابتها كلياً باستخدام لغة جافا لذلك هي تسمى Pure Java Components وهو ما يعني احتفاظها بشكلها حتى لو اختلف نظام التشغيل.

AWT Package

المكونات الرسومية الموجودة داخل الحزمة الرسومية `Java.awt` مرتبطة مباشرة مع الامكانيات الرسومية للجهاز المستخدمة فيه وهي تظهر تبعاً لنظام التشغيل المستخدم في الجهاز.

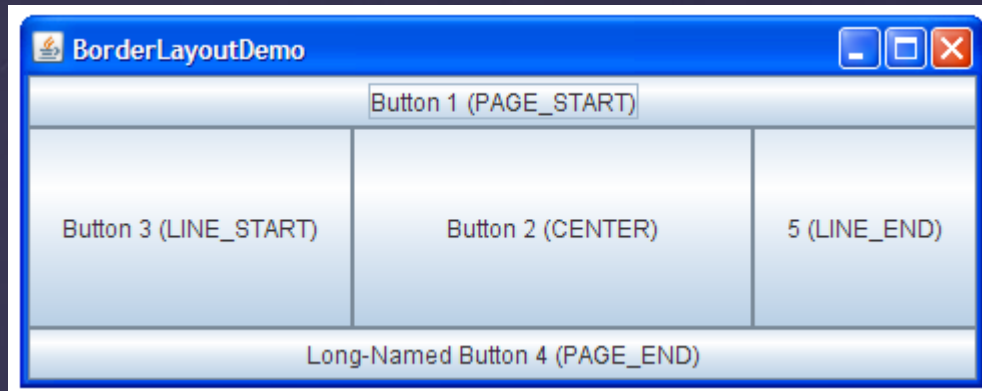
العناصر الرسومية لحزمة SWING



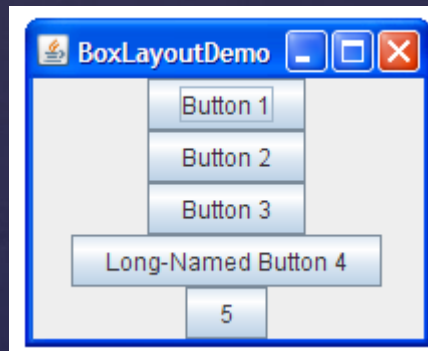
مدير عرض العناصر الرسومية Layout Manager

يستخدم Layout Manager في تنسيق وترتيب العناصر الرسومية عند تكوين الواجهة الرسومية وتوجد عدة أنواع منه أهمها:

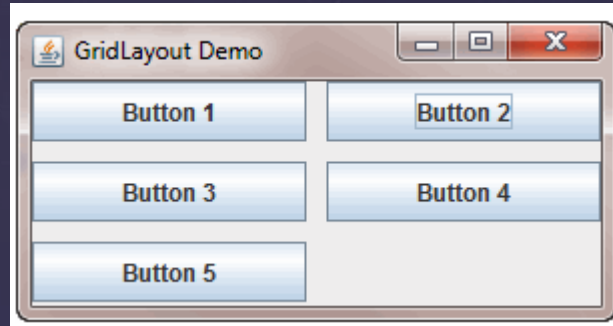
الوصف	Layout Manager
هو مدير العرض التلقائي لكل من <code>Java.awt.Panel</code> , <code>Javax.swing.JPanel</code> ويستخدم في عرض العناصر الرسومية بشكل متسلسل حسب ترتيب أدراجها.	FlowLayout
هو مدير العرض التلقائي <code>Javax.swing.JFrame</code> ويقوم بترتيب العناصر في خمس مناطق : شمال ، جنوب ، شرق ، غرب ، وسط.	BorderLayout
يقوم بترتيب العناصر الرسومية في سطور وأعمدة.	GridLayout
يقوم بترتيب العناصر الرسومية بشكل أفقي أو عمودي	BoxLayout



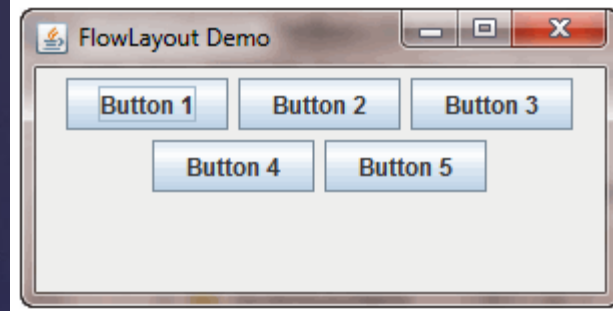
BorderLayout



BoxLayout



GridLayout



FlowLayout

معالجة الاحداث - Event Handling

Event

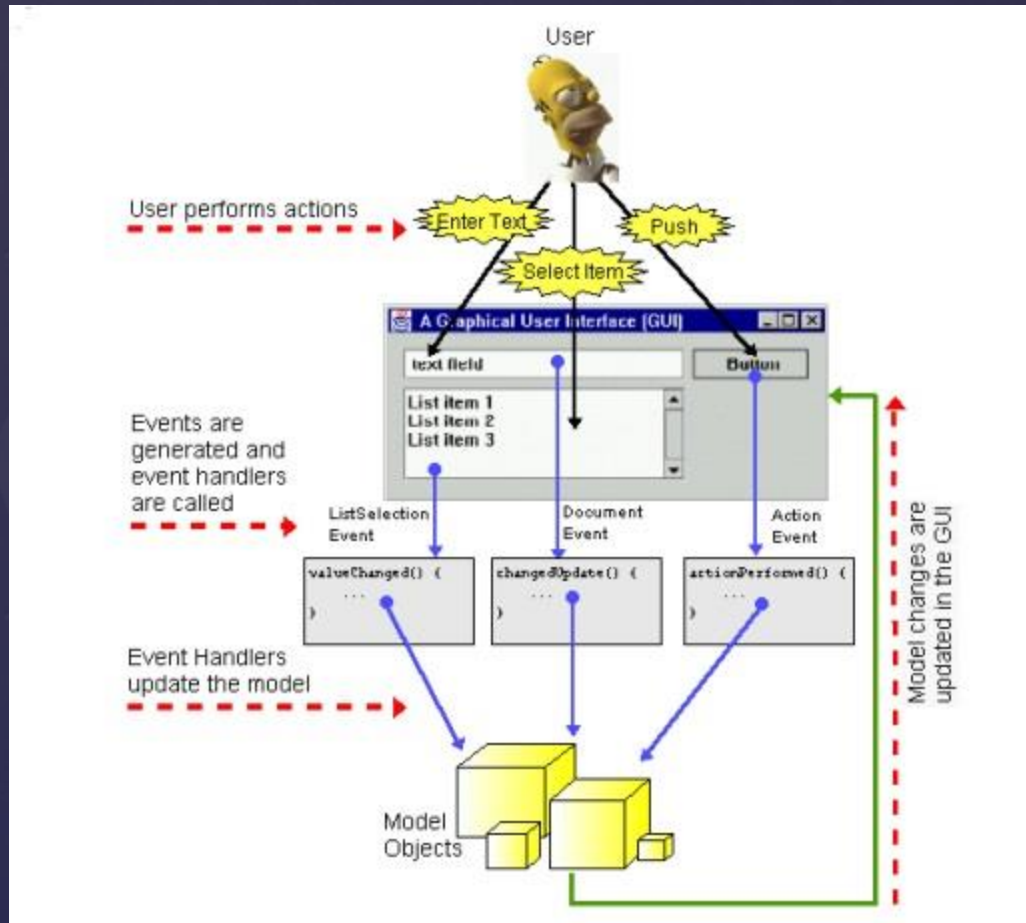
تنشأ Events نتيجة لتفاعل المستخدم مع الواجهة الرسومية مثل :

- النقر على زر.
- تحريك الفأرة .
- الضغط على مفتاح.
- الاختيار من قائمة.

Event Handler

يقصد بمعالجة الاحداث (event handler) بالاجراء الذي يحدد العمليات التي سيتم تنفيذها عند حصول حدث (Event) معين.

Event Handling - معالجة الاحداث



بعض الاحداث الناتجة من تفاعل المستخدم مع العنصر الرسومي

Source Object	User Action	Event Type Generated
JButton	Click a button	ActionEvent
JCheckBox	Click a check box	ItemEvent, ActionEvent
JRadioButton	Click a radio button	ItemEvent, ActionEvent
JTextField	Press return on a text field	ActionEvent
JComboBox	Select a new item	ItemEvent, ActionEvent
Window	Window opened, closed, etc.	WindowEvent
Component	Mouse pressed, released, etc.	MouseEvent
Component	Key released, pressed, etc.	KeyEvent

الجدول التالي يبين بعض الدوال الخاصة بالتعامل مع بعض الاحداث الناتجة من تفاعل المستخدم مع العنصر الرسومي

Event Class	Listener Interface	Listener Methods (Handlers)
ActionEvent	ActionListener	actionPerformed(ActionEvent)
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
WindowEvent	WindowListener	windowClosing(WindowEvent)
MouseEvent	MouseListener	mousePressed(MouseEvent)
KeyEvent	KeyListener	keyPressed(KeyEvent)

الجدول التالي يوضح بعض الـ interface الخاص بالتعامل مع كل منها:

Event	Interface to Implement
ActionEvent - generated when button pressed, menu item selected, enter key pressed in a text field or from a timer event	<pre>public interface ActionListener { public void actionPerformed(ActionEvent e); }</pre>
DocumentEvent - generated when changes have been made to a text document such as insertion, removal in an editor	<pre>public interface DocumentListener { public void changedUpdate(DocumentEvent e); public void insertUpdate(DocumentEvent e); public void removeUpdate(DocumentEvent e); }</pre>
ListSelectionEvent - generated when selecting (i.e., click or double click) a list item	<pre>public interface ListSelectionListener { public void valueChanged(ListSelectionEvent e); }</pre>
WindowEvent - generated when open/close, activate/deactivate, iconify/deiconify a window	<pre>public interface WindowListener { public void windowOpened(WindowEvent e); public void windowClosed(WindowEvent e); public void windowClosing(WindowEvent e); public void windowActivated(WindowEvent e); public void windowDeactivated(WindowEvent e); public void windowIconified(WindowEvent e); public void windowDeiconified(WindowEvent e); }</pre>
KeyEvent - generated when pressing and/or releasing a key while within a component	<pre>public interface KeyListener { public void keyPressed(KeyEvent e); public void keyReleased(KeyEvent e); public void keyTyped(KeyEvent e); }</pre>
MouseEvent - generated when pressing/releasing/clicking a mouse button, moving a mouse onto or away from a component	<pre>public interface MouseListener { public void mouseClicked(MouseEvent e); public void mouseEntered(MouseEvent e); public void mouseExited(MouseEvent e); public void mousePressed(MouseEvent e); public void mouseReleased(MouseEvent e); }</pre>
MouseEvent - generated when moving mouse within a component while button is up or down	<pre>public interface MouseMotionListener { public void mouseDragged(MouseEvent e); public void mouseMoved(MouseEvent e); }</pre>

معالجة الاحداث - Event Handling

بعد كتابة event handler نحتاج إلى تسجيله حتى يتم أشعاره عند حدوث الحدث ليقوم بمعالجته وتتم عملية التسجيل كالتالي:
نقوم بإضافة listener إلى العنصر الرسومي باستخدام الدالة addXXXListener حيث XXX تعتمد على event المراد التعامل معه.

مثال:

```
aButton.addActionListener(anActionListener);  
aJPanel.addMouseListener(aMouseListener);  
aJFrame.addWindowListener(aWindowListener);
```

البرنامج التالي يبين طريقة التعامل مع الحدث الناشيء عن الضغط على Button .

```
import java.awt.event.*; // Need this for ActionEvent and ActionListener
import javax.swing.*;    // Need this for JFrame and JButton

public class SimpleEventTest extends JFrame implements ActionListener {
    public SimpleEventTest(String name) {
        super(name);

        getContentPane().setLayout(null);

        JButton aButton = new JButton("Press Me");
        aButton.setLocation(20,10);
        aButton.setSize(100, 30);
        getContentPane().add(aButton);

        // Plugin button event handler using THIS class as the listener
        // (i.e., tell JAVA to call the actionPerformed() in THIS class)
        aButton.addActionListener(this);

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(250, 90);
    }
    // Must write this method now since SimpleEventTest
    // implements the ActionListener interface
    public void actionPerformed(ActionEvent e) {
        System.out.println("I have been pressed");
    }

    public static void main(String[] args) {
        JFrame frame = new SimpleEventTest("Making a Listener");
        frame.setVisible(true);
    }
}
```



هي عبارة عن class يتم فيها عمل g implementation ل interface باستخدام مجموعة من dummy methods. ولكل listener interface به أكثر من method توجد adapter class خاصة به.

MouseListener => MouseAdapter

MouseMotionListener => MouseMotionAdapter

DocumentListener => DocumentAdapter

WindowListener => WindowAdapter

مثال:

```
public abstract class WindowAdapter implements WindowListener {  
    public void windowOpened(WindowEvent e) {};  
    public void windowClosed(WindowEvent e) {};  
    public void windowClosing(WindowEvent e) {};  
    public void windowActivated(WindowEvent e) {};  
    public void windowDeactivated(WindowEvent e) {};  
    public void windowIconified(WindowEvent e) {};  
    public void windowDeiconified(WindowEvent e) {};  
}
```

البرنامج التالي يبين طريقة استخدام WindowAdapter .

```
import java.awt.event.*;    // Need this for WindowEvent and WindowListener
import javax.swing.*;      // Need this for JFrame

public class WindowEventTest2 extends JFrame {
    public WindowEventTest2 (String name) {
        super(name);

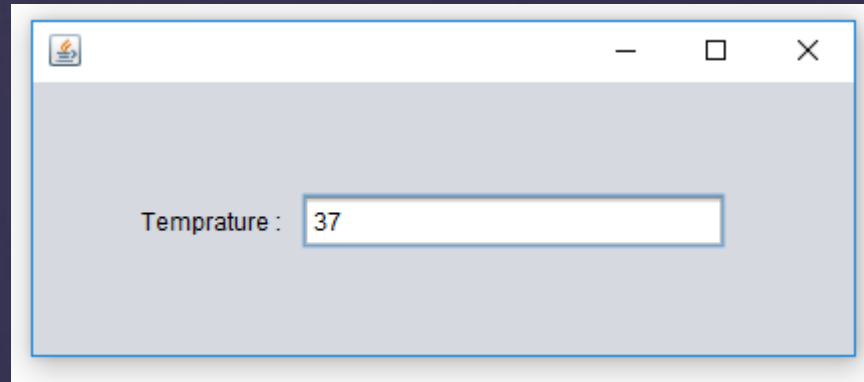
        // Plugin window event handler by creating a separate class
        // that extends WindowAdapter so that only one method needs
        // to be written. This is called an "inner class", which
        // must have default access (e.g., not public nor private).
        class MyWindowHandler extends WindowAdapter {
            public void windowOpened(WindowEvent event) {
                System.out.println("Window has been opened");
            }
        }

        this.addWindowListener(new MyWindowHandler());

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(400, 300);
    }
    public static void main(String[] args) {
        JFrame frame = new WindowEventTest2("Inner Class Example");
        frame.setVisible(true);
    }
}
```

توفر لغة جافا مجموعة من classes لكل نوع من أنواع البيانات الأساسية تقدم مجموعة من الوظائف من أهمها عمليات التحويل بين أنواع البيانات.

Primitive Type	Listener Methods (Handlers)
boolean	Boolean
byte	Byte
char	Char
float	Float
int	Integer
long	Long
short	short
double	double

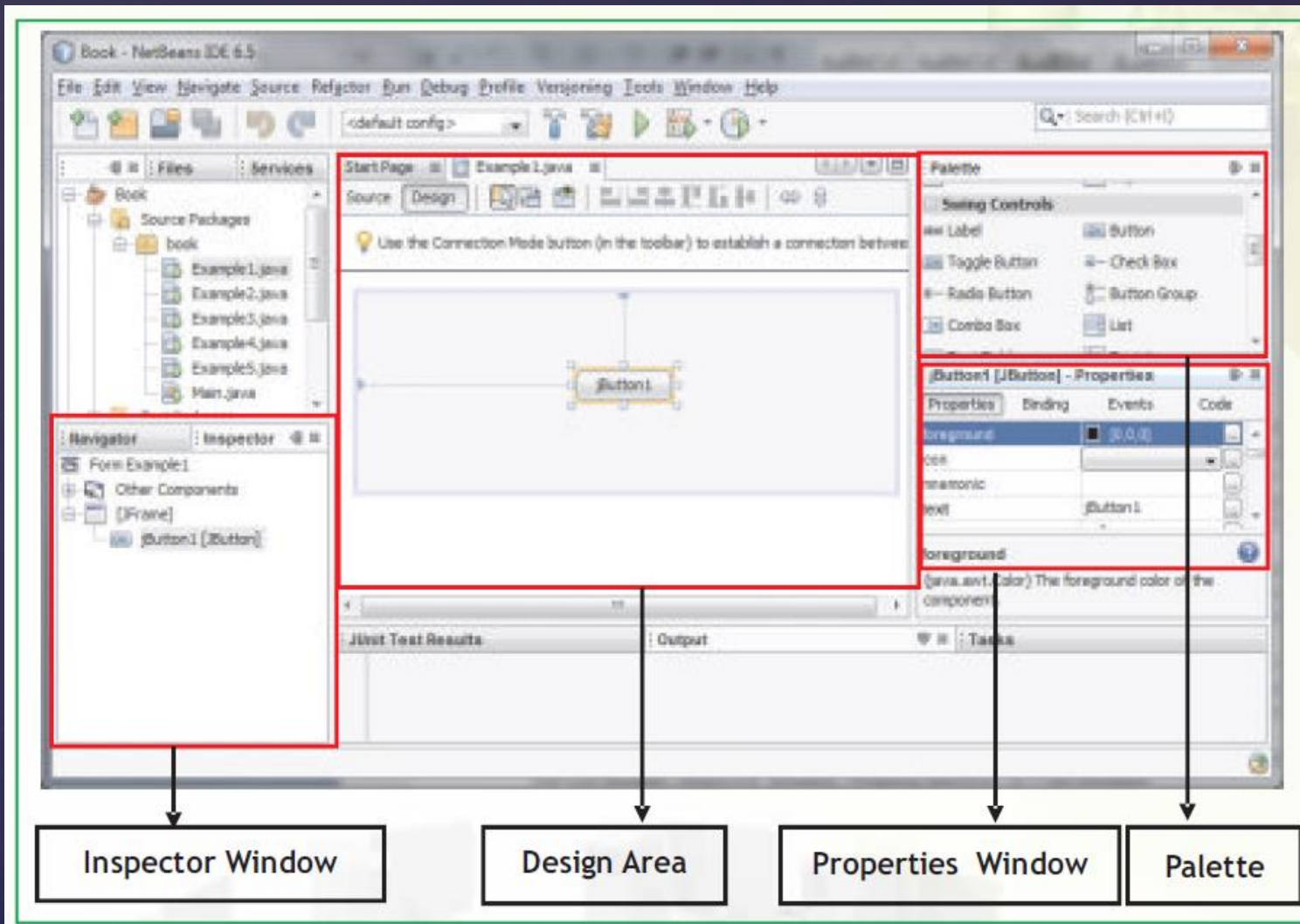


```
double temp = Double.parseDouble(jTextField1.getText());
```

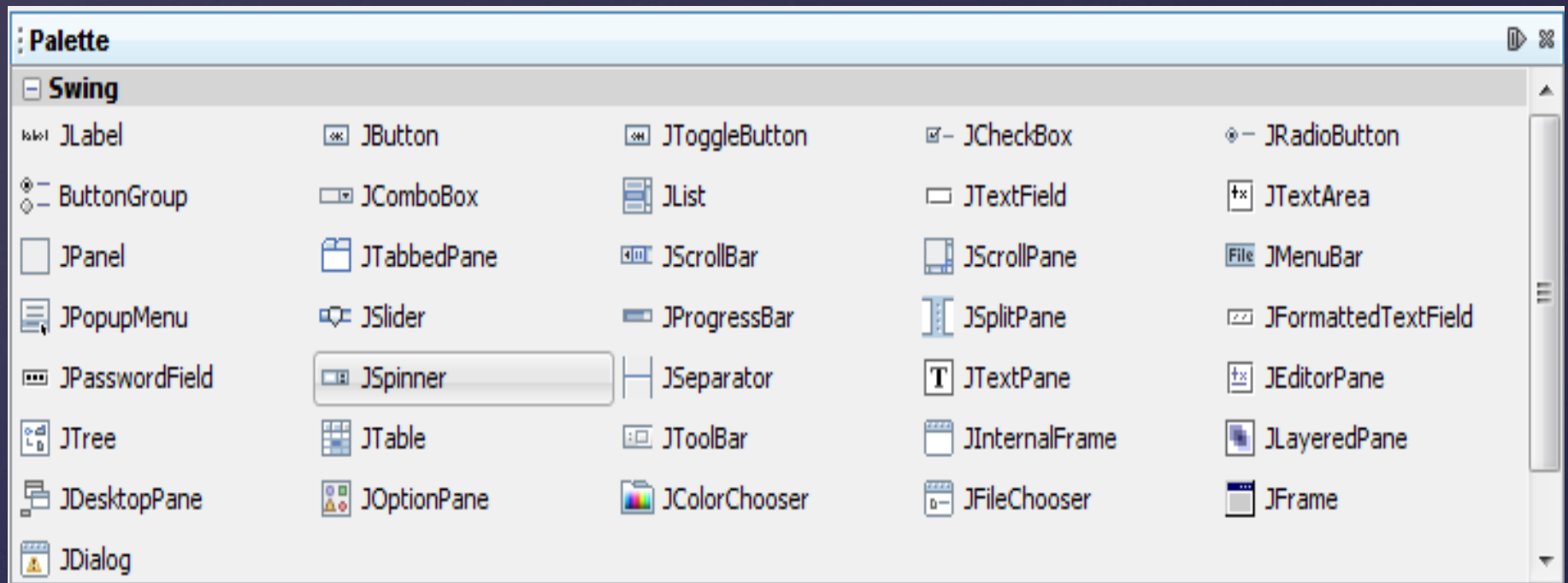
واجهة المستخدم الرسومية بأستخدام NETBEANS

تستخدم Netbeans هذه الادوات في تصميم الواجهة الرسومية:

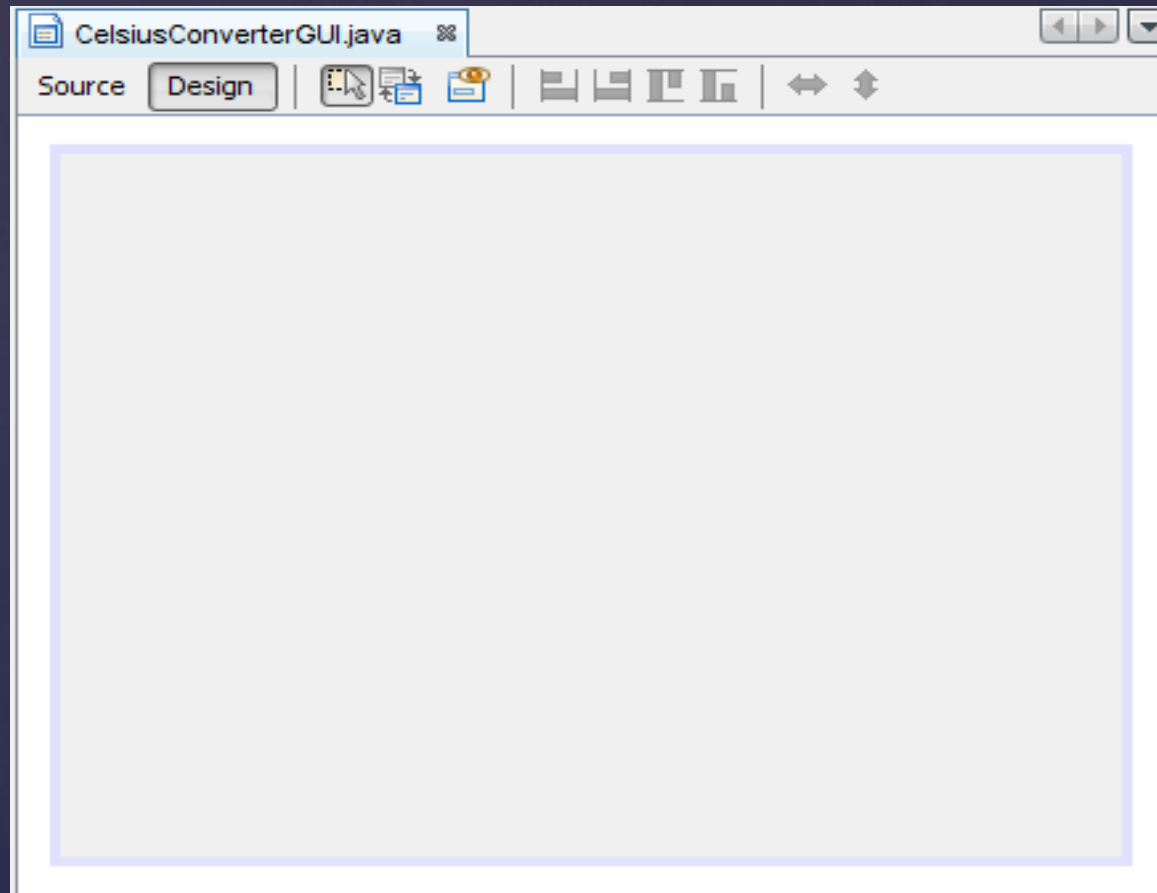
- The Palette
- The Design Area
- The property Editor
- The Inspector



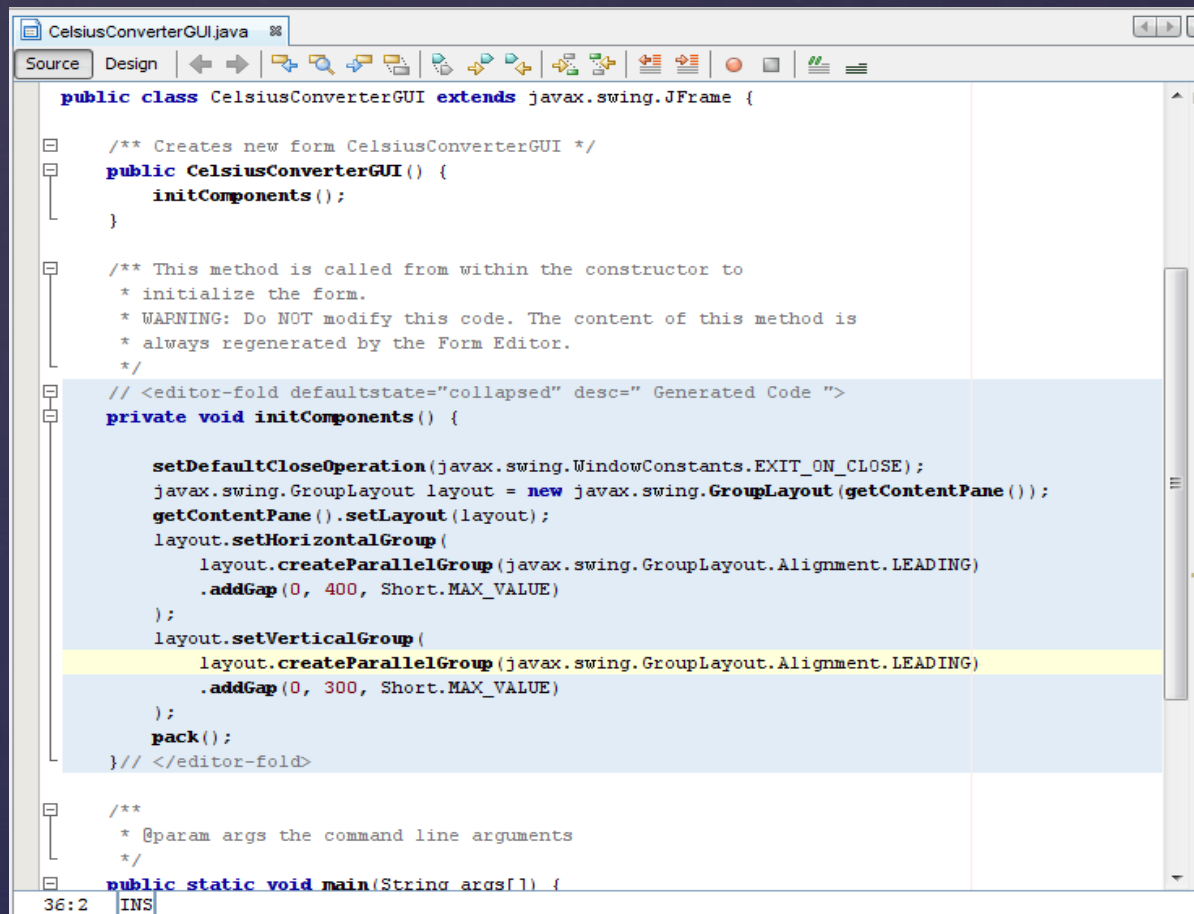
تحتوي على جميع المكونات التي تقدمها الحزمة Swing مثل JLabel, JList, JTextField وغيرها.



هي المكان الذي سيتم فيه تصميم الواجهة الرسومية.



design view



```
public class CelsiusConverterGUI extends javax.swing.JFrame {

    /** Creates new form CelsiusConverterGUI */
    public CelsiusConverterGUI() {
        initComponents();
    }

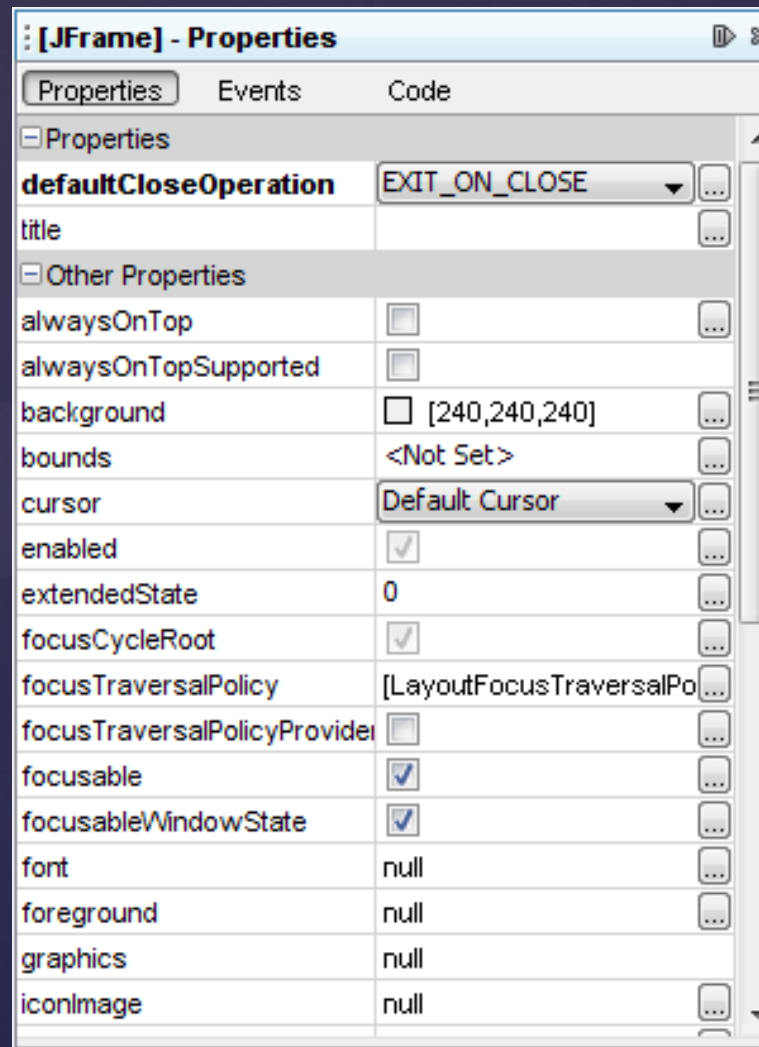
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {

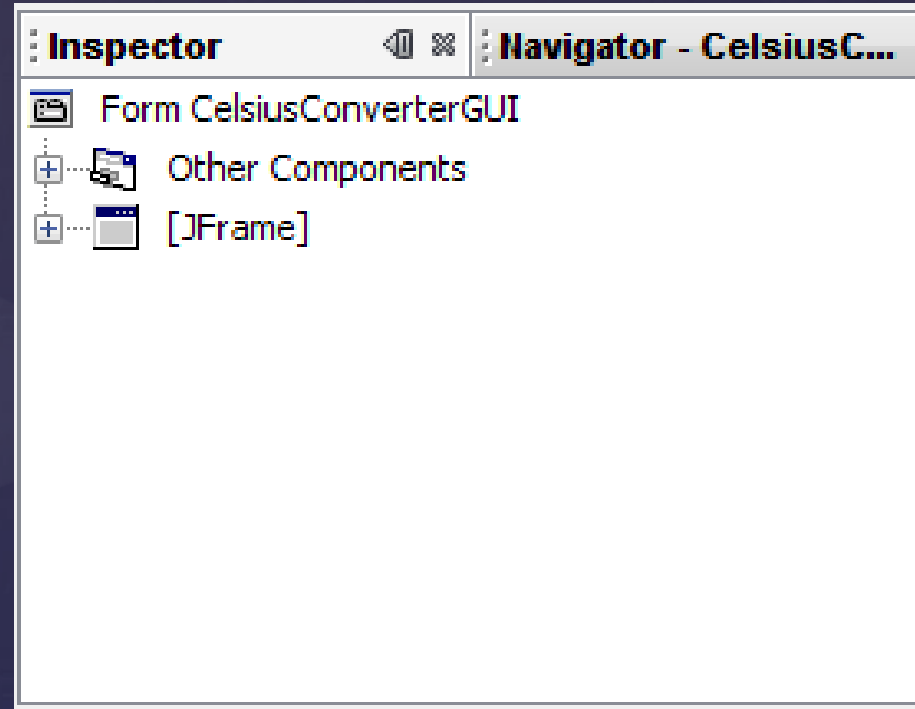
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(0, 400, Short.MAX_VALUE)
                )
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(0, 300, Short.MAX_VALUE)
                )
        );
        pack();
    } // </editor-fold>

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        36:2 INS
```

source view

هي المكان الذي يتم فيه تحديد خصائص كل مكون من مكونات الواجهة الرسومية.

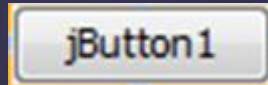




JFram

Property	Description
defaultCloseOperation	Sets action to be performed when the user attempts to close the form.
Title	Sets the text to be displayed in the Title bar of the form window.

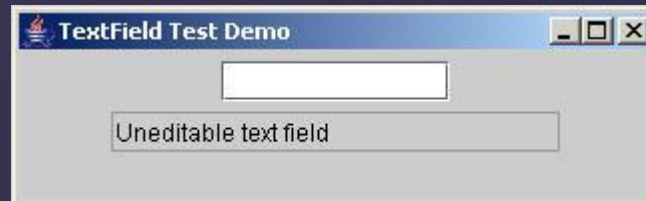
JButton



Property	Description
Background	Sets the background color.
Enabled	Contains enabled state of component - true if enabled else false.
Font	Sets the font.
Foreground	Sets the foreground color.
horizontalalignment	Sets the horizontal alignment of text displayed on the button.
Label	Sets the display text.
Text	Sets the display text

Method	Description
getText()	Retrieves the text typed in jButton. <code>String result=<button-name>.getText () ;</code>
setEnabled	Enables or disables the button. <code><button-name>.setEnabled(boolean b) ;</code>
setText()	Changes the display text at runtime. <code><button-name>.setText(String text) ;</code>
setVisible	Makes the component visible or invisible - true to make the component visible; false to make it invisible. <code><button-name>.setVisible(boolean aFlag) ;</code>

JTextField



Property	Description
Background	Sets the background color.
Border	Sets the type of border that will surround the text field.
editable	If set true user can edit textfield. Default is true.
enabled	Contains enabled state of component- True if enabled else false.
font	Sets the font.
foreground	Sets the foreground color.
horizontalAlignment	Sets the horizontal alignment of text displayed in the textfield.
text	Sets the display text
toolTipText	Sets the text that will appear when cursor moves over the component.

Method	Description
getText()	Retrieves the text in typed in jTextField. String result=<textfield-name>.getText () ;
isEditable()	Returns true if the component is editable else returns false. boolean b=<textfield-name>.isEditable () ;
isEnabled()	Returns true if the component is enabled,else returns false. boolean b =<textfield-name>.isEnabled() ;
setEditable	Sets whether the user can edit the text in the textfield. true if editable else false. <textfield-name>.setEditable (boolean b) ;
setText()	Changes the display text at runtime. <textfield-name>.setText (String t) ;
setVisible()	Makes the component visible or invisible - true to make the component visible; false to make it invisible. <textfield-name>.setVisible (boolean b) ;

JLabel

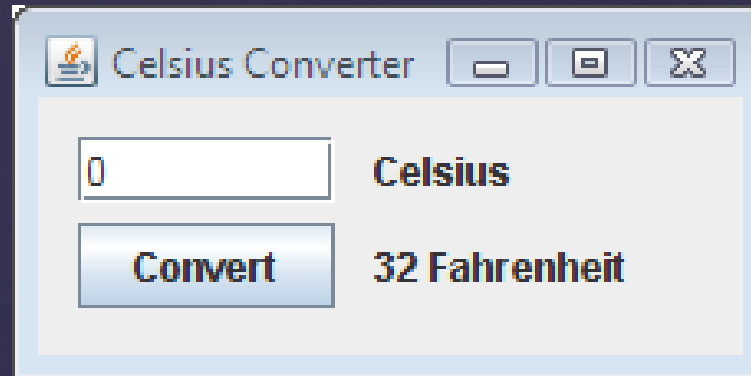
jLabel1

Property	Description
background	Sets the background color.
enabled	Contains enabled state of component- true if enabled else false.
font	Sets the font.
foreground	Sets the foreground color.
horizontalAlignment	Sets the horizontal alignment of text displayed in the component.
text	Sets the display text

Method	Description
getText()	Retrieves the text in typed in jLabel. String result=<label-name>.getText() ;
isEnabled()	Returns true if the component is enabled,else returns false. boolean b=<label-name>.isEnabled() ;
setText()	Changes the display text at runtime. <label-name>.setText(String t) ;
setVisible()	Makes the component visible or invisible - true to make the component visible; false to make it invisible. <label-name>.setVisible(boolean b) ;

مثال:

تصميم واجهه رسومية ستستخدم في عملية تحويل درجة الحرارة من المئوي (Celsius) إلى الفهرنهايت كما هو مبين بالشكل التالي:



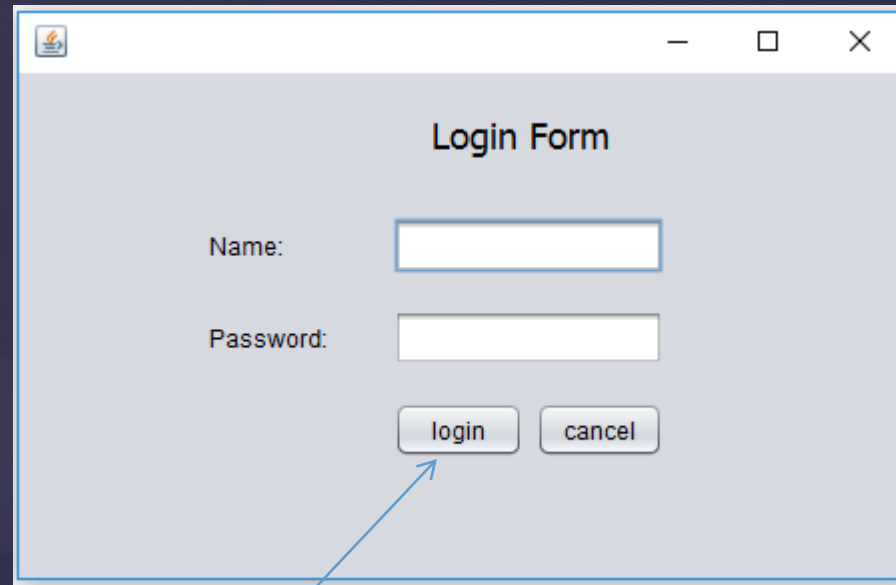
العناصر الرسومية المستخدمة:

- JFarme
- JButton
- JTextField
- JLabel

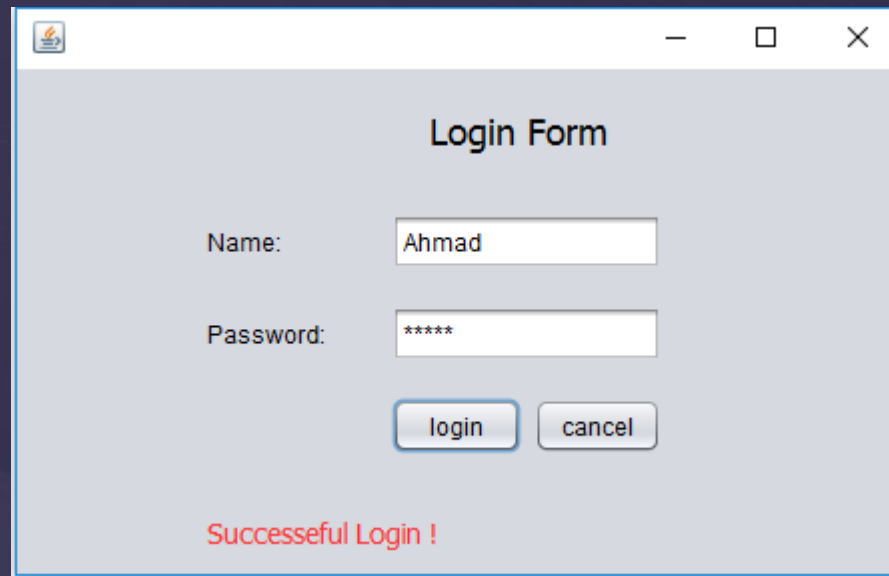
الحدث الناشئ من تفاعل المستخدم مع الواجهة الرسومية:

- الضغط على الزر Convert

تصميم واجهه رسومية تقوم بعمل Login Form كما هو مبين بالشكل التالي:



```
private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {  
  
    char [] myPassword ={'1','2','3','4','5'};  
    if (Arrays.equals (passwordField.getPassword(),myPassword)) {  
        messageLable.setText("Successeful Login !");  
    }else  
    {  
        messageLable.setText("incorrect Password !");  
    }  
}
```



A screenshot of a Java Swing window titled "Login Form". The window has a standard title bar with a small icon on the left and minimize, maximize, and close buttons on the right. The main content area has a light gray background. It contains two text input fields: the first is labeled "Name:" and contains the text "Ahmad"; the second is labeled "Password:" and contains six asterisks "*****". Below the password field are two buttons: "login" and "cancel". At the bottom of the window, the text "Successseful Login !" is displayed in red.

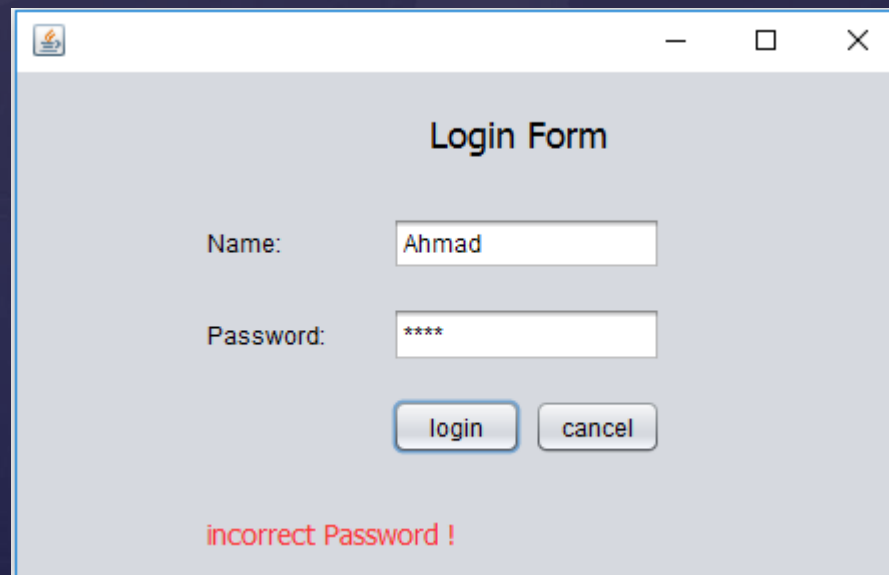
Login Form

Name: Ahmad

Password: *****

login cancel

Successseful Login !



A screenshot of a Java Swing window titled "Login Form", identical in layout to the one above. The "Name:" field contains "Ahmad" and the "Password:" field contains "****". The "login" and "cancel" buttons are present. At the bottom of the window, the text "incorrect Password !" is displayed in red.

Login Form

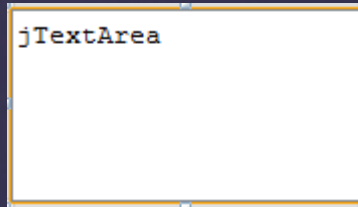
Name: Ahmad

Password: ****

login cancel

incorrect Password !

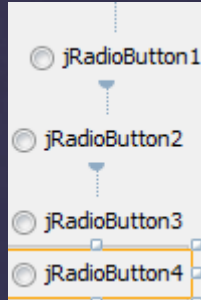
JTextArea



Property	Description
background	Sets the background color.
columns	Sets number of columns preferred for display.
editable	If set true user can edit textfield. Default is true.
enabled	Contains enabled state of component- true if enabled else false.
font	Sets the font.
foreground	Sets the foreground color.
lineWrap	Indicates whether line of text should wrap in case it exceeds allocated width.(Default is false)
rows	Sets number of rows preferred for display.
text	Sets the display text
wrapStyleWord	Sends word to next line in case lineWrap is true and it results in breaking of a word, when lines are wrapped.

Method	Description
append()	Adds data at the end. <code><textarea-name>.append(String str) ;</code>
getText()	Retrieves the text in typed in jTextArea. <code>String str = <textarea-name>.getText() ;</code>
isEditable()	Returns true if the component is editable else returns false. <code>boolean b = <textarea-name>.isEditable() ;</code>
isEnabled()	Returns true if the component is enabled, else returns false. <code>boolean b = <textarea-name>.isEnabled() ;</code>
setText()	Changes the display text at runtime. <code><textarea-name>.setText(String t) ;</code>

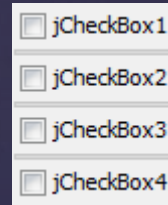
JRadioButton



Property	Description
background	Sets the background color.
buttonGroup	Specifies the name of the group of button to which the JRadioButton belongs.
enabled	Contains enabled state of component -true if enabled else false.
font	Sets the font.
foreground	Sets the foreground color.
label	Sets the display text.
text	Sets the display text.
Selected	Sets the button as selected, if set to true, default is false.

Method	Description
getText()	Retrieves the text displayed by radio button. <code>String str = <radiobutton-name>.getText();</code>
isSelected()	Returns true if the component is checked else returns false. <code>boolean b = <radiobutton-name>.isSelected();</code>
setText()	Changes the display text at runtime. <code><radiobutton-name>.setText(String t);</code>
setSelected()	Checks(true) or unchecks the radio button. <code><radiobutton-name>.setSelected(boolean b);</code>

jCheckBox



Property	Description
background	Sets the background color.
buttonGroup	Specifies the name of the group of button to which the jCheckBox belongs.
font	Sets the font.
foreground	Sets the foreground color.
label	Sets the display text.
text	Sets the display text
selected	Sets the check box as selected if set to true, default is false.

Method	Description
getText()	Retrieves the text typed in <code>String str = <checkbox-name>.getText() ;</code>
isSelected()	Returns true if the component is checked else returns false. <code>boolean b = <checkbox-name>.isSelected() ;</code>
setText()	Changes the display text at runtime. <code><checkbox-name>.setText(String t) ;</code>
setSelected()	Checks(true) or unchecks the checkbox. <code><checkbox-name>.setSelected(boolean b) ;</code>

تصميم واجهه رسومية تقوم بعمل License Agreement Form كما هو مبين بالشكل التالي:

License Agreement

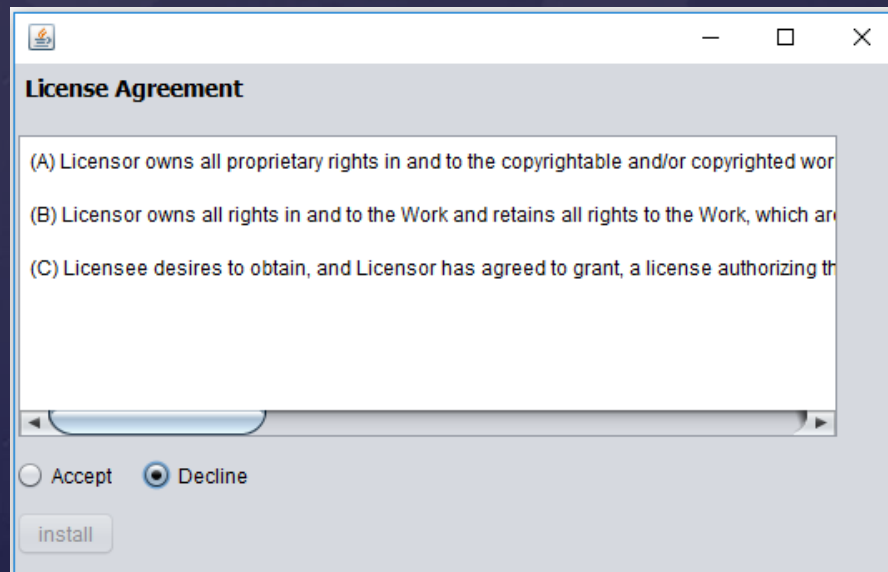
s described in Appendix A, incorporated herein by reference, and hereinafter collectively kn
 t transferred herein, and retains all common law copyrights and all federal copyrights whic
 se of the Work by Licensee in accordance with the terms and conditions of this Agreement.

☐ Accept ☐ Decline

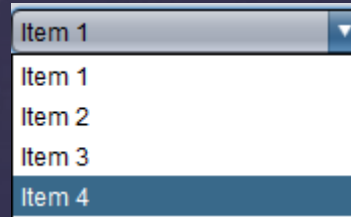
Install

```
private void acceptRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
    installButton.setEnabled(true);
}

private void declineRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
    installButton.setEnabled(false);
}
```



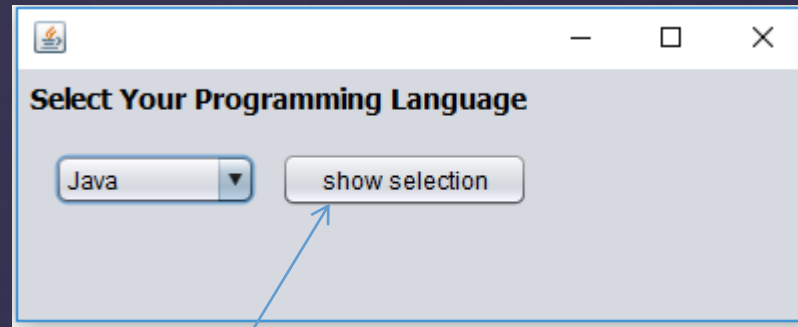
jComboBox



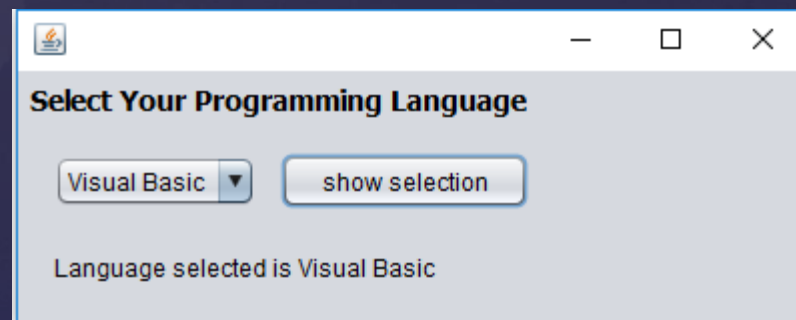
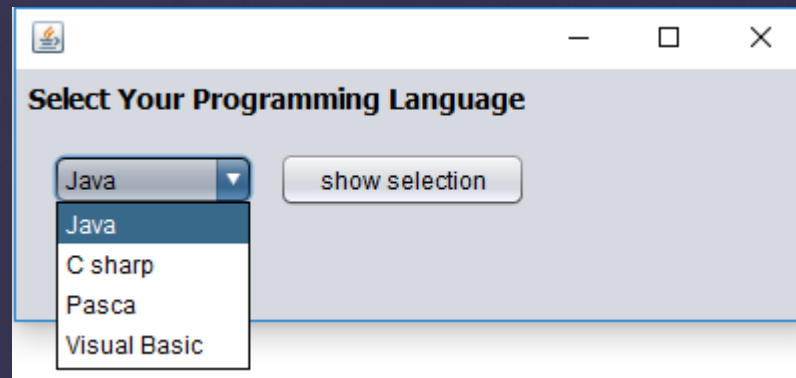
Property	Description
background	Sets the background color.
buttongroup	Specifies the name of the group of button to which the jComboBox belongs.
editable	If set true user can edit ComboBox. Default is true.
enabled	Contains enabled state of component- True if enabled else false.
font	Sets the font.
foreground	Sets the foreground color.
model	Contains the values to be displayed in the combobox.
text	Sets the display text
selectedIndex	Sets the index number of the element which should be selected by default.
selectedItem	Sets the selected item in the combobox. selectedItem and selectedIndex are in synchronization with each other.

Method	Description
getSelectedItem()	Retrieves the selected item. Object result = <code><combobox-name>.getSelectedItem();</code>
getSelectedIndex()	Retrieves the index of the selected item. int result = <code><combobox-name>.getSelectedIndex();</code>
setModel()	Sets the data model that the combo box uses to get its list of elements. <code><combobox-name>.setModel (ComboBoxModel aModel);</code>

تصميم واجهه رسومية تقوم بعمل JComboBox لاستخدامها في اختيار إحدى لغات البرمجة كما هو مبين بالشكل التالي:



```
private void selectionButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    selectionLabel.setText("Language selected is " + selectionComboBox.getSelectedItem().toString());  
}
```

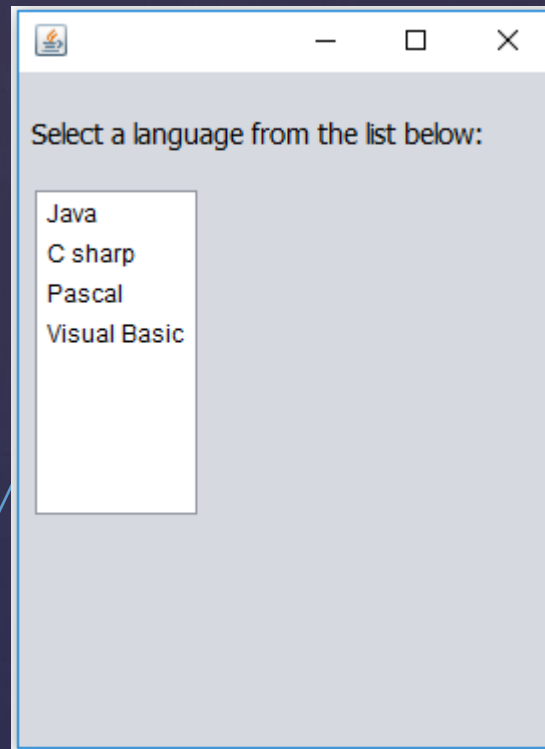
jList

Item 1
Item 2
Item 3
Item 4
Item 5

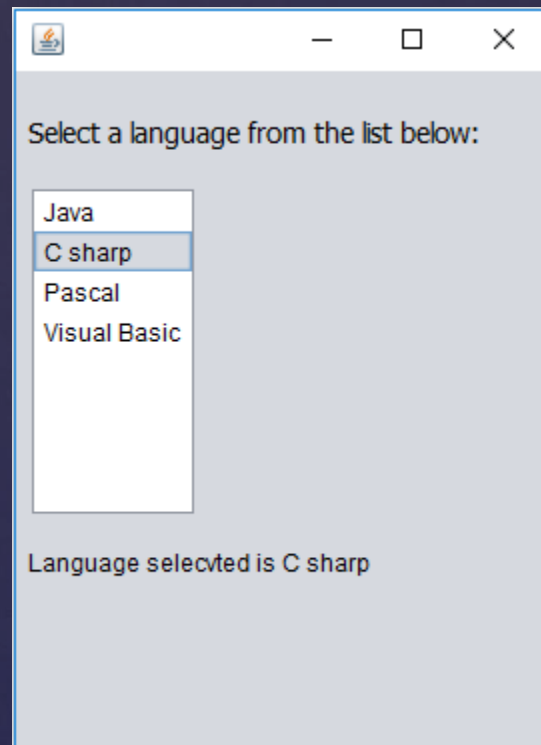
Property	Description
background	Sets the background color.
enabled	Contains enabled state of component- true if enabled else false.
font	Sets the font.
foreground	Sets the foreground color.
model	Contains the values to be displayed in the list.
selectedIndex	Contains the index value of selected option of the control.
selectionMode	Describes the mode for selecting values. <ul style="list-style-type: none">- SINGLE (List box allows single selection only)- SINGLE_INTERVAL (List box allows single continuous selection of options using shift key of keyboard)- MULTIPLE_INTERVAL (List box allows multiple selections of options using ctrl key of keyboard)

Method	Description
getSelectedValue()	Returns the selected value when only a single item is selected, if multiple items are selected then returns first selected value. Returns null in case no item selected Object result= <list-name>.getSelectedValue();
isSelectedIndex()	Returns true if specified index is selected. boolean b = <list-name>.isSelectedIndex(int index);

تصميم واجهه رسومية تقوم بعمل jCobobox لاستخدامها في اختيار إحدى لغات البرمجة كما هو مبين بالشكل التالي:



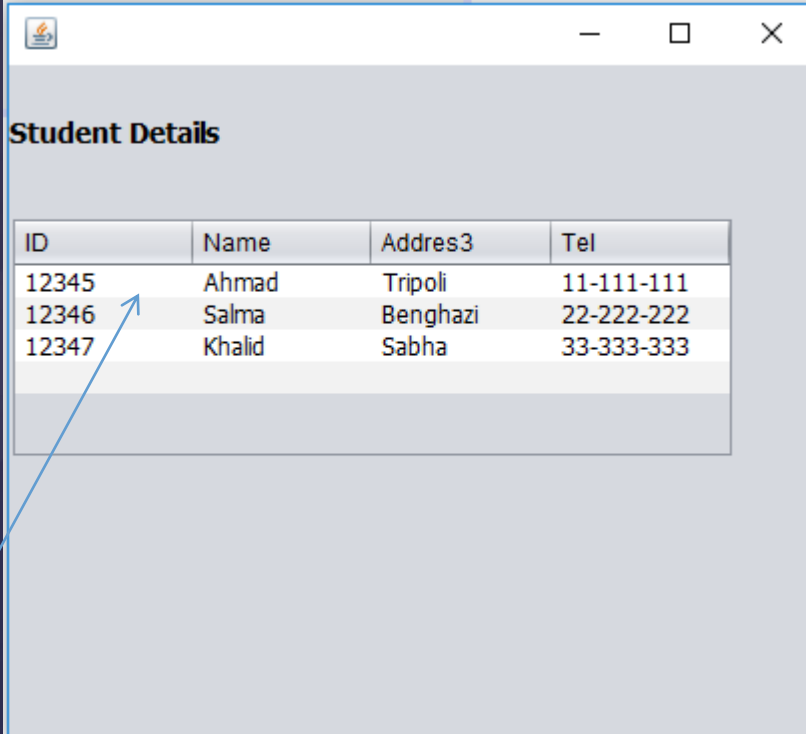
```
private void languageListValueChanged(javax.swing.event.ListSelectionEvent evt) {  
    selectedLanguage.setText("Language selecvted is "+languageList.getSelectedValue());  
}
```



jTable


Title 1	Title 2	Title 3	Title 4

تصميم واجهه رسومية تستخدم jTable لاستخدامها في عرض بيانات مجموعة من الطلبة كما هو مبين بالشكل التالي:



ID	Name	Address	Tel
12345	Ahmad	Tripoli	11-111-111
12346	Salma	Benghazi	22-222-222
12347	Khalid	Sabha	33-333-333

```
private void studentTableMouseClicked(java.awt.event.MouseEvent evt) {
    studentIdLabel.setText("StudentID: "+studentTable.getModel().getValueAt(studentTable.getSelectedRow(),0));
    nameLabel.setText("Name: "+studentTable.getModel().getValueAt(studentTable.getSelectedRow(),1));
    addressLabel.setText("Address: "+studentTable.getModel().getValueAt(studentTable.getSelectedRow(),2));
    telLabel.setText("Tel: "+studentTable.getModel().getValueAt(studentTable.getSelectedRow(),3));
}
```

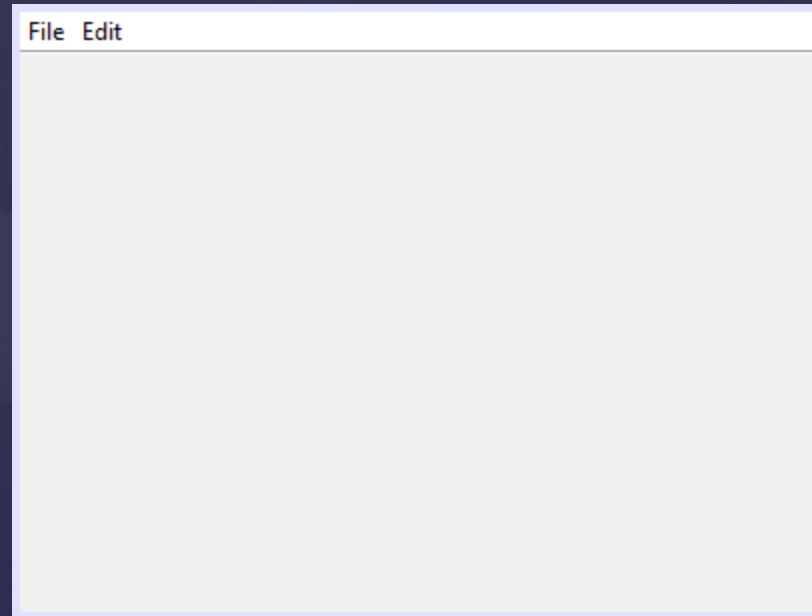
 — □ ×

Student Details

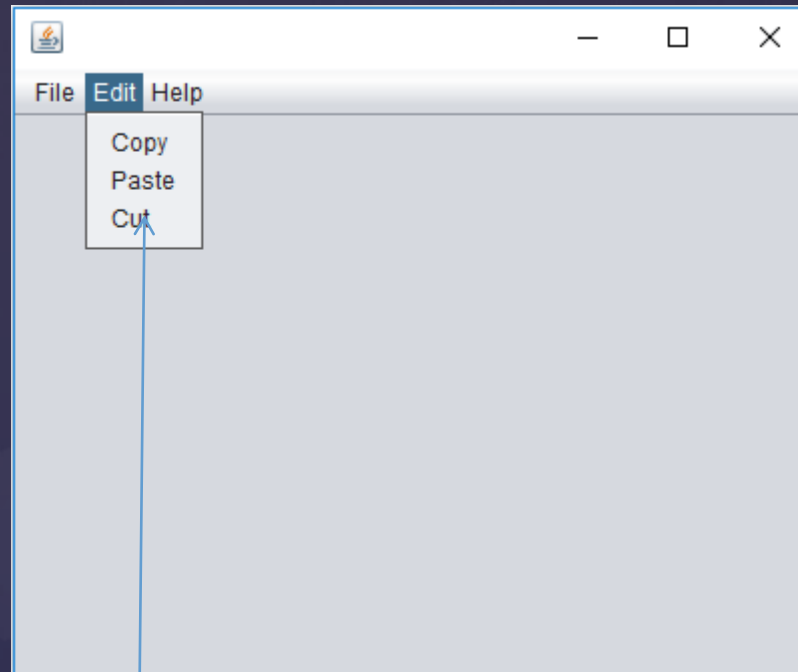
ID	Name	Addres3	Tel
12345	Ahmad	Tripoli	11-111-111
12346	Salma	Benghazi	22-222-222
12347	Khalid	Sabha	33-333-333

StudentID: 12346
Name: Salma
Address: Benghazi
Tel: 22-222-222

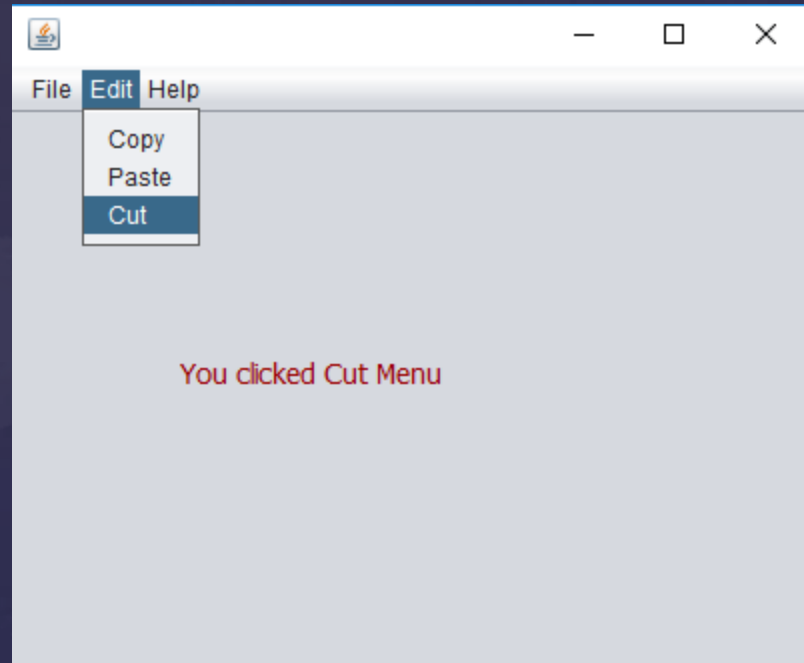
jMenu



تصميم واجهه بها قائمة كما هو مبين بالشكل التالي:



```
private void openMenuActionPerformed(java.awt.event.ActionEvent evt) {  
    messageLabel.setText("You clicked Open Menu");  
}  
  
private void cutMenuActionPerformed(java.awt.event.ActionEvent evt) {  
    messageLabel.setText("You clicked Cut Menu");  
}  
  
private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {  
    messageLabel.setText("You clicked About Menu");  
}
```



شكراً