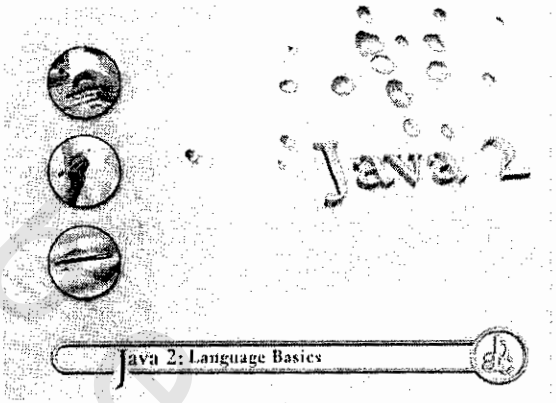
الفصل السابع عشر



معالجة الحدث Event Handling

في هذا الفصل نتناول واحداً من أهم موضوعات البرمجة وهو الخاص بمعالجة الحدث Event Handling وذلك من خلال النقاط التالية:

- الحدث Event.
- معالجة الحدث Event Handling.
- مكونات الحدث Event (كائن الحدث Event Object ومصدر الحدث Event Source ومعالج الحدث Event Handler).
- كيف نتعامل مع الحدث Event؟
- أحداث الزر JButton Events.
- الاستدعاء الرجعي Callback.
- الفصائل الداخلية Inner Classes.
- معالجة حدث أكثر من زر JButton باستخدام فصائل Classes متعددة.
- معالجة حدث أكثر من زر JButton باستخدام فصيلة Class واحدة.
- معالجة حدث أكثر من زر JButton باستخدام الفصائل الداخلية اللاسمية Inner Anonymous Classes.
- أحداث أداة الإطار JFrame Events.
- معالجة أحداث الإطار JFrame Events عن طريق Adapter.
- معالجة الحدث Event Handling عن طريق استخدام نفس الفصيلة class.
- أحداث الضغط علي زر من لوحة المفاتيح Key Events.
- أحداث تغيير النص في أداة النص JTextField.
- أحداث تغيير موضع نقطة الإدخال في أداة النص JTextField.
- أحداث أداة شريط التمرير JScrollBar وأداة قائمة الاختيارات JComboBox وأداة السرد JList وأداة زر الراديو JRadioButton وأداة صندوق التحقق JCheckBox وأداة اللوحة المبوبة JTabbedPane وأداة القوائم JMenu وأحداث الفارة Mouse Events وأحداث تحريك الفارة Mouse Motion Events.

مقدمة:

- تناولنا فى الفصلين السابقين أدوات مكتبة Swing وعرفنا كيف يمكن أن نضع أكثر من أداة Control في الواجهة الرسومية GUI وكيفية تحديد موضعهم باستخدام مدير التخطيط Layout Manager.
- ولكن هناك المزيد ، لأن تصميم وتنفيذ شاشات الواجهة الرسومية GUI فقط يعتبر كإنشاء سيف بلا مقبض ، فلا بد للبرنامج أن يتفاعل مع المستخدم وأن تكون هناك قابلية لتبادل المعلومات بين المستخدم والبرنامج بحيث يمكن للمستخدم أن يدخل بعض البيانات للبرنامج ثم يعطي للبرنامج أمراً لتنفيذ وإخراج نتيجة ما تعتمد علي البيانات التي أدخلها المستخدم.
- وفي هذا الفصل نتعرف علي كيفية معالجة الأحداث المختلفة Event Handling التي تتيح للمستخدم التفاعل مع البرنامج كما سنري في هذا الفصل.
- قبل البدء في شرح الأمثلة ، فلا بد أولاً أن نتعرف علي بعض المفاهيم الأساسية.

الحدث Event:

- الحدث Event هو أى شئ يفعله المستخدم في نافذة التطبيق ، فمثلاً إذا ضغط المستخدم علي زر JButton في التطبيق فإنه بذلك يكون قد أنشأ حدثاً Event ، وبالمثل يتم إنشاء حدث Event عند الضغط علي زر الفارة Mouse وعند تحريك الفارة Mouse وعند فتح نافذة التطبيق نفسها أو إغلاقها أو تصغيرها Minimize أو تكبيرها Maximize.
- أي أنه توجد العديد من الأحداث Events في أي برنامج ، ولكن ليست كل أفعال المستخدم ذات أهمية ، فنجد أن بعض الأحداث Events ذات أهمية ولا بد أن يكون للبرنامج رد فعل لهذه الأحداث Events ، أما بعض الأحداث Events الأخري فلا أهمية لها ولا نريد إنشاء رد فعل لها.
- ولكي نفهم النقطة السابقة ، فيجب أن نذكر أن أهمية الحدث Event تنبع من احتياجات التطبيق نفسه ، فمثلاً إذا قمت بتصميم برنامج رسم ، إذن فتحريك وسحب وضغط مؤشر الفارة Mouse تعتبر أحداثاً Events ذات أهمية قصوي

لأن معظم برامج الرسم تعتمد اعتماداً كبيراً علي الفارة Mouse ، أما إذا قمت بتصميم برنامج كتابة ، إذن فتحريك مؤشر الفارة Mouse يعتبر حدثاً Event بلا أهمية ، فتجد أن برنامج الوورد Word مثلاً لا يقوم بتنفيذ أي وظيفة عند تحريك مؤشر الفارة Mouse ، ولكن الضغط علي زر الفارة Mouse يعتبر حدثاً Event ذا أهمية قصوي لأنه يحدد موضع بداية الكتابة ، وبالمثل بالنسبة لسحب Drag مؤشر الفارة Mouse لأنه يقوم بتظليل select النصوص في البرنامج.

