بسم الله الرحمن الرحيم

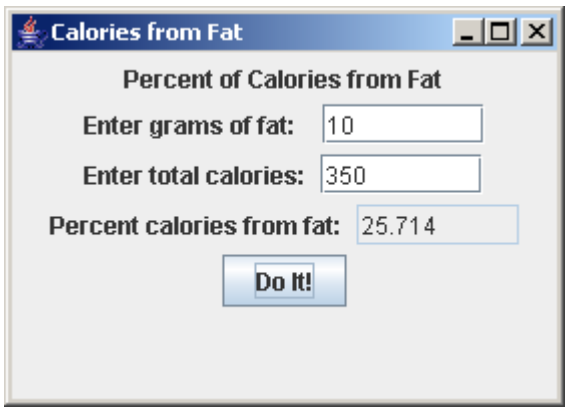
لغة البرمجة جافا
Java Programming Language

الدرس الحادي عشر

برمجة واجهات التطبيقات

7-1. مقدمة .

تتطلب معظم التطبيقات البرمجية بلغة الجافا التفاعل مع عدة كائنات وواجهات رسومية مثل النوافذ والأزرار والمربعات النصية والقوائم المختلفة وغيرها . وتتضمن الحزمة (Swing) مجموعة من الصفوف التي تساعدنا في تصميم تلك الكائنات والواجهات. وفي الوقت نفسه يتجلى تفاعل المستثمر مع التطبيق فيما يقوم به من أعمال وأحداث أثناء التعامل مع ذاك التطبيق .
والصورة أدناه مثال على تطبيق بواجهة رسومية منجزة بلغة الجافا اعتماداً على الحزمة(Swing):



كما تتضمن الحزمة (Swing) مجموعة من الصفوف ، كما هو موضح بالصورة أدناه . ومن تلك الصفوف :

الصف (Object) : وهو الصف الجد والأب لكل الصفوف بلغة الجافا .

الصف (Component) : وهو صف ينتمي للحزمة (AWT) ويقوم بتمثيل كل غرض رسومي على الشاشة.ومن دواله (setVisable(),setBounds()) ، فالأول يهتم بإظهار الكائن الرسومي أو حجبه ، والثاني يهتم بموقع وحجم الكائن على الشاشة.

الصف (Container) : يدعى بالصف الحاوية وينتمي للحزمة (AWT) . وكل غرض منه يمكن أن يحتوي مجموعة من الكائنات الرسومية . وبالتالي فالحاوية رسومياً هي عبارة عن مساحة على الشاشة تتضمن مساحات صغيرة مخصصة للكائنات التي تتضمنها . ومن دواله (add()) لإضافة الكائنات الرسومية.

الصف (Window) : يدعى بالصف النافذة وهو حالة خاصة من الصف الحاوية ، وهي تتضمن رسومياً إطاراً وشريطاً للعنوان مزود بالأزرار (إغلاق ، تكبير ، تصغير). مع التنويه إلى أن المستخدم باستطاعته نقل النافذة وتغيير حجمها.

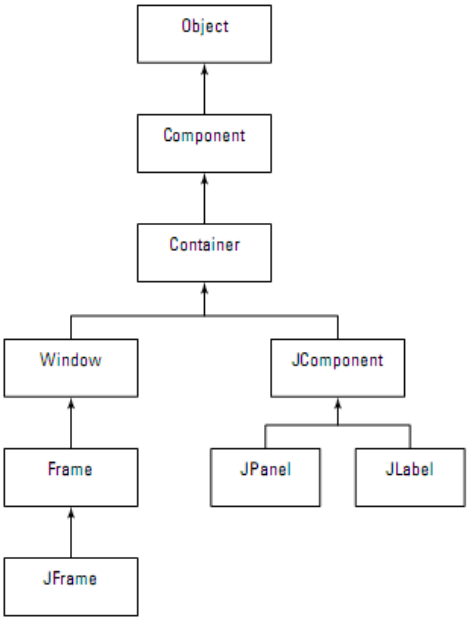
الصف (Frame) : يدعى بالصف الإطار وهو من الصف النافذة وينتمي للحزمة (AWT).

الصف (JFrame) : وهو نسخة مطوّرة للصف (Frame) وينتمي للحزمة (Swing). ومعظم تطبيقات الجافا تتضمن إطاراً على الأقل .

الصف (JComponent) : وهو صف أساس لكافة الكائنات الرسومية ما عدا الإطارات.

الصف (JPanel) : يدعى بصف الألواح وهو عبارة عن حاوية يضم مجموعة من الكائنات الرسومية المتعلقة ببعضها. ورسومياً اللوح هو عبارة عن مساحة مستطيلة الشكل من الشاشة بدون حدود أو بحدود تضم مجموعة من الكائنات ، ويمكن التعامل معه ككائن واحد .

الصف (JLabel) : يدعى بصف الكائنات المخصص لإضافة تسمية توضيحية .



7-2. الكائنات الرسومية.

- **الإطارات (Frames) :**

من أهم الكائنات الرسومية التي تستخدم في برمجة واجهات التطبيقات بلغة الجافا هي الإطار (أي الصف javax.Swing.Jframe). ومن أهم الواني والدوال الهامة المتعلقة بذاك الصف موضحة بالجدول الآتي :

Constructor and methods	Description
JFrame()	الباني الافتراضي وينشئ إطاراً بدون عنوان
JFrame(String title)	باني بوسيط وينشئ إطاراً عنوانه الوسيط
void add(Component c)	دالة لإضافة كائن للإطار
void pack()	دالة لضبط الإطار بحيث يتسع لكلفة الكائنات المضافة إليه
void remove(Component c)	دالة لحذف الكائن الوسيط من الإطار
void setDefaultCloseOperation	دالة وتقابل في معظم الأحيان إغلاق الإطار (JFrame.EXIT_ON_CLOSE)
void setIconImage (Icon image)	دالة لعرض الأيقونة الوسيط عند تصغير الإطار
void setLayout (LayoutManager layout)	دالة لتحديد مدير التصميم المستخدم علماً بأن مدير التصميم الافتراضي هو BorderLayout
void setLocation(int x, int y)	دالة لتحديد موقع الإطار على الشاشة بالإحداثيات (x,y) من الزاوية العليا واليسرى.
void setLocationRelativeTo (Component c)	دالة توسيط الإطار ضمن الشاشة عندما يكون الوسيط مساوياً (null)
void setResizeable(boolean value)	دالة لتغيير حجم الإطار عندما يكون الوسيط (true)
void setSize(int width, int height)	دالة لتحديد حجم الإطار

ولنحاول بناء تطبيق ذو واجهة رسومية بسيطة اعتماداً على الدوال السابقة ، مع الإشارة إلى ضرورة استخدام ثلاثة دوال أساسية على الأقل لإنجاز ذلك وهي موضحة بالصيغة :

```
JFrame frame = new JFrame("This is the title");
frame.setSize(350, 260);
frame.setVisible(true);
```

وبذلك يمكن أن يكون التطبيق بالشكل :

```
import javax.swing.*;
public class HelloFrame1 extends JFrame
```

```
{
    public  static  void  main(String[]  args)
    {
        JFrame  f=new  HelloFrame1();
        f.setSize(200,100);
        //f.setLocation(0,0);
        f.setLocationRelativeTo(null);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setTitle("Hello  World!");
        f.setVisible(true);
    }
}
```

أو بالشكل :

```
import  javax.swing.*;
public  class  HelloFrame2  extends  JFrame
{
    public  static  void  main(String[]  args)
    {
        new  HelloFrame2();
    }
    public  HelloFrame2()
    {
        this.setSize(200,100);
        this.setLocation(0,0);
        //this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Hello  World!");
        this.setVisible(true);
    }
}
```

ملاحظة هامة : للتحكم بمكان توضع الإطار بحرية أكثر يجب معرفة أبعاد شاشة المستخدم، والتي يمكن الحصول عليها باستخدام الصف (Toolkit) من خلال استيراد الحزمة (java.awt.*) واستخدام بعض دواله كما هو موضح بالصيغة التالية :

```
Toolkit tk = Toolkit.getDefaultToolkit();
Dimension d = tk.getScreenSize();
int x = d.width / 2;
int y = d.height / 2
frame.setLocation(x, y);
```

حيث تقوم الدالة (getScreenSise()) بإيجاد أبعاد شاشة المستخدم من خلال متحولات الغرض (d) وهما (int width, height) وإسنادها بطريقة مناسبة لوسطاء الدالة (setLocation()). ويمكن اختبار ذلك كما هو موضح بالتطبيق :

```
import java.awt.*;
import javax.swing.*;
public class HelloFrame3 extends JFrame
{
    public static void main(String[] args)
    {
        new HelloFrame3();
    }
    public HelloFrame3()
    {
        this.setSize(200,100);
        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension d = tk.getScreenSize();
        int x = d.width / 4;
        int y = d.height / 6;
        this.setLocation(x,y);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Hello World!");
        this.setVisible(true);
    }
}
```

- **الألواح (Panels).**

يتمثّل اللوح برمجياً بالصف الحاوية (JPanel) والذي يمكن أن يضم مجموعة من الكائنات الرسومية المتعلقة ببعضها. ورسومياً اللوح هو عبارة عن مساحة مستطيلة الشكل من الشاشة بدون

حدود او بحدود تضم مجموعة من الكائنات ، ويمكن التعامل معه ككائن واحد . ومن أهم البواني والدوال الهامة المتعلقة بذاك الصف موضحة بالجدول الآتي :

Constructor and methods	Description
JPanel()	الباني الافتراضي
JPanel(boolean isDoubleBuffered)	باني بوسيط وإذا كان مساوياً (true) عندها تستخدم تقنية (دوبل بوفر) من أجل السرعة وتستخدم بشكل خاص في الألعاب
JPanel(LayoutManager layout)	باني بوسيط يحدد مدير التصميم علماً بأن مدير التصميم الافتراضي هو (Flowlayout)
void add(Component c)	دالة لإضافة كائن
void remove(Component c)	دالة لحذف كائن
void setLayout(LayoutManager layout)	دالة لتحديد مدير التصميم المستخدم
void setLocation(int x, int y)	دالة لتحديد موقع اللوح
void setToolTipText(String text)	دالة لإدراج تلميح عند وقوف الماوس على منطقة خالية من اللوح

ويمكن إضافة الألواح بعدة طرق وأبسطها كما هو موضح بالصيغة الآتية :

```
// HelloFrame constructor
public HelloFrame()
{
    this.setSize(200,100);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setTitle("Hello, World!");
    JPanel panel = new JPanel();
       // code to add components to the panel goes here
    this.setVisible(true);
}
```

- **التسميات التوضيحية (Labels).**

بعد أن تعلمنا كيف ننشئ الإطارات والألواح لا بد من إنشاء كائن ما لإضافته إلى الألواح والإطارات ومن أبسطها التسمية التوضيحية (labels). والتسمية باختصار كائن لعرض نص ما أو

رسالة ويمكن أن تعرض صورة ونص معاً أيضاً ، مع إمكانية التحكم بخصائص النص وتنسيقه. وتعرّف التسميات باستخدام الصف (JLabel) . ومن أهم البواني والدوال الهامة المتعلقة بذاك الصف موضحة بالجدول الآتي :

Constructor and methods	Description
JLabel()	الباني الافتراضي
JLabel(String text)	باني بوسيط يمثل نص التسمية التوضيحية
String getText()	دالة تعيد نص التسمية التوضيحية
void setText(String text)	دالة لإسناد نص التسمية التوضيحية
void setToolTipText (String text)	دالة لإدراج تلميح عند وقوف الماوس على الكائن
void setVisible (boolean value)	دالة لإخفاء أو إظهار التسمية التوضيحية

ولإضافة تسمية تحمل النص (Hello World!) إلى تطبيقنا السابق نقوم بما يلي كما هو موضح في نص التطبيق :

```
import java.awt.*;
import javax.swing.*;
public class HelloFrame4 extends JFrame
{
    public static void main(String[] args)
    {
        new HelloFrame4();
    }
    public HelloFrame4()
    {
        this.setSize(200,100);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Hello World!");
        JPanel panel = new JPanel();
        JLabel label = new JLabel("Hello, Hello, Hello World!");
        panel.add(label);
        this.add(panel);
        this.setVisible(true);
    }
```

```
}
```

وبتنفيذ التطبيق نحصل على النتيجة الآتية :

● **الأزرار (Buttons) .**

بعد أن تعلمنا كيف ننشئ الإطارات والألواح ونضيف بعض التسميات التوضيحية نحاول الآن إضافة كائنات الأزرار (Buttons) إلى التطبيق. وتعرّف الأزرار باستخدام الصف (JButton) . ومن أهم البوانى والدوال الهامة المتعلقة بذاك الصف موضحة بالجدول الآتي :

Constructor and methods	Description
JButton()	الباني الافتراضي
JButton(String text)	باني بوسيط يمثل نص الزر
doClick()	دالة عند الضغط على الزر يولّد الاستجابة المطلوبة
String getText()	دالة تعيد نص الزر
void setBorderPainted(boolean value)	دالة لإخفاء أو إظهار حدود الزر
void setContentAreaFilled(boolean value)	دالة تظليل الزر
void setEnabled(boolean value)	دالة تنشيط الزر
void setRolloverEnabled(boolean value)	دالة تنشيط الزر بشكل مرئي
void setText(String text)	دالة لإسناد نص الزر
void setToolTipText(String text)	دالة تلميح عند الوقوف على الزر
void setVisible(boolean value)	دالة لإخفاء أو إظهار الزر

ولإضافة زر يحمل النص (click me!) إلى تطبيقنا السابق نقوم بما يلي كما هو موضح في نص التطبيق :

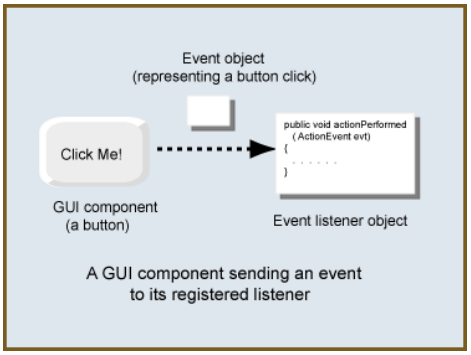
```
import  java.awt.*;
import  javax.swing.*;
public  class  HelloFrame5  extends  JFrame
{
    public  static  void  main(String[]  args)
    {
        new  HelloFrame5();
    }
    public  HelloFrame5()
    {
        this.setSize(200,100);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Hello  World!");
        JPanel  panel  =  new  JPanel();
        JLabel  label  =  new  JLabel("Hello,  Hello,  Hello  World!");
        JButton  button  =  new  JButton("click  me!");
        panel.add(label);
        panel.add(button);
        this.add(panel);
        this.setVisible(true);
    }
}
```

وبتنفيذ التطبيق نحصل على النتيجة الآتية :

7-3. معالجة الأحداث.

لقد اقتصر بناء التطبيق السابق (HelloFrame) حتى الآن على بناء إطار بداخله لوح يتضمن زراً وتسمية توضيحية، علماً بأن التطبيق لا يقوم بأي شيء. ولجعل المستثمر أو المستخدم لمثل هذا

التطبيق أن يتفاعل ويتعامل معه خلال القيام ببعض الأحداث (تحريك الفأرة ، الضغط على الفأرة، الضغط على زر ما ، الكتابة في مربع نص) لا بد من المعالجة بحيث يستجيب التطبيق لتلك الأحداث . ولكي يستجيب التطبيق لأي حدث يُبنى غرض منصت للحدث (event listener object) يمتلك الدوال المنصتة (Listener Methods) لمختلف الأحداث ، مع العلم بأنه يمكن أن يتضمن بعض الدوال الأخرى . إذاً كل منصت أو مستمع للحدث (event listener) هو عبارة عن غرض يصغي للحدث القادم من أي كائن رسومي ولّده المستثمر أو المستخدم خلال تفاعله مع التطبيق . ولكي يكون التطبيق أو البرنامج المزوّد بواجهة رسومية قادراً على أن يستجيب للأحداث يجب بناء غرض منصت للحدث وتسجيل وربط هذا الغرض المنصت للحدث مع الكائن الرسومي الذي يولّد الحدث . وفي الصورة الآتية لدينا كائن رسومي هو عبارة عن زر (button) محتوى في الإطار ، والحدث هو عبارة عن ضغط المستثمر بالفأرة على هذا الزر . وعند الضغط على الزر يُرسل الحدث كغرض إلى الغرض المنصت الذي تم تسجيله لهذا الزر ليقوم بفعل شيء ما .



إن الصف المنصت للزر يجب أن ينفّذ الواجهة الصفية (ActionListener) . وهذه الواجهة تحوي دالة واحدة ذات وسيط واحد هو عبارة عن غرض الحدث المتمثل بالضغط على الزر . وهنالك صفوف عدة تبنى منها الأغراض المنصتة للأحداث والتي بدورها تنفذ مجموعة من الواجهات الصفية ، التي تتضمن مجموعة من الدوال المتعلقة بها . والجداول التالية توضح تلك الصفوف والواجهات الصفية ودوالها ومتى يتم استدعائها :

Event Class	Listener Interface	Description
ActionEvent	ActionListener	لبناء غرض منصت لحدث كالضغط بالماوس على زر
ItemEvent	ItemListener	لبناء غرض منصت لحدث كاختيار بند من قائمة منسدلة
DocumentEvent	DocumentListener	لبناء غرض منصت لحدث كتغيير نص في مربع نصي

WindowEvent	WindowListener	لبناء غرض منصت لحدث يتعلق بتغييرات في الإطار
KeyEvent	KeyListener	لبناء غرض منصت لحدث كالضغط على لوحة المفاتيح
MouseEvent	MouseListener	لبناء غرض منصت لحدث عند تحريك الماوس وسحبه
FocusEvent	FocusListener	لبناء غرض منصت لحدث عند تفيل وتنشيط كائن

Listener Interface	void Methods	When method called
ActionListener	actionPerformed(ActionEvent e)	عند ضغط الزر
ItemListener	itemStateChanged(ItemEvent e)	عند اختيار بند من قائمة
DocumentListener	changeUpdate(DocumentEvent e) insertUpdate(DocumentEvent e) removeUpdate(DocumentEvent e)	عند تغير النص في مربع نصي عند إدخال نص في مربع نصي عند حذف نص في مربع نصي
WindowListener	windowActivated(WindowEvent e) windowClosed(WindowEvent e) windowClosing(WindowEvent e) windowDeactivated(WindowEvent e) windowDeiconified(WindowEvent e) windowIconified(WindowEvent e) windowOpened(WindowEvent e)	عند تفعيل وتنشيط النافذة عند إغلاق النافذة عند محاولة المستخدم إغلاق النافذة عند تعطيل وعدم تفعيل النافذة عند تكبير النافذة عند تصغير النافذة عند فتح النافذة
KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)	عند الضغط على مفتاح عند تحرير المفتاح عند إدخال حرف
MouseListener	mouseClicked(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e)	عند الضغط على الماوس عند تحريك الماوس فوق الكائن عند تحريك الماوس بجانب الكائن عند الضغط على زر الماوس عند تحرير زر الماوس
FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)	عند تنشيط وتعطيل الكائن

واعتماداً على ما سبق فإن بناء تطبيق يستجيب لحدث ما لا بد من القيام بما يلي :

-إضافة الكائن الذي يولدّ ويرتبط بالحدث (كإضافة زر أو تسمية توضيحية للإطار).

حيث يتم التصريح عن متحول متعلق بالكائن المراد إضافته في الإطار بإحدى الصيغ التالية :

```
JPanel panel = new JPanel();
button = new JButton("Click me!");
panel.add(button);
this.add(panel);
_____ OR _____
button = new JButton("Click me!");
this.add(button);
```

-بناء صف يقوم بتنفيذ الواجهة الصفية المنصتة للحدث الذي نريد معالجته بالشكل :

```
public class XXX extends JFrame implements ActionListener
```

-إعادة صياغة الدالة المنصتة للواجهة الصفية بما يتناسب مع الحدث والمعالجة المطلوبة كالآتي :

```
public void actionPerformed(ActionEvent e)
{
   if (e.getSource() == button)
      button.setText("You clicked!");
}
```

حيث أن الدالة (getSource()) تعيد الكائن صاحب الحدث .

-تسجيل الحدث المنصت مع صاحب الحدث ، والذي يتم من خلال استدعاء الدالة (addActionListener()) من قبل الكائن .

ولنحاول الآن اعتماداً على الصفوف والدوال السابقة ولتوضيح ما سبق بناء تطبيق ذو واجهة رسومية تتضمن زراً يستجيب لحدث يتلخص في الضغط عليه وعندها يتغير نص الزر إلى نص آخر يشير إلى مرات الضغط على الزر :

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

```
public class ClickMe1 extends JFrame implements ActionListener
{
    public static void main(String [] args)
    {
        new ClickMe1();
    }
    private JButton button;
    public ClickMe1()
    {
        this.setSize(200,100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("I'm Listening");
        JPanel panel = new JPanel();
        button = new JButton("Click Me!");
        button.addActionListener(this);
        panel.add(button);
        this.add(panel);
        this.setVisible(true);
    }
    private int clickCount = 0;
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button)
        {
            clickCount++;
            if (clickCount == 1)
                button.setText("I've been clicked!");
            else
                button.setText("I've been clicked "+ clickCount + " times!");
        }
    }
}
```

ويمكننا اقتراح صيغة أخرى باستخدام الصفوف الداخلية :

```
import javax.swing.*;
import java.awt.event.*;
public class ClickMe2 extends JFrame
{
    public static void main(String [] args)
```

```
    {
        new ClickMe2();
    }
    private JButton button;
    public ClickMe2()
    {
        this.setSize(200,100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("I'm Listening");
        ClickListener cl = new ClickListener();
        JPanel panel = new JPanel();
        button = new JButton("Click Me!");
        button.addActionListener(cl);
        panel.add(button);
        this.add(panel);
        this.setVisible(true);
    }
    private class ClickListener implements ActionListener
    {
        private int clickCount = 0;
        public void actionPerformed(ActionEvent e)
        {
            if (e.getSource() == button)
            {
                clickCount++;
                if (clickCount == 1)
                    button.setText("I've been clicked!");
                else
                    button.setText("I've been clicked"+ clickCount +" times!");
            }
        }
    }
}
```

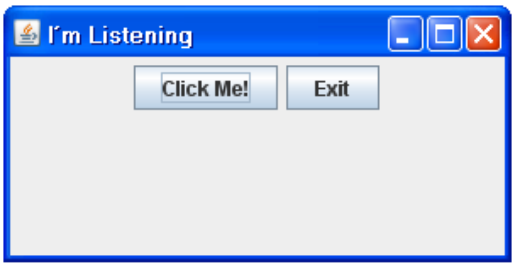
وسنحاول الآن إضافة زر آخر لتطبيقنا السابق يحمل الاسم (exitButton) ، مرتبط بحدث هو الضغط عليه ، بحيث يستجيب التطبيق عند ذلك بالخروج من التطبيق وإنهائه . كما هو موضح بالصيغة التالية :

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == button)
    {
        clickCount++;
        if (clickCount == 1)
            button.setText("I've been clicked!");
        else
            button.setText("I've been clicked "+ clickCount + " times!");
    }
    else if (e.getSource() == exitButton)
    {
        if (clickCount > 0)
          System.exit(0);
        else
        {
            JOptionPane.showMessageDialog(exitButton,"You must click at
            least once!","Not so fast, buddy",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

وبذلك يأخذ التطبيق الشكل الآتي وتكون نافذته كما هي موضحة بالصورة أدناه :

```
import javax.swing.*;
import java.awt.event.*;
public class ClickMe3 extends JFrame implements ActionListener
{
    public static void main(String [] args)
    {
        new ClickMe3();
    }
    private JButton button,exitButton;
    public ClickMe3()
    {
        this.setSize(300,150);
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        this.setTitle("I'm Listening");
        JPanel panel = new JPanel();
        button = new JButton("Click Me!");
        exitButton = new JButton("Exit");
        this.add(button);
```

```
        this.add(exitButton);
        button.addActionListener(this);
        exitButton.addActionListener(this);
        panel.add(button);panel.add(exitButton);
        this.add(panel);
        this.setVisible(true);
    }
    private int clickCount = 0;
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == button)
        {
            clickCount++;
            if (clickCount == 1)
                button.setText("I've been clicked!");
            else
                button.setText("I've been clicked "+ clickCount + " times!");
        }
        else if (e.getSource() == exitButton)
        {
            if (clickCount > 0)
                System.exit(0);
            else
            {
                JOptionPane.showMessageDialog(exitButton,"You must click
  at least once!","Not so fast, buddy",JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```



ملاحظة هامة : نلاحظ من الصيغة السابقة للتطبيق تعطيل إغلاق التطبيق من واجهته وذلك باستخدام التعليمة :

```
this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
```

وبالتالي فإن الخروج من التطبيق وإغلاقه يعتمد على التعليمة :

```
if (clickCount > 0)
    System.exit(0);
```

7-4. الكائنات المرتبطة بعملية الإدخال .

بعد أن تعلمنا كيف ننشئ الإطارات والألواح وإضافة الأزرار والتسميات التوضيحية إليها. لا بد من إضافة بعض الكائنات الرسومية التي تسمح للمستخدم بإدخال بعض المحارف والنصوص ومنها:

- **مربعات النصوص (Text Fields)** :

كائنات رسومية تسمح للمستخدم بإدخال بعض المحارف والنصوص . ومربعات النصوص هي باختصار كائن لإدخال بعض المحارف والنصوص ، مع إمكانية التحكم ببعض الخصائص . وتعرّف المربعات النصيّة باستخدام الصف (JTextField) . ومن أهم البواني والدوال الهامة المتعلقة بذاك الصف موضحة بالجدول الآتي :

Constructor and methods	Description
JTextField()	الباني الافتراضي وينشئ مربعاً نصياً
JTextField(int cols)	باني بوسيط يحدد عرض الكائن
JTextField(String text, int cols)	باني بوسيطين يحددان عرض الكائن ومحتواه
String getText()	دالة تعيد محتوى الكائن
void requestFocus()	دالة طلب تنشيط وتفعيل الكائن
void setColumns (int cols)	دالة لتحديد عرض الكائن
void setEditable (boolean value)	دالة تسمح بتحرير النص إذا كان الوسيط (true)
void setText(String text)	دالة لإسناد الوسيط كمحتوى للكائن
void setToolTipText (String text)	دالة تلميح عند وقوف الماوس فوق الكائن

ولنحاول بناء تطبيق ذو واجهة رسومية بسيطة يوضح استخدام المربعات النصيّة ، كما هو موضح بالمثال الآتي :

```
import javax.swing.*;
import java.awt.event.*;
public class Namer extends JFrame
{
    public static void main(String [] args)
    {
        new Namer();
    }
    private JButton buttonOK;
    private JTextField textName;
    public Namer()
    {
        this.setSize(325,100);
        this.setTitle("Who Are You?");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ButtonListener bl = new ButtonListener();
        JPanel panel1 = new JPanel();
        panel1.add(new JLabel("Enter your name: "));
        textName = new JTextField(15);
        panel1.add(textName);
        buttonOK = new JButton("OK");
        buttonOK.addActionListener(bl);
        panel1.add(buttonOK);
        this.add(panel1);
        this.setVisible(true);
    }
    private class ButtonListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            if (e.getSource() == buttonOK)
            {
                String name = textName.getText();
```

```
            if (name.length() == 0)
            {

                JOptionPane.showMessageDialog
                (Namer.this,"You didn't enter anything!","Moron",
                                JOptionPane.INFORMATION_MESSAGE);
            }
            else
            {
                JOptionPane.showMessageDialog
                (Namer.this,"Good morning " + name,"Salutations",
                                JOptionPane.INFORMATION_MESSAGE);
            }
            textName.requestFocus();
        }
    }
  }
}
```

ملاحظة : إذا أردنا إدخال بعض الأعداد بهدف المعالجة والحساب لا بد من استدعاء الدوال (parseInt(), parseShort(), parseLong(),...) ، حيث يتم تحويل النص المحرفي إلى عدد ، كما هو موضح بالصيغة :

```
int count = Integer.parseInt(textCount.getText());
```

ويمكن تعميم الأمر بإنشاء دالة منطقية يمكن استدعائها من أجل كل مربع نصي للتأكد من عملية إدخال الأعداد بالصيغة التالية :

```
private boolean isInt(JTextField f, String msg)
{
    try
    {
        Integer.parseInt(f.getText());
        return true;
    }
    catch (NumberFormatException e)
    {
        JOptionPane.showMessageDialog(f,"Entry Error", msg,
                                JOptionPane.ERROR_MESSAGE);
        f.requestFocus();
```

```
            return false;
        }
    }
```

- **مناطق نصيّة (Text Areas) :**

كائنات رسومية تسمح للمستخدم بإدخال بعض المحارف والنصوص وبأكثر من سطر .

فالمناطق النصيّة هي باختصار كائن لإدخال عدة أسطر من المحارف والنصوص ، مع إمكانية التحكم ببعض الخصائص . وتعرّف المناطق النصيّة باستخدام الصف (JTextArea) . كما تضاف إليها أشرطة التمرير والتي تعرّف باستخدام الصف (JScrollPane) كما هو موضح بالصيغة التالية :

```
textNovel = new JTextArea(10, 20);
JScrollPane scroll = new JScrollPane(textNovel,
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
panel1.add(scroll);
```

حيث لدينا منطقة نصيّة من عشرة أسطر وعشرون عموداً أضيف إليها شريط التمرير العمودي ، والذي بدوره تمت إضافته إلى اللوح والإطار .

ومن أهم البواني والدوال الهامة المتعلقة بالصف (JTextArea) موضحة بالجدول الآتي :

Constructor and methods	Description
JTextArea()	الباني الافتراضي
JTextArea(int rows, int cols)	باني بوسيطين يحددان عدد الأسطر والأعمدة
JTextArea(String text, int rows, int cols)	باني بثلاث وسطاء تحدد أبعاد الكائن ومحتواه
void append(String text)	دالة لإضافة الوسيط لمحتوى الكائن
int getLineCount()	دالة تعيد عدد الأسطر
String getText()	دالة تعيد نص الكائن
void insert(String str, int pos)	دالة تسمح بإضافة نص في مكان محدد
void requestFocus()	دالة لإسناد الوسيط كمحتوى للكائن
replaceRange(String str, int start, int end)	دالة استبدال النص المحدد بالوسيطين الثاني والثالث بنص الوسيط الأول
void setColumns(int cols)	دالة لتحديد عدد الأعمدة
void setEditable(boolean value)	دالة السماح بتحرير النص

void setLineWrap(boolean value)	دالة التفاف النص
void setText(String text)	دالة لإسناد نص للكائن
void setToolTipText (String text)	دالة تلميح عند وقوف الماوس فوق الكائن
void setWrapStyleWord()	دالة التفاف النص عند نهاية الكلمات

وأما الصف (JScrollPane) فهو يتضمن بانيين مع بعض قيم الوسطاء لأشرطة التمرير الآتي :

Constructor
JScrollPane((Component view)
JScrollPane(Component, int vert, int hor)
Fields
vert= JScrollPane .VERTICAL_SCROLLBAR_ALWAYS
vert= JScrollPane .VERTICAL_SCROLLBAR_AS_NEEDED
vert= JScrollPane .VERTICAL_SCROLLBAR_NEVER
hor= JScrollPane .HORIZONTAL_SCROLLBAR_ALWAYS
hor= JScrollPane .HORIZONTAL_SCROLLBAR_AS_NEEDED
hor= JScrollPane .HORIZONTAL_SCROLLBAR_NEVER

- **مربعات الخيارات (Check Boxes) :**

كائنات رسومية تسمح للمستخدم بتأكيد الموافقة أو عدم الموافقة على شيء ما تم التصريح عنه وتسمى بمربعات الخيارات ، حيث يسمح للمستخدم بالتأكيد على اختيارها أو عدم اختيارها كلاً على حدا (نعم أو لا). وتعرّف مربعات الخيارات باستخدام الصف (JCheckBox) . ومن أهم البواني والدوال الهامة المتعلقة بهذا الصف موضح بالجدول الآتي :

Constructor and methods	Description
JCheckBox()	الباني الافتراضي لمربع الخيارات وبدون اختيار
JCheckBox(String text)	باني بوسيط يحدد نص كقيمة لمربع الخيارات
JCheckBox(String text, boolean selected)	باني بوسيطين الأول يحدد نص مربع الخيارات والثاني يشير إلى اختياره بنعم
void addActionListener (ActionListener listener)	دالة لإضافة دالة منصتة لأي حدث متعلق بالكائن
void addItemListener(ItemListener listener)	دالة لإضافة دالة منصتة لأي حدث يخص (Item)
String getText()	دالة تعيد نص مربع الخيارات
Boolean isSelected()	دالة تعيد حالة مربع الخيارات

void setSelected(boolean value)	دالة لإسناد قيمة منطقية (نعم أو لا) للكائن
void setText(String text)	دالة لإسناد نص كقيمة للكائن
void setToolTipText(String text)	دالة تلميح

ولنحاول توضيح تلك الدوال من خلال تطبيق صغير لطلب شطيرة بيتزا يتضمن إطار مع لوح بداخله ثلاث مربعات خيارات لاختيار نوع البيتزا (سجق – فطر- سمك) بالإضافة إلى زر عند الضغط عليه تظهر رسالة لبيان ما تم طلبه كما هو موضح بالصورة أدناه :

وبناء عليه يمكن صياغة التطبيق كالآتي :

```
import javax.swing.*;
import java.awt.event.*;
import javax.swing.border.*;
public class Pizza1 extends JFrame
{
    public static void main(String [] args)
    {
        new Pizza1();
    }
    private JButton buttonOK;
    private JCheckBox pepperoni, mushrooms, anchovies;
    public Pizza1()
    {
        this.setSize(320,200);
        this.setTitle("Order Your Pizza");
```

```
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ButtonListener bl = new ButtonListener();
        JPanel mainPanel = new JPanel();
        JPanel topPanel = new JPanel();
        Border b2 = BorderFactory.createTitledBorder("Toppings");
        topPanel.setBorder(b2);
        pepperoni = new JCheckBox("Pepperoni");
        topPanel.add(pepperoni);
        mushrooms = new JCheckBox("Mushrooms");
        topPanel.add(mushrooms);
        anchovies = new JCheckBox("Anchovies");
        topPanel.add(anchovies);
        mainPanel.add(topPanel);
        buttonOK = new JButton("OK");
        buttonOK.addActionListener(bl);
        mainPanel.add(buttonOK);
        this.add(mainPanel);
        this.setVisible(true);
    }
    private class ButtonListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            if (e.getSource() == buttonOK)
            {
                String tops = "";
                if (pepperoni.isSelected())          tops += "Pepperoni\n";
                if (mushrooms.isSelected())          tops += "Mushrooms\n";
                if (anchovies.isSelected())           tops += "Anchovies\n";
                String msg = "You ordered a ";
                if (tops.equals(""))                  msg += "no toppings.";
                else        msg += "the following toppings:\n" + tops;
                JOptionPane.showMessageDialog(buttonOK, msg, "Your
                            Order",JOptionPane.INFORMATION_MESSAGE);
                pepperoni.setSelected(false);
                mushrooms.setSelected(false);
                anchovies.setSelected(false);

            }
        }
```

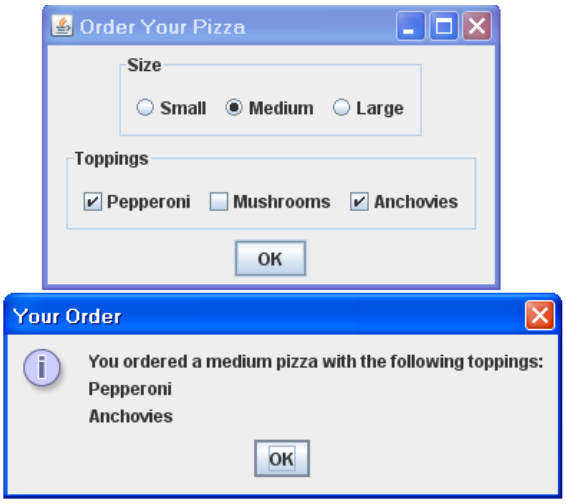
```
    }
}
```

● **أزرار الخيارات (Radio Buttons)** :

كائنات رسومية تسمح للمستخدم بتأكيد الموافقة أو عدم الموافقة على شيء واحد ضمن عدة أشياء تم التصريح عنها وتدعى أزرار الخيارات حيث يسمح للمستخدم باختيار زر وحيد ضمن مجموعة أزرار، (نعم أو لا). وتعرّف أزرار الخيارات باستخدام الصف (JRadioButton) . ومن أهم البواني والدوال الهامة المتعلقة بهذا الصف موضح بالجدول الآتي :

Constructor and methods	Description
JRadioButton()	الباني الافتراضي لزر الخيارات
JRadioButton(String text)	باني بوسيط يحدد نص كقيمة لزر الخيارات
void addActionListener (ActionListener listener)	دالة لإضافة دالة منصتة لأي حدث متعلق بالكائن
void addItemListener(ItemListener listener)	دالة لإضافة دالة منصتة لأي حدث يخص (Item)
String getText()	دالة تعيد نص زر الخيارات
Boolean isSelected()	دالة تعيد حالة زر الخيارات
void setSelected(boolean value)	دالة لإسناد قيمة منطقية (نعم أو لا) للكائن
void setText(String text)	دالة لإسناد نص كقيمة للكائن
void setToolTipText(String text)	دالة تلميح

ولنحاول تحسين التطبيق السابق بحيث نضيف لوحاً يتضمن ثلاث أزرار خيارات تدل على حجم شطيرة البيتزا ، كما هو موضح بالصورة أدناه :



وبناء عليه يمكن صياغة التطبيق كالآتي :

```
import  javax.swing.*;
import  java.awt.event.*;
import  javax.swing.border.*;
public  class  Pizza2  extends  JFrame
{
    public  static  void  main(String  []  args)
    {
        new  Pizza2();
    }
    private  JButton  buttonOK;
    private  JRadioButton  small,  medium,  large;
    private  JCheckBox  pepperoni,  mushrooms,  anchovies;
    public  Pizza2()
    {
        this.setSize(320,200);
        this.setTitle("Order  Your  Pizza");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ButtonListener  bl  =  new  ButtonListener();
        JPanel  mainPanel  =  new  JPanel();
        JPanel  sizePanel  =  new  JPanel();
        Border  b1  =  BorderFactory.createTitledBorder("Size");
        sizePanel.setBorder(b1);
        ButtonGroup  sizeGroup  =  new  ButtonGroup();
        small  =  new  JRadioButton("Small");
        small.setSelected(true);       sizePanel.add(small);
```

```
        sizeGroup.add(small);
        medium = new JRadioButton("Medium");
        sizePanel.add(medium);          sizeGroup.add(medium);
        large = new JRadioButton("Large");
        sizePanel.add(large);           sizeGroup.add(large);
        mainPanel.add(sizePanel);
        JPanel topPanel = new JPanel();
        Border b2 = BorderFactory.createTitledBorder("Toppings");
        topPanel.setBorder(b2);
        pepperoni = new JCheckBox("Pepperoni");
        topPanel.add(pepperoni);
        mushrooms = new JCheckBox("Mushrooms");
        topPanel.add(mushrooms);
        anchovies = new JCheckBox("Anchovies");
        topPanel.add(anchovies);
        mainPanel.add(topPanel);
        buttonOK = new JButton("OK");
        buttonOK.addActionListener(bl);
        mainPanel.add(buttonOK);
        this.add(mainPanel);
        this.setVisible(true);
    }
    private class ButtonListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            if (e.getSource() == buttonOK)
            {
                String tops = "";
                if (pepperoni.isSelected())         tops += "Pepperoni\n";
                if (mushrooms.isSelected())         tops += "Mushrooms\n";
                if (anchovies.isSelected())          tops += "Anchovies\n";
                String msg = "You ordered a ";
                if (small.isSelected())             msg += "small pizza with ";
                if (medium.isSelected())            msg += "medium pizza with ";
                if (large.isSelected())             msg += "large pizza with ";
                if (tops.equals(""))                    msg += "no toppings.";
                else        msg += "the following toppings:\n" + tops;
                JOptionPane.showMessageDialog(buttonOK, msg, "Your
                            Order",JOptionPane.INFORMATION_MESSAGE);
```

```
                pepperoni.setSelected(false);
                mushrooms.setSelected(false);
                anchovies.setSelected(false);
            }
        }
    }
}
```

ملاحظة هامة : نلاحظ استخدام باني الصف (GroupButton) من أجل مجموعة أزرار الخيارات (small, medium, large) وهو ما لم يحصل مع مربعات الخيارات بالصيغة الآتية :

```
ButtonGroup sizeGroup = new ButtonGroup();
sizeGroup.add(small);
sizeGroup.add(medium);
sizeGroup.add(large);
```

ملاحظة هامة : لقد تم تأطير (وضع حدود) كل من الألواح التي تمت إضافتها إلى إطار التطبيق باستخدام كل من الصف (BorderFactory) والواجهة الصفيّة (Border) . من خلال استيراد الحزمتين :

```
import javax.swing.*;
import javax.swing.Border.*;
```

وفيما يلي جدول يوضح دوال الصف (BorderFactory) :

Methods	Description
Border createBevelBorder(int type)	دالة لرسم حدود من الشكل (type)
Border createEmptyBorder (int top, int left, int bottom, int right)	دالة لرسم حدود في مكان محدد
Border createEtchedBorder()	دالة لرسم حدود محفورة
Border createLineBorder()	دالة لرسم حدود خطية
Border createLoweredBevelBorder()	دالة لرسم حدود محفورة ومطموسة
Border createRaisedBevelBorder()	دالة لرسم حدود محفورة نافرة
Border createTitledBorder(String title)	دالة لرسم حدود بعنوان
Border createTitledBorder(Border b, String title)	دالة لرسم حدود من الشكل (b) بالعنوان

- **الأدوات المترلقة (Sliders) :**

كائنات رسومية تسمح للمستخدم بتأكيد الموافقة أو عدم الموافقة على شيء واحد ضمن عدة أشياء تم التصريح عنها وتدعى أزرار الخيارات حيث يسمح للمستخدم باختيار زر وحيد ضمن مجموعة أزرار، (نعم أو لا). وتعرّف أزرار الخيارات باستخدام الصف (JRadioButton) . ومن أهم البواني والدوال الهامة المتعلقة بهذا الصف موضح بالجدول الآتي :

Constructor and methods	Description
JSlider()	الباني الافتراضي
JSlider(int min, int max)	باني بوسيطين يحددان قيم البداية والنهاية
JSlider(int min, int max,int value)	باني بوسطاء تحدد قيم البداية والنهاية والايتدائية
JSlider(int orientation, int min, int max, int value)	باني بوسطاء تحدد قيم البداية والنهاية والايتدائية
void addChangeListener (ChangeListener listener)	دالة تعيد نص زر الخيارات
int getValue()	دالة تعيد حالة زر الخيارات
void setSelected(boolean value)	دالة لإسناد قيمة منطقية (نعم أو لا) للكائن
void setInvert(bolean value)	دالة لإسناد نص كقيمة للكائن
void setInvert(bolean value)	
void setMajorTickSpacing(int value)	
void setMinimum(int value)	
void setMaximum(int value)	
void setMinorTickSpacing(int value)	
setOrientation(int orientation)	
void setPaintLabels(boolean value)	
void setSnapToTicks(boolean value)	دالة تلميح
void setToolTipText	

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == buttonOK)
    {
        int level = slider.getValue();
```

```
            JOptionPane.showMessageDialog(slider,"Remember, this is for
                    posterity.\n"+ "Tell me...how do you feel?", "Level " +
                    level,JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private class SliderListener implements ChangeListener
    {
        public void stateChanged(ChangeEvent e)
        {
            if (slider.getValue() == 50)
            {
                JOptionPane.showMessageDialog(slider,"No! Not 50!","The Machine",
                                        JOptionPane.WARNING_MESSAGE);
            }
        }
    }
```

- **القوائم المنسدلة (Combo Boxs) :**

كائنات رسومية تسمح للمستخدم باختيار بند من قائمة منسدلة ، حيث يسمح للمستخدم باختيار بند وحيد من القائمة . وتعرّف القوائم المنسدلة باستخدام الصف (JComboBox) . ومن أهم البواني والدوال الهامة المتعلقة بهذا الصف موضح بالجدول الآتي :

Constructor and methods	Description
JComboBox()	الباني الافتراضي
JComboBox(Object[] items)	باني بوسيط يحدد مصفوفة أغراض القائمة
JComboBox(Vector[] items)	باني بوسيط يحدد مصفوفة من الأغراض
void addActionListener (ActionListener listener)	دالة لإضافة دالة منصتة لأي حدث متعلق بالكائن
void addItem(Object item)	دالة لإضافة بند للقائمة المنسدلة
void addItemListener(ItemListener listener)	دالة لإضافة دالة منصتة لأي حدث يخص (Item)
Object getItemAt(int index)	دالة تعيد بند من القائمة المنسدلة
int getItemCount()	دالة تعيد ترقيم البند من القائمة المنسدلة
int getSelectedIndex()	دالة تعيد ترقيم البند المحدد
Object getSelectedItem()	دالة تعيد البند المحدد

void insertItemAt(Object item, int index)	دالة لإضافة بند في مكان محدد بالوسيط الثاني
Boolean isEditable()	دالة للسماح بتحرير البند
void removeAllItems()	دالة لحذف كافة البنود من القائمة المنسدلة
void removeItem(Object item)	دالة لحذف بند من القائمة المنسدلة
void removeItemAt(int index)	دالة لحذف بند ترقيمه أو ترتيبه الوسيط
void setEditable(boolean value)	دالة لإسناد السماح أو عدم السماح بتحرير البند
void setMaximumRowCount(int count)	دالة لتحديد عدد البنود المنسدلة عند فتح القائمة
void setSelectedIndex(int index)	دالة تحديد بند ترتيبه الوسيط
void setSelectedItem(Object item)	دالة تحديد البند الوسيط

ويتم إنشاء قائمة منسدلة باستخدام أحد البواني كما هو موضح بالصيغة الآتية :

```
JComboBox combo1 = new JComboBox();
```

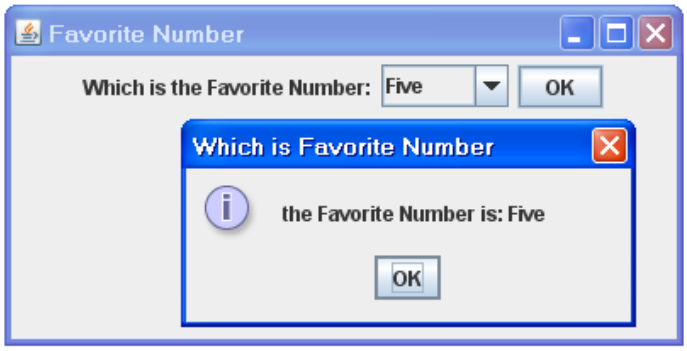
ومن ثم تتم إضافة بنود قائمة على الشكل التالي :

```
combo1.addItem("Bashful");
combo1.addItem("Doc");
combo1.addItem("Dopey");
combo1.addItem("Grumpy");
combo1.addItem("Happy");
combo1.addItem("Sleepy");
combo1.addItem("Sneezy");
```

كما يمكننا إنشاء قائمة منسدلة بدءاً بتعريف مصفوفة بنود أو وجود مصفوفة من الصف (Vector) أو أي قائمة ، كما هو موضح بالصيغة الآتية :

```
String[] theSeven = {"Bashful", "Doc", "Dopey", "Grumpy",
                                    "Happy", "Sleepy", "Sneezy"};
JComboBox combo1 = new JComboBox(theSeven);
---------------------------- or ----------------------------------
JComboBox combo1 = new JComboBox(vector1);
---------------------------- or ----------------------------------
JComboBox combo1 = new JComboBox(arraylist1.toArray());
```

ولنحاول كتابة تطبيق بحيث نضيف لوحاً يتضمن قائمة منسدلة وزر ، كما هو موضح بالصورة أدناه :



ولنحاول كتابة تطبيق بإطار يضاف إليه لوحاً يتضمن قائمة منسدلة وزر ، وعند تنفيذه يمكن الضغط على الزر بعد اختيار بند من القائمة المنسدلة لتظهر رسالة تفيد بأن الرقم المفضل لديكم هو الرقم المحدد ،كما هو موضح بالصورة أدناه :

```
import javax.swing.*;
import java.awt.event.*;
public class Favorite1 extends JFrame
{
    public static void main(String [] args)
    {
        new Favorite1();
    }
    private JLabel label;
    private JButton buttonOK;
    private JComboBox combo;
    public Favorite1()
    {
        this.setSize(400,200);
        this.setTitle("Favorite Number");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ButtonListener bl = new ButtonListener();
        JPanel mainPanel = new JPanel();
        combo=new JComboBox();
                    combo.addItem("One");
                    combo.addItem("Two");
                    combo.addItem("Tree");
                    combo.addItem("Four");
```

```
                combo.addItem("Five");
                combo.addItem("Six");
                combo.addItem("Seven");
                combo.addItem("Eight");
                combo.addItem("Nine");
                combo.addItem("Ten");
                combo.addItem("Eleven");
                combo.addItem("Twelve");
                combo.addItem("Thirteen");


    //String[] comboA = {"One","Two","Tree","Four","Five","Six","Seven",
                        //  "Eight","Nine","Ten","Eleven","Twelve","Thirteen"};
    //combo = new JComboBox(comboA);

    label=new JLabel("Which is the Favorite Number:");
    mainPanel.add(label);
    mainPanel.add(combo);
    buttonOK = new JButton("OK");
    buttonOK.addActionListener(bl);
    //combo.addActionListener(bl);
    mainPanel.add(buttonOK);
    this.add(mainPanel);
    this.setVisible(true);
}
private class ButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        //if (e.getSource() == combo)
        if (e.getSource() == buttonOK)
        {
            String s="";
            if (combo.getSelectedItem()=="One") s="One";
            if (combo.getSelectedItem()=="Two") s="Two";
            if (combo.getSelectedItem()=="Tree") s="Tree";
            if (combo.getSelectedItem()=="Four") s="Four";
            if (combo.getSelectedItem()=="Five") s="Five";
            if (combo.getSelectedItem()=="Six") s="Six";
            if (combo.getSelectedItem()=="Seven") s="Seven";
```

```
            if (combo.getSelectedItem()=="Eight") s="Eight";
            if (combo.getSelectedItem()=="Nine") s="Nine";
            if (combo.getSelectedItem()=="Ten") s="Ten";
            if (combo.getSelectedItem()=="Eleven") s="Eleven";
            if (combo.getSelectedItem()=="Twelve") s="Twelve";
            if (combo.getSelectedItem()=="Thirteen") s="Thirteen";
            //JOptionPane.showMessageDialog(combo,
            JOptionPane.showMessageDialog(buttonOK, "the Favorite
                        Number is: "+s, "Which is Favorite
                    Number",JOptionPane.INFORMATION_MESSAGE);

        }
    }
  }
}
```

● **القوائم (Lists)** :

كائنات رسومية تسمح للمستخدم باختيار بند أو أكثر من القائمة ، وذلك باستخدام مفتاحي (ctrl , shift) وتختلف عن القوائم المنسدلة بإمكانية عرضها وهي منسدلة وبعدة بنود . مع وجود إمكانية تغيير قيم القائمة . وتعرّف القوائم باستخدام الصف (JList) . ومن أهم البواني والدوال الهامة المتعلقة بهذا الصف موضح بالجدول الآتي :

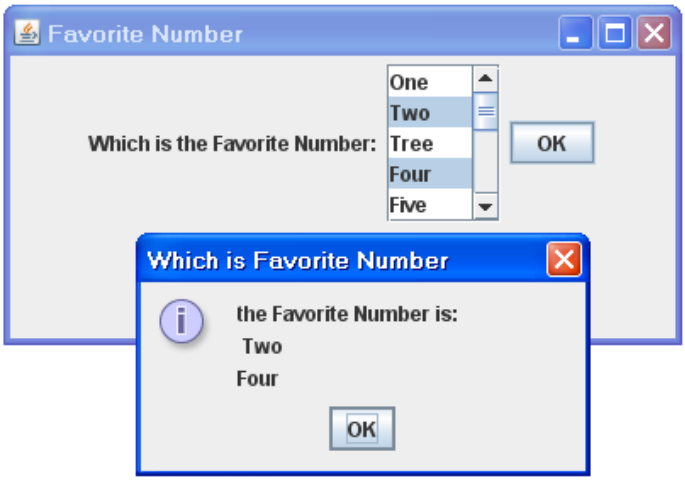
Constructor and methods	Description
JList()	الباني الافتراضي
JList(ListModel list)	باني بوسيط يحدد شكل للقائمة
JList (Object[] items)	باني بوسيط يحدد مصفوفة أغراض القائمة
JList (Vector[] items)	باني بوسيط يحدد مصفوفة من الأغراض للقائمة
void clearSelection()	دالة إلغاء أي تحديد
int getSelectedIndex()	دالة تعيد ترقيم البند المحدد من القائمة
int[] getSelectedIndexes()	دالة تعيد مصفوفة ترقيمات البنود المحددة
Object getSelectedValue()	دالة تعيد بند من القائمة

Object[] getSelectedValues()	دالة تعيد بنود من القائمة
boolean isSelectedIndex(int index)	دالة تأكيد على اختيار بند من خلال ترتيبه
boolean isSelectionEmpty()	دالة تأكيد على عدم وجود أي بند محدد
void setFixedCellHeight(int height)	دالة لتحديد ارتفاع السطر في القائمة
void setFixedCellWidth(int width)	دالة لتحديد عرض السطر في القائمة
void setSelectedIndex(int index)	دالة لتحديد بند في القائمة من خلال ترتيبه
void setSelectedIndices(int[] indices)	دالة لتحديد بنود في القائمة من خلال تراتيبها
void setSelectionMode(int mode)	دالة لتحديد نوع التحديد في القائمة
void setVisibleRowCount(int count)	دالة لتحديد عدد البنود التي ستظهر في القائمة

ويتم إنشاء قائمة باستخدام أحد البواني بعد تمرير مصفوفة البنود ، كما هو موضح بالصيغة الآتية :

```
String[] toppings = {“Pepperoni”, “Sausage”, “Linguica”, “Canadian Bacon”,
“Salami”, “Tuna”, “Olives”, “Mushrooms”, “Tomatoes”,“Pineapple”, “Kiwi”,
“Gummy", "Worms”};
list1 = new JList(toppings);
list1.setVisibleRowCount(5);
JScrollPane scroll = new JScrollPane(list1);
```

ولنحاول كتابة تطبيق بإطار يضاف إليه لوحاً يتضمن قائمة وزر ، وعند تنفيذه يمكن الضغط على الزر بعد اختيار بند من القائمة لتظهر رسالة تفيد بأن الرقم المفضل لديكم هو الرقم المحدد ، مع التنويه إلى وجود إمكانية اختيار أكثر من بند من القائمة (بنود متتالية أو غير متتالية)باستخدام كل من مفتاحي (ctrl , shift) ،كما هو موضح بالصورة أدناه :



وبناء عليه يمكن صياغة التطبيق كالآتي :

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

public  class  Favorite2  extends  JFrame
{
    public  static  void  main(String  []  args)
    {
        new  Favorite2();
    }
    private  JLabel  label;
    private  JButton  buttonOK;
    private  JList  list;
    public  Favorite2()
    {
        this.setSize(400,200);
        this.setTitle("Favorite  Number");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ButtonListener  bl  =  new  ButtonListener();
        JPanel  mainPanel  =  new  JPanel();
        String[]  A  ={"One","Two","Tree","Four","Five","Six","Seven",
                              "Eight","Nine","Ten","Eleven","Twelve","Thirteen"};
        list  =  new  JList(A);
        list.setVisibleRowCount(5);
        JScrollPane  scroll  =  new  JScrollPane(list);
```

```
        label=new JLabel("Which is the Favorite Number:");
        mainPanel.add(label);
        mainPanel.add(scroll);
        buttonOK = new JButton("OK");
        buttonOK.addActionListener(bl);
        mainPanel.add(buttonOK);
        this.add(mainPanel);
        this.setVisible(true);
    }
    private class ButtonListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            if (e.getSource() == buttonOK)
            {
                Object[] F = list.getSelectedValues();
                String s = "the Favorite Number is:\n ";
                for (Object f : F)
                    s+= (String)f + "\n";
                JOptionPane.showMessageDialog(buttonOK, s, "Which is
                  Favorite Number",JOptionPane.INFORMATION_MESSAGE);
                list.clearSelection();
            }
        }
    }
}
```

ملاحظة هامة : يمكن تحرير بنود القائمة باستخدام الصف (DefaultListModel) كما هو موضح بالصيغة الآتية :

```
String[] values ={"One","Two","Tree","Four","Five","Six","Seven",
                  "Eight","Nine","Ten","Eleven","Twelve","Thirteen"};
DefaultListModel model = new DefaultListModel();
for (String value : values)
    model.addElement(value);
list = new JList(model);
model.addElement("Fourteen");
```

حيث يتم إضافة بند للقائمة بعد إنشائها .

ومن الدوال الهامة التي تساعد على تحرير القائمة في الصف (DefaultListModel) مبينة في الجدول الآتي :

Constructor and methods	Description
DefaultListModel()	الباني الافتراضي
void add(Object element, int index)	دالة لإضافة بند أو عنصر للقائمة في مكان محدد
void addElement(Object element))	دالة لإضافة بند أو عنصر للقائمة في النهاية
void clear()	دالة لحذف بنود القائمة
boolean Contains(Object element)	دالة التأكد من وجود بند أو عنصر
Object firstElement()	دالة تعيد البند الأول من القائمة
Object get(int index)	دالة تعيد بند ترتيبه الوسيط
boolean isEmpty()	دالة تعيد فيما إذا كانت القائمة فارغة
Object lastElement()	دالة تعيد البند الأخير من القائمة
void remove(int index)	دالة لحذف بند ترتيبه الوسيط
void removeElement(Object element)	دالة لحذف بند
int size()	دالة تعيد عدد بنود القائمة
Object[] toArray()	دالة تعيد مصفوفة ببنود القائمة

- **قوائم العدادات (Spinners) :**

كائنات رسومية تسمح للمستخدم باختيار بند من خلال زيادة أو تخفيض العداد . وتعرّف قوائم العدادات باستخدام الصف (JSpinner) . ومن أهم البواني والدوال الهامة المتعلقة بهذا الصف موضح بالجدول الآتي :

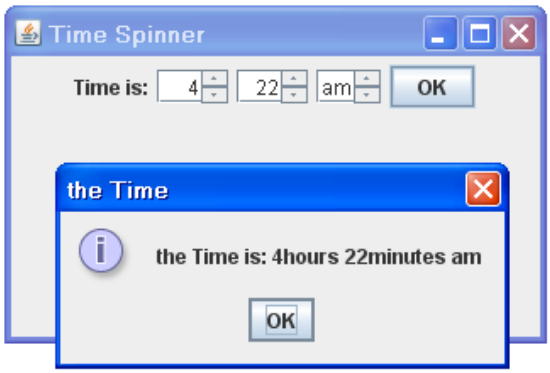
Constructor and methods	Description
JSpinner()	الباني الافتراضي
JSlider(SpinnerModel model)	باني بوسيط يحدد نوع قائمة العداد
void addChangeListener(ChangeListener listener)	دالة لإضافة منصت لحدث التغيير
int getValue()	دالة تعيد قيمة البند

void setToolTipText(String text)	دالة تلميح
SpinnerNumberModel(int init, int min, int max,int step)	قائمة عداد صحيح
SpinnerNumberModel(double init, double min, double max, double step)	قائمة عداد حقيقي
SpinnerListModel(Object[] values)	قائمة عداد قيم
SpinerListModel(List collection)	قائمة عداد من مجموعة

وبناء عليه فلإنشاء قائمة عداد للساعات والدقائق يمكن أن نكتب :

```
JSpinner hours = new JSpinner(new SpinnerNumberModel(1, 1, 12, 1));
JSpinner minutes = new JSpinner(new SpinnerNumberModel(0, 0, 59, 1));
String[] ampmString = {“am”, “pm”};
ampm = new JSpinner(new SpinnerListModel(ampmString));
```

ولنحاول كتابة تطبيق يستخدم قوائم العدادات ، كما هو موضح بالصورة :



وبناء عليه يمكن صياغة التطبيق كالآتي :

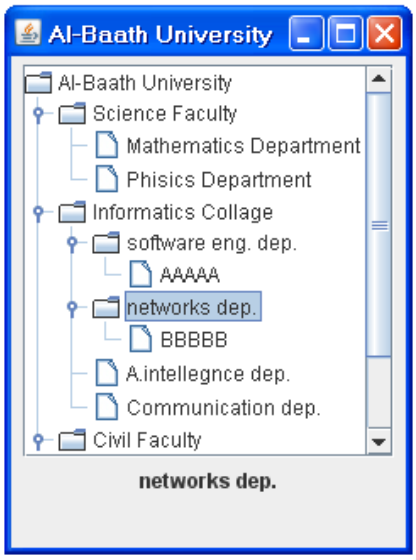
```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class Spinner extends JFrame
{
    public static void main(String [] args)
    {
        new Spinner();
    }
    private JLabel label;
```

```
private JButton buttonOK;
private JSpinner hours,minutes,ampm;
public Spinner()
{
    this.setSize(320,200);
    this.setTitle("Time Spinner");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    ButtonListener bl = new ButtonListener();
    JPanel mainPanel = new JPanel();
    hours=new JSpinner( new SpinnerNumberModel(1, 1, 12, 1));
    minutes = new JSpinner( new SpinnerNumberModel(0, 0, 59, 1));
    String[] ampmString = {"am", "pm"};          ampm = new
    JSpinner(new SpinnerListModel(ampmString));
    label=new JLabel("Time is:");
    mainPanel.add(label);
    mainPanel.add(hours);
    mainPanel.add(minutes);
    mainPanel.add(ampm);
    buttonOK = new JButton("OK");
    buttonOK.addActionListener(bl);
    mainPanel.add(buttonOK);
    this.add(mainPanel);
    this.setVisible(true);
}
private class ButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == buttonOK)
        {
            Object h = hours.getValue();
            String hh=h.toString();
            Object m = minutes.getValue();
            String mm=m.toString();
            String s = (String) ampm.getValue();
            JOptionPane.showMessageDialog(buttonOK,"the Time is:
                    "+hh+"hours "+mm+"minutes "+s, "the Time",
                            JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

```
    }
}
```

● **قوائم الأشجار (Trees) :**

كائنات رسومية تسمح للمستخدم بعرض البيانات على شكل شجرة متفرعة ومشابه لاستعراض المجلدات والأدلة الفرعية على الحاسب كما هو موضح بالصورة أدناه. وتعرّف قوائم الأشجار باستخدام الصف (JTree) إلى جانب بعض الصفوف الداعمة للتحكم بها.



وقبل البدء بإنشاء قوائم الأشجار لنحاول التعرّف إلى بعض المفاهيم المتعلقة بالأشجار :

- Node : كل عنصر في الشجرة تدعى بالعقدة والتي تبنى من الصفوف التي تنفذ الواجهة الصفية (TreeNode) ومن أهم تلك الصفوف والداعمة للتحكم بالأشجار (DefaultMutableTreeNode) .
- (Root node) : ويمثل جذر الشجرة ويمرر كوسيط لباني الشجرة (JTree) .
- (Child node) : العقد الأولاد .
- (Parent node) : العقد الآباء .
- (Sibling nodes) : العقد الأخوة .

- (Leaf node) : العقد الطرفية .
- (Path) : مجموعة العقد التي تشكل مساراً إلى الجذر .
- (Expanded node) : عقدة مع ظهور الأولاد .
- (Collapsed node) : عقدة مع إخفاء الأولاد .

وبناء عليه وإضافة إلى ذلك فلإنشاء شجرة (ككائن رسومي من الصف (Swing)) يتطلب الأمر أولاً إنشاء شجرة وإضافة العقد المطلوبة باستخدام الصف الداعم (DefaultMutableTreeNode) . ومن أهم البواني والدوال في هذا الصف موضح بالجدول الآتي :

Constructor and methods	Description
DefaultMutableTreeNode()	الباني الافتراضي (شجرة بدون عقد)
DefaultMutableTreeNode(Object userObject)	باني بوسيط يحدد غرض العقدة
void add(TreeNode child)	دالة لإضافة عقدة
TreeNode getFirstChild()	دالة تعيد أول ابن للعقدة
DefaultMutableTreeNode getNextSibling()	دالة تعيد العقدة الأخ أو الأخت التالية
TreeNode getParent()	دالة تعيد أب العقدة
Object getUserObject()	دالة تعيد غرض العقدة

ومن أهم البواني والدوال الهامة المتعلقة بالصف (JTree) موضح بالجدول الآتي :

Constructor and methods	Description
void JTree()	الباني الافتراضي
void JTree(TreeNode root)	باني بوسيط يحدد الجذر
void addTreeSelectionListener(TreeSelectionListener listener)	دالة لإضافة منصت
Object getLastSelectedPathComponent()	دالة تعيد العقدة الأخيرة
TreeSelectionModel getSelectionModel()	دالة تعيد نموذج الاختيار
void setVisibleRowCount(int count)	دالة تحدد عدد الأسطر المرئية

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

```
import javax.swing.event.*;
public class SpinOffs extends JFrame
{
    public static void main(String [] args)
    {
        new SpinOffs();
    }
    private JTree tree1;
    private DefaultTreeModel model;
    private JLabel showName;
    public SpinOffs()
    {
        this.setSize(225,325);
        this.setTitle("Al-Baath University");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel panel1 = new JPanel();
        DefaultMutableTreeNode root, fac1, fac2, fac3, dep1, dep2;
        root = new DefaultMutableTreeNode("Al-Baath University");
        fac1 = makeShow("Science Faculty", root);
        makeShow("Mathematics Department", fac1);
        makeShow("Physics Department", fac1);
        fac2 = makeShow("Informatics Collage", root);
        dep1 = makeShow("software eng. dep.", fac2);
        makeShow("AAAAA", dep1);
        dep2 = makeShow("networks dep.", fac2);
        makeShow("BBBBB", dep2);
        makeShow("Artificial intelligence dep.", fac2);
        makeShow("Communication dep.", fac2);
        fac3 = makeShow("Civil engineering Faculty", root);
        makeShow("Dep-1", fac3);
        makeShow("Dep-2", fac3);
        makeShow("Dep-3", fac3);
        tree1 = new JTree(root);
        tree1.getSelectionModel().setSelectionMode(
        TreeSelectionModel.SINGLE_TREE_SELECTION);
        tree1.setVisibleRowCount(12);
        tree1.addTreeSelectionListener(new TreeListener());
        JScrollPane scroll = new JScrollPane(tree1);
        panel1.add(scroll);
        showName = new JLabel();
```

```
        panel1.add(showName);
        this.add(panel1);
        this.setVisible(true);
    }
    private DefaultMutableTreeNode makeShow(
    String title, DefaultMutableTreeNode parent)
    {
        DefaultMutableTreeNode show;
        show = new DefaultMutableTreeNode(title);
        parent.add(show);
        return show;
    }
    private class TreeListener implements TreeSelectionListener
    {
        public void valueChanged(TreeSelectionEvent e)
        {
            Object o = tree1.getLastSelectedPathComponent();
            DefaultMutableTreeNode show;
            show = (DefaultMutableTreeNode) o;
            String title = (String)show.getUserObject();
            showName.setText(title);
        }
    }
}
```

ساهم بنشر الكتاب ولك الأجر والثواب إن شاء الله
لا تنسوني من صالح دعائكم
تم بحمد الله