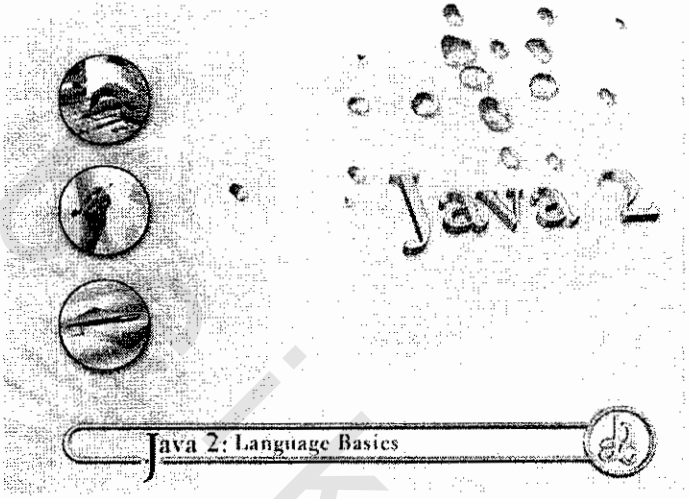
الفصل الحادي والعشرون



تطبيقات مكتبة Swing
Swing Applications

في هذا الفصل نتعرف علي العديد من التطبيقات الهامة للغة Java حيث نتعرف علي العديد من تطبيقات مكتبة Swing ومدير التخطيط Layout Manager ومعالجة الأحداث Event Handling.

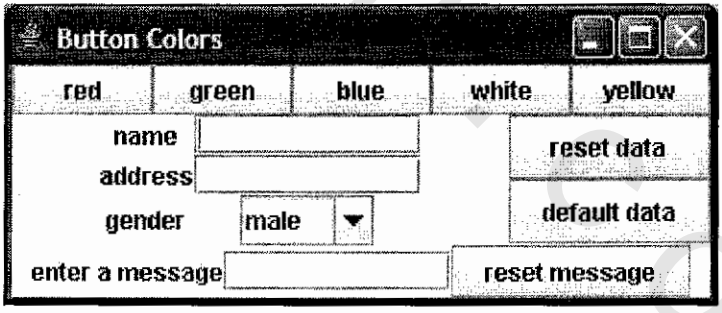
مقدمة:

نتعلم في هذا الفصل مشاريع مكتبة Swing ومدير التخطيط Layout Manager ومعالجة الأحداث Event Handling.

مثال (1): معالجة الأحداث Event Handling:

أولاً: هدف المثال:

في هذا المثال نريد إنشاء برنامج يحتوي علي أداة إطار JFrame كما هو واضح في الشكل (21-1) ، بحيث أنه إذا ضغط المستخدم علي أي من الأزرار الخمسة العليا ، فإن لون خلفية الأزرار يتغير حسب اللون المكتوب علي الزر ، وإذا ضغط علي الزر reset data فإنه يتم مسح بيانات الاسم والعنوان وتغيير النوع إلي ذكر ، وإذا ضغط علي الزر default data فإنه يتم وضع البيانات الافتراضية التالية: الاسم يكون mostafa والعنوان يكون alex والنوع يكون ذكر ، وإذا ضغط علي الزر reset message فإنه يتم مسح بيانات الرسالة.



الشكل (21-1) تنفيذ البرنامج

ثانياً: كود البرمجة:

```
1: import javax.swing.*;
2: import java.awt.*;
3: import java.awt.event.*;
4:
```

```
5:  public class ButtonColors extends JFrame
6:  {
7:    JPanel northpanel,southpanel,eastpanel,centerpanel;
8:    JButton
    redbutton,greenbutton,bluebutton,yellowbutton,whitebu
    tton,
9:    resetbutton,messagebutton,defaultbutton;
10:  JLabel name,address,gender,message;
11:  JTextField nametext,addresstext,messagetext;
12:  JComboBox gendercombo;
13:  GridBagLayout centergrid,southgrid;
14:  GridBagConstraints centergbc,southgbc;
15:
16:  ButtonColors()
17:  {
18:       MyWindowListener        w        =        new
    MyWindowListener();
19:       addWindowListener(w);
20:
21:       Container c = getContentPane();
22:       c.setLayout(new BorderLayout());
23:
24:       northpanel = new JPanel();
25:       southpanel = new JPanel();
26:       eastpanel = new JPanel();
27:       centerpanel = new JPanel();
28:
29:       ButtonListener l = new ButtonListener();
30:
31:       redbutton = new JButton("red");
32:       redbutton.addActionListener(l);
33:
34:       greenbutton = new JButton("green");
35:       greenbutton.addActionListener(l);
```

```
36:
37:     bluebutton = new JButton("blue");
38:     bluebutton.addActionListener(l);
39:
40:     yellowbutton = new JButton("yellow");
41:     yellowbutton.addActionListener(l);
42:
43:     whitebutton = new JButton("white");
44:     whitebutton.addActionListener(l);
45:
46:     resetbutton = new JButton("reset data");
47:     resetbutton.addActionListener(l);
48:
49:     defaultbutton = new JButton("default data");
50:     defaultbutton.addActionListener(l);
51:
52:     messagebutton = new JButton("reset message");
53:     messagebutton.addActionListener(l);
54:
55:     name = new JLabel("name");
56:     address = new JLabel("address");
57:     gender = new JLabel("gender");
58:     message = new JLabel("enter a message");
59:
60:     nametext = new JTextField(10);
61:
62:     nametext.requestFocus();
63:
64:     addresstext = new JTextField(10);
65:     messagetext = new JTextField(10);
66:     String kind[] = {"male","female"};
67:     gendercombo = new JComboBox(kind);
68:
```

```
69:     c.add("North",northpanel);
70:     c.add("South",southpanel);
71:     c.add("East",eastpanel);
72:     c.add("Center",centerpanel);
73:
74:     northpanel.setLayout(new GridLayout(1,5));
75:     eastpanel.setLayout(new GridLayout(2,1));
76:     centergrid = new GridBagLayout();
77:     southgrid = new GridBagLayout();
78:     centergbc = new GridBagConstraints();
79:     southgbc = new GridBagConstraints();
80:     southpanel.setLayout(southgrid);
81:     centerpanel.setLayout(centergrid);
82:
83:     northpanel.add(redbutton);
84:     northpanel.add(greenbutton);
85:     northpanel.add(bluebutton);
86:     northpanel.add(whitebutton);
87:     northpanel.add(yellowbutton);
88:
89:     eastpanel.add(resetbutton);
90:     eastpanel.add(defaultbutton);
91:
92:     centergbc.gridx = 1;
93:     centergbc.gridy = 1;
94:     centergrid.setConstraints(name,centergbc);
95:     centerpanel.add(name);
96:
97:     centergbc.gridx = 2;
98:     centergbc.gridy = 1;
99:     centergrid.setConstraints(nametext,centergbc);
100:        centerpanel.add(nametext);
101:
102:        centergbc.gridx = 1;
```

```
103:        centergbc.gridy = 2;
104:        centergrid.setConstraints(address,centergbc);
105:        centerpanel.add(address);
106:
107:        centergbc.gridx = 2;
108:        centergbc.gridy = 2;
109:
    centergrid.setConstraints(addresstext,centergbc);
110:        centerpanel.add(addresstext);
111:
112:        centergbc.gridx = 1;
113:        centergbc.gridy = 3;
114:        centergrid.setConstraints(gender,centergbc);
115:        centerpanel.add(gender);
116:
117:        centergbc.gridx = 2;
118:        centergbc.gridy = 3;
119:
    centergrid.setConstraints(gendercombo,centergbc);
120:        centerpanel.add(gendercombo);
121:
122:        southgbc.gridx = 1;
123:        southgbc.gridy = 1;
124:        southgrid.setConstraints(message,southgbc);
125:        southpanel.add(message);
126:
127:        southgbc.gridx = 3;
128:        southgbc.gridy = 1;
129:
    southgrid.setConstraints(messagetext,southgbc);
130:        southpanel.add(messagetext);
131:
132:        southgbc.gridx = 5;
133:        southgbc.gridy = 1;
```

```
134:
    southgrid.setConstraints(messagebutton,southgbc);
135:            southpanel.add(messagebutton);
136:
137:            setTitle("Button Colors");
138:            pack();
139:            setVisible(true);
140:        }
141:
142:        public static void main(String args[])
143:        {
144:            ButtonColors buttonColors = new
    ButtonColors();
145:        }
146:
147:        class ButtonListener implements ActionListener
148:        {
149:            public void actionPerformed(ActionEvent e)
150:            {
151:                Object obj = e.getSource();
152:
153:                if ( obj == redbutton )
154:                {
155:                    redbutton.setBackground(Color.red);
156:                }
157:
158:                else if ( obj == greenbutton )
159:                {
160:
    greenbutton.setBackground(Color.green);
161:                }
162:
163:                else if ( obj == yellowbutton )
164:                {
```

```
165:
    yellowbutton.setBackground(Color.yellow);
166:            }
167:
168:            else if ( obj == bluebutton )
169:            {
170:                bluebutton.setBackground(Color.blue);
171:            }
172:
173:            else if ( obj == whitebutton )
174:            {
175:
    whitebutton.setBackground(Color.white);
176:            }
177:
178:            else if ( obj == resetbutton )
179:            {
180:                nametext.setText("");
181:                addresstext.setText("");
182:
    gendercombo.setSelectedItem("male");
183:                nametext.requestFocus();
184:            }
185:
186:            else if ( obj == defaultbutton )
187:            {
188:                nametext.setText("mostafa");
189:                addresstext.setText("alex");
190:
    gendercombo.setSelectedItem("male");
191:                nametext.selectAll();
192:                nametext.requestFocus();
193:            }
194:
```

```
195:            else if ( obj == messagebutton )
196:            {
197:                messagetext.setText("");
198:                messagetext.requestFocus();
199:            }
200:        }
201:    }
202:
203:    class MyWindowListener extends
    WindowAdapter
204:    {
205:        public void windowOpened(WindowEvent e)
206:        {
207:            nametext.requestFocus();
208:        }
209:
210:        public void windowClosing(WindowEvent
    we)
211:        {
212:            dispose();
213:            System.exit(0);
214:        }
215:    }
216:}
```

مثال (2): أحداث الفارة Mouse Events:

أولاً: هدف المثال:

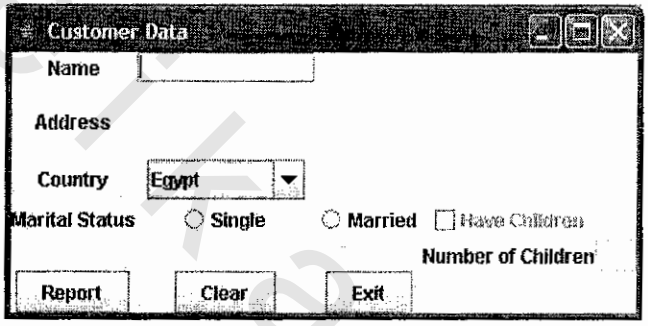
في هذا المثال نريد إنشاء برنامج يحتوي علي أداة إطار JFrame كما هو واضح في الشكل (21-2) ، بحيث يتم تنفيذ الآتي:

- نريد تغيير لون خلفية أداة الإطار JFrame عندما يضغط المستخدم علي زر الفارة Mouse.

- نريد تغيير لون خلفية أداة الإطار JFrame عندما يحرر Release المستخدم زر الفارة Mouse.
- نريد تغيير لون خلفية أداة الإطار JFrame عندما يدخل مؤشر الفارة Mouse داخل مساحة أداة الإطار JFrame كما نريد تغيير شكل مؤشر الفارة Mouse.
- نريد تغيير لون خلفية أداة الإطار JFrame عندما يخرج مؤشر الفارة Mouse من مساحة أداة الإطار JFrame.
- نريد تغيير لون خلفية أداة الإطار JFrame عندما يتحرك مؤشر الفارة Mouse في مساحة أداة الإطار JFrame.
- نريد إظهار رسالة ترحيب للمستخدم عند فتح أداة الإطار JFrame ، كما نريد إيقاف عمل أداة صندوق التحقق JCheckBox الخاصة بالأولاد وبالمثل لأداة النص JTextField الخاصة بعدد الأولاد لأنه من المنطقي أن يكون الشخص متزوجاً قبل أن يكون له أولاد ، ولذلك يتم إيقاف عمل هاتين الأداتين للتأكد من أن الشخص متزوج أولاً.
- نريد نقل مؤشر الفارة Mouse لأداة النص JTextField الخاصة بالعنوان عندما يضغط المستخدم علي زر الإدخال Enter (وهذه العملية معروفة باسم Focusing).
- نريد إظهار رسالة للمستخدم لتبين اختياره عندما يقوم بتغيير الدولة من أداة قائمة الاختيارات JComboBox.
- نريد إيقاف عمل أداة صندوق التحقق JCheckBox الخاصة بالأولاد وبالمثل لأداة النص JTextField الخاصة بعدد الأولاد عندما يقوم المستخدم بالضغط علي أداة زر الراديو JRadioButton الخاصة بالشخص الأعزب لاختيارها.
- نريد إعادة تمكين Enable عمل أداة صندوق التحقق JCheckBox الخاصة بالأولاد عندما يقوم المستخدم بالضغط علي أداة زر الراديو JRadioButton الخاصة بالشخص المتزوج لاختيارها.
- نريد تمكين Enable عمل أداة النص JTextField الخاصة بعدد الأولاد عندما

يقوم المستخدم بالضغط علي أداة صندوق التحقق JCheckBox الخاصة بـالأولاد لاختيارها.

- نريد إغلاق نافذة البرنامج عندما يقوم المستخدم بالضغط علي الزر Exit.
- نريد مسح جميع اختيارات المستخدم عندما يقوم المستخدم بالضغط علي الزر Clear.
- نريد إظهار رسالة لتبين جميع اختيارات المستخدم عندما يقوم المستخدم بالضغط علي الزر Report.



الشكل (21-2) تنفيذ البرنامج

ثانياً: كود البرمجة:

```
1:  import javax.swing.*;
2:  import java.awt.*;
3:  import java.awt.event.*;
4:
5:  public class MyData extends JFrame
6:  {
7:   JLabel
    labelName,labelAddress,labelCountry,labelMaritalStatu
    s,labelNumberOfChildren;
8:   JTextField textName,textNumberOfChildren;
9:   JTextArea textAddress;
10:  JComboBox comboCountry;
11:  ButtonGroup marital;
12:  JRadioButton single,married;
```

```
13: JCheckBox haveChildren;
14: JButton report,clear,exit;
15:
16: Container c;
17: GridBagLayout grid;
18: GridBagConstraints gbc;
19:
20: MyData()
21: {
22:     MyMouseListener listen = new
    MyMouseListener();
23:     addMouseListener(listen);
24:
25:     MyMouseMotionListener motionlisten = new
    MyMouseMotionListener();
26:     addMouseMotionListener(motionlisten);
27:
28:     MyWindowListener w = new
    MyWindowListener();
29:     addWindowListener(w);
30:
31:     MyKeyListener k = new MyKeyListener();
32:
33:     CheckBoxListener checkBoxListener = new
    CheckBoxListener();
34:
35:     MyButtonListener blisten = new
    MyButtonListener();
36:
37:     c = getContentPane();
38:
39:     grid = new GridBagLayout();
40:     gbc = new GridBagConstraints();
41:
```

```
42:       c.setLayout(grid);
43:
44:       labelName = new JLabel("Name");
45:       labelAddress = new JLabel("Address");
46:       labelCountry = new JLabel("Country");
47:       labelMaritalStatus = new JLabel("Marital Status");
48:       labelNumberOfChildren = new JLabel("Number
   of Children");
49:
50:       textName = new JTextField(10);
51:       textName.addKeyListener(k);
52:       textNumberOfChildren = new JTextField(2);
53:       textAddress = new JTextArea(3,10);
54:
55:       String countryOptions[] = {"Egypt","Saudi
   Arabia","Labanon","Morocco"};
56:       comboCountry = new
   JComboBox(countryOptions);
57:
58:       comboCountry.addItemListener(new
   ItemListener()
59:       {
60:           public void itemStateChanged(ItemEvent e)
61:           {
62:               String message = "You are from " +
   (String)comboCountry.getSelectedItem();
63:
   JOptionPane.showMessageDialog(c,message);
64:           }
65:       });
66:
67:       marital = new ButtonGroup();
68:       single = new JRadioButton("Single");
69:       married = new JRadioButton("Married");
```

```
70:        marital.add(single);
71:        marital.add(married);
72:
73:        RadioListener radioListener = new
   RadioListener();
74:        single.addActionListener(radioListener);
75:        married.addActionListener(radioListener);
76:
77:        haveChildren = new JCheckBox("Have
   Children");
78:
   haveChildren.addActionListener(checkBoxListener);
79:
80:        report = new JButton("Report");
81:        report.addActionListener(blisten);
82:        clear = new JButton("Clear");
83:        clear.addActionListener(blisten);
84:        exit = new JButton("Exit");
85:        exit.addActionListener(blisten);
86:
87:        gbc.gridx = 1;
88:        gbc.gridy = 1;
89:        grid.setConstraints(labelName,gbc);
90:        c.add(labelName);
91:
92:        gbc.gridx = 2;
93:        gbc.gridy = 1;
94:        grid.setConstraints(textName,gbc);
95:        c.add(textName);
96:
97:        gbc.gridx = 1;
98:        gbc.gridy = 2;
99:        grid.setConstraints(labelAddress,gbc);
100:           c.add(labelAddress);
```

```
101:
102:        gbc.gridx = 2;
103:        gbc.gridy = 2;
104:        grid.setConstraints(textAddress,gbc);
105:        c.add(textAddress);
106:
107:        gbc.gridx = 1;
108:        gbc.gridy = 3;
109:        grid.setConstraints(labelCountry,gbc);
110:        c.add(labelCountry);
111:
112:        gbc.gridx = 2;
113:        gbc.gridy = 3;
114:        grid.setConstraints(comboCountry,gbc);
115:        c.add(comboCountry);
116:
117:        gbc.gridx = 1;
118:        gbc.gridy = 4;
119:        grid.setConstraints(labelMaritalStatus,gbc);
120:        c.add(labelMaritalStatus);
121:
122:        gbc.gridx = 2;
123:        gbc.gridy = 4;
124:        grid.setConstraints(single,gbc);
125:        c.add(single);
126:
127:        gbc.gridx = 3;
128:        gbc.gridy = 4;
129:        grid.setConstraints(married,gbc);
130:        c.add(married);
131:
132:        gbc.gridx = 4;
133:        gbc.gridy = 4;
134:        grid.setConstraints(haveChildren,gbc);
```

```
135:            c.add(haveChildren);
136:
137:            gbc.gridx = 4;
138:            gbc.gridy = 5;
139:
        grid.setConstraints(labelNumberOfChildren,gbc);
140:            c.add(labelNumberOfChildren);
141:
142:            gbc.gridx = 5;
143:            gbc.gridy = 5;
144:
        grid.setConstraints(textNumberOfChildren,gbc);
145:            c.add(textNumberOfChildren);
146:
147:            gbc.gridx = 1;
148:            gbc.gridy = 6;
149:            grid.setConstraints(report,gbc);
150:            c.add(report);
151:
152:            gbc.gridx = 2;
153:            gbc.gridy = 6;
154:            grid.setConstraints(clear,gbc);
155:            c.add(clear);
156:
157:            gbc.gridx = 3;
158:            gbc.gridy = 6;
159:            grid.setConstraints(exit,gbc);
160:            c.add(exit);
161:
162:            setTitle("Customer Data");
163:            pack();
164:            setVisible(true);
165:            repaint();
166:        }
```

```
167:
168:    public static void main(String args[])
169:    {
170:        MyData myData = new MyData();
171:    }
172:
173:    class MyMouseListener implements
    MouseListener
174:    {
175:        public void mouseClicked(MouseEvent e)
176:        {
177:            c.setBackground(Color.BLUE);
178:        }
179:
180:        public void mouseEntered(MouseEvent e)
181:        {
182:            c.setBackground(Color.RED);
183:            setCursor(Cursor.HAND_CURSOR);
184:        }
185:
186:        public void mouseExited(MouseEvent e)
187:        {
188:            c.setBackground(Color.GREEN);
189:        }
190:
191:        public void mousePressed(MouseEvent e)
192:        {
193:            c.setBackground(Color.YELLOW);
194:        }
195:
196:        public void mouseReleased(MouseEvent e)
197:        {
198:            c.setBackground(Color.BLUE);
199:        }
```

```
200:    }
201:
202:    class    MyMouseMotionListener    extends
    MouseMotionAdapter
203:    {
204:        public void mouseMoved(MouseEvent e)
205:        {
206:            int x = e.getX();
207:            int y = e.getY();
208:
209:            if ( x < 0 )
210:            {
211:                x = 0;
212:            }
213:
214:            if ( x > 255 )
215:            {
216:                x = 255;
217:            }
218:
219:            if ( y < 0 )
220:            {
221:                y = 0;
222:            }
223:
224:            if ( y > 255 )
225:            {
226:                y = 255;
227:            }
228:
229:            c.setBackground(new Color(x,y,0));
230:        }
231:    }
232:
```

```
233:    class MyWindowListener extends
    WindowAdapter
234:    {
235:        public void windowClosing(WindowEvent e)
236:        {
237:            dispose();
238:            System.exit(0);
239:        }
240:
241:        public void windowOpened(WindowEvent e)
242:        {
243:
     JOptionPane.showMessageDialog(MyData.this,"Welc
    ome to my application");
244:            haveChildren.setEnabled(false);
245:            textNumberOfChildren.setEnabled(false);
246:        }
247:    }
248:
249:    class MyKeyListener extends KeyAdapter
250:    {
251:            public void keyReleased(KeyEvent e)
252:            {
253:                char c = e.getKeyChar();
254:
255:                if ( c == '\n' )
256:                {
257:                    textAddress.requestFocus();
258:                }
259:            }
260:    }
261:
262:    class RadioListener implements ActionListener
263:    {
```

```
264:        public void actionPerformed(ActionEvent e)
265:        {
266:            Object source = e.getSource();
267:
268:            if ( source == single )
269:            {
270:                boolean isSingle = single.isSelected();
271:
272:                if ( isSingle == true )
273:                {
274:
    textNumberOfChildren.setText("0");
275:
    textNumberOfChildren.setEnabled(false);
276:                    haveChildren.setEnabled(false);
277:                }
278:            }
279:
280:            else if ( source == married )
281:            {
282:                boolean isMarried =
    married.isSelected();
283:
284:                if ( isMarried == true )
285:                {
286:
    textNumberOfChildren.setText("");
287:                    haveChildren.setEnabled(true);
288:                }
289:            }
290:        }
291:    }
292:
293:    class CheckBoxListener implements ActionListener
294:    {
```

```
295:        public void actionPerformed(ActionEvent e)
296:        {
297:            Object source = e.getSource();
298:
299:            if ( source == haveChildren )
300:            {
301:                boolean isChildren =
    haveChildren.isSelected();
302:
303:                if ( isChildren == true )
304:                {
305:
    textNumberOfChildren.setText("");
306:
    textNumberOfChildren.setEnabled(true);
307:
    textNumberOfChildren.requestFocus();
308:                }
309:
310:                else
311:                {
312:
    textNumberOfChildren.setText("0");
313:
    textNumberOfChildren.setEnabled(false);
314:                }
315:            }
316:        }
317:    }
318:
319:    class MyButtonListener implements
    ActionListener
320:    {
321:        public void actionPerformed(ActionEvent e)
322:        {
```

```
323:            Object source = e.getSource();
324:
325:            if ( source == exit )
326:            {
327:                System.exit(0);
328:            }
329:
330:            else if ( source == clear )
331:            {
332:                textName.setText("");
333:                textAddress.setText("");
334:                textNumberOfChildren.setText("");
335:
    comboCountry.setSelectedItem("Egypt");
336:                single.setSelected(false);
337:                married.setSelected(false);
338:                haveChildren.setSelected(false);
339:                textName.requestFocus();
340:            }
341:
342:            else if ( source == report )
343:            {
344:                String message = "Name is " +
    textName.getText() + "\n";
345:                message += "Address is " +
    textAddress.getText() + "\n";
346:                message += "Country is " +
    comboCountry.getSelectedItem() + "\n";
347:                message += "Single is " +
    single.isSelected() + "\n";
348:                message += "Marrried is " +
    married.isSelected() + "\n";
349:                message += "Have Children is " +
    haveChildren.isSelected() + "\n";
```

```
350:                    message += "Number Of Children is "
    + textNumberOfChildren.getText();
351:
352:
     JOptionPane.showMessageDialog(MyData.this,
    message);
353:                }
354:            }
355:        }
356:}
```

ملحوظات:

- إن عملية تغيير لون خلفية أداة الإطار JFrame به بعض المشاكل في لغة Java حيث لا يتم تغيير اللون بشكل سليم عند استخدام الدالة ( )setBackground ، وسبب ذلك هو طريقة عمل أدوات Swing التي تقوم بإنشاء حاوية Container ثم يتم إضافة الأدوات المطلوبة إلي الحاوية Container ، ولذلك عندما تريد تغيير لون خلفية أداة الإطار JFrame ، فقم باستخدام الدالة ( )setBackground مع الحاوية Container وليس مع أداة الإطار JFrame ، وستحصل علي نفس النتيجة المطلوبة وسيتم تغيير لون خلفية أداة الإطار JFrame.
- عند استخدام أدوات AWT القديمة ، لم تكن هذه المشكلة موجودة حيث كان يتم إضافة الأدوات إلي الفصيلة Frame مباشرة عن طريق الدالة ()add بدلاً من استخدام الفصيلة Container ، ولذلك كان يمكننا استخدام الدالة ( )setBackground مع الفصيلة Frame لتغيير لون الخلفية.

ملخص الفصل:

- تعلمنا في هذا الفصل العديد من تطبيقات مكتبة Swing ومدير التخطيط Layout Manager ومعالجة الأحداث Event Handling.
- في الفصل القادم نتعلم - بإذن الله - كيفية استخدام التوثيق Documentation للغة Java ، فتابع معنا الفصل القادم.