- إذن نخرج من المناقشة السابقة بأن كل أداة لها أحداث Events خاصة بها بحيث أن بعض الأحداث Events تكون ذات أهمية ونريد أن يكون للبرنامج رد فعل لهذه الأحداث Events.
- ويعتمد رد فعل البرنامج علي الأوامر التي نكتبها لكل حدث Event وهو ما يعرف باسم معالجة الحدث Event Handling.

معالجة الحدث Event Handling:

هى رد فعل البرنامج للحدث Event الذى يطلقه المستخدم وتتمثل في كتابة مجموعة من الأوامر لتقوم بتنفيذ وظيفة معينة عند وقوع الحدث Event.

مكونات الحدث Event:

يتكون الحدث من 3 مكونات:

1. كائن الحدث Event Object.
2. مصدر الحدث Event Source.
3. معالج الحدث Event Handler.

(1) كائن الحدث Event Object:

عندما يتفاعل المستخدم مع البرنامج بضغط لوحة المفاتيح أو الضغط على الفارة Mouse مثلاً ، فهنا يتولد الحدث Event الذي يتم تمثيله بهدف object يصف الحدث Event نفسه فى لغة Java ، وتحتوى لغة Java على عدد كبير من الفصائل classes التى تصف العديد من الأحداث Events.

مثال لبعض الأحداث:

- عند الضغط على زر يتولد هدف object من نوع الفصيلة ActionEvent.
- عند تحريك الفارة Mouse يتولد هدف object من نوع الفصيلة MouseEvent.
- عند تغيير النص في أداة النص JTextField يتولد هدف object من نوع الفصيلة TextEvent.
- عند التعامل مع النافذة من تصغير Minimize وتكبير Restore يتولد هدف object من نوع الفصيلة WindowEvent.

(2) مصدر الحدث Event Source:

هو الأداة control التي ولدت الحدث Event ، فمثلاً عند الضغط على زر فإنك أنشأت حدثاً Event ومصدر هذا الحدث Event هو هذا الزر.

(3) معالج الحدث Event Handler:

- هي دالة Method مختصة بالتعامل مع حدث Event معين وتستطيع تنفيذ بعض الأوامر التي نكتبها كرد فعل للحدث Event.
- وفي لغة Java ، كل حدث Event له دالة Method محددة تتعامل معه ، وتجد دائماً أن هذه الدالة Method تستقبل معاملاً Parameter من نوع الهدف object المتولد. قم بالرجوع إلي نقطة كائن الحدث Event Object لتتذكر بعض أمثلة الأحداث Events المتولدة في لغة Java.
- مثلاً: معالج حدث الضغط علي الزر هو الدالة

```
public void actionPerformed (ActionEvent e)
```

- وكما تري فإن هذه الدالة Method تستقبل معاملاً Parameter من نوع الفصيلة ActionEvent والتي - كما ذكرنا - يتولد منها هدف Object عند الضغط علي الزر. إذن هذه الدالة Method تعالج حدث الضغط علي الزر.
- أيضاً معالج حدث إغلاق النافذة هو الدالة

```
public void windowClosing (WindowEvent w)
```

- وكما تري فإن هذه الدالة Method تستقبل معاملاً Parameter من نوع الفصيلة WindowEvent والتي تتولد عند محاولة إغلاق نافذة الرنامج. إذن هذه الدالة Method تعالج حدث إغلاق النافذة.
- تتبقي نقطة واحدة لاستكمال معالجة الحدث Event وهي أنه لا بد أن يتم تعريف البرنامج بأن الفصيلة class التي تتولي معالجة الحدث Event هي المسئولة عن أحداث الأداة التي تمثل مصدر الحدث Event Source ، ويتم ذلك عن طريق إنشاء هدف Object من نوع الفصيلة class التي تتولي معالجة الحدث Event ثم استدعاء دالة Method خاصة لكل نوع من أنواع الأحداث Events ، فمثلاً لإضافة مستمع حدث Event Listener لحدث الضغط علي زر JButton ، فإننا نكتب:

```
JButton button = new JButton("Click");
MyButtonListener listener = new MyButtonListener();
button.addActionListener(listener);
```

- حيث تم استخدام الدالة addActionListener( ) لإضافة مستمع حدث Event Listener الضغط علي زر JButton.

كيف نتعامل مع الحدث Event؟

- تتم معالجة الحدث Event عن طريق إنشاء فصيلة class تسمى بمستمع الحدث Event Listener ، وهذه الفصيلة هي المسئولة عن معالجة الحدث Event Handling وذلك عن طريق احتوائها علي الدوال Method المسئولة عن معالجة الحدث Event.
- هذه الفصيلة class تتميز بشيئين حتي يمكننا القول بأنها مستمع حدث Event Listener:

1. لا بد لهذه الفصيلة class أن تنفذ الـ interface الذي يتناسب مع الحدث Event ، فمثلاً: لإنشاء فصيلة class لمعالجة حدث الضغط علي الزر ، فإن الـ interface الذي ننفذه هو ActionListener. لمعالجة حدث إغلاق النافذة فإن الـ interface الذي ننفذه هو WindowListener.

2. لا بد لهذه الفصيلة class أن تنفذ جميع الدوال Methods الموجودة في الـ interface الذي تنفذه.

الجدول التالي يلخص النقاط السابقة.

الحدث Event	نوع الهدف المتولد Event Object	الـ interface المطلوب تنفيذه	الدالة Method التي تعالج الحدث Event Handler
الضغط علي زر	ActionEvent	ActionListener	actionPerformed
إغلاق النافذة	WindowEvent	WindowListener	windowClosing
الضغط علي زر من لوحة المفاتيح keyboard	KeyEvent	KeyListener	keyPressed

إذن لإنشاء فصيلة class تعالج حدث الضغط علي زر مثلاً ، فإن هذه الفصيلة class تأخذ الشكل التالى:

```
class ButtonListener implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
        //event handling code
    }
}
```

من الجدول السابق ، فإنه لمعالجة حدث الضغط علي الزر ، فإننا لا بد أن ننشئ فصيلة class تنفذ الـ interface المسمي ActionListener والذي يحتوي علي الدالة actionPerformed( ) التي تستقبل هدفاً object من نوع الحدث Event المتولد وهو ActionEvent ، وبداخل هذه الدالة Method فإننا نكتب كود البرمجة الذي نريد تنفيذه عند الضغط علي الزر.

أحداث الزر JButton Events:

تتوافر الأحداث التالية لأداة الزر JButton:

- حدث الضغط علي الزر.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع أداة الزر JButton الذي نريد معالجة حدث الضغط عليه.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ActionListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدالة Method التالية:

```
public void actionPerformed(ActionEvent e)
```

- وبداخل هذه الدالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي أداة الزر JButton عن طريق الدالة addActionListener( ).

المثال التالي يوضح النقاط السابقة.

مثال (1): معالجة أحداث أداة الزر JButton:

أولاً: هدف المثال:

نريد أن ننشئ تطبيقاً يحتوي على أداة زر JButton واحدة فقط وإذا ضغط المستخدم علي هذا الزر فإنه يتم إغلاق البرنامج تماماً.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class ButtonEventUsingSeparateClass extends
   JFrame
```

```
6.  {
7.   JButton exit;
8.
9.   ButtonEventUsingSeparateClass()
10.  {
11.       Container c = getContentPane();
12.       c.setLayout(new BorderLayout());
13.
14.       exit = new JButton("exit");
15.       ButtonListener b = new ButtonListener();
16.       exit.addActionListener(b);
17.       c.add("Center",exit);
18.
19.       setTitle("event");
20.       setSize(200,200);
21.       setVisible(true);
22.  }
23.
24.  public static void main(String args[])
25.  {
26.       ButtonEventUsingSeparateClass event = new
     ButtonEventUsingSeparateClass();
27.  }
28.}
29.
30.class ButtonListener implements ActionListener
31.{
32.  public void actionPerformed(ActionEvent e)
33.  {
34.       System.exit(0);
35.  }
36.}
```

ثالثاً: شرح الكود:

- في السطور من 1 إلي 3 يتم تضمين الحزم packages اللازمة لعمل البرنامج.
- في السطر رقم 5 يتم إنشاء الفصيلة ButtonEventUsingSeparateClass التي ترث من الفصيلة JFrame حتي يمكنها إنشاء شاشة البرنامج كما شرحنا سابقاً.
- في السطر رقم 7 يتم إنشاء متغير من نوع JButton.
- في السطر رقم 9 يتم إنشاء دالة البناء Constructor.
- في السطر رقم 14 يتم إنشاء هدف Object من نوع JButton والذي يمثل أداة العنوان JButton.
- في السطر رقم 15 يتم إنشاء هدف object من نوع الفصيلة ButtonListener والمسئولة عن معالجة حدث الضغط علي الزر JButton Events كما سنري لاحقاً.
- في السطر رقم 16 يتم إضافة مستمع الحدث Event Listener عن طريق الدالة addActionListener( ).
- في السطور من 24 إلي 27 يتم إنشاء الدالة الرئيسية main() التي يتم تنفيذها عند تنفيذ البرنامج وفيها يتم إنشاء هدف Object من نوع الفصيلة ButtonEventUsingSeparateClass وينتج عن ذلك استدعاء دالة البناء Constructor وإظهار نافذة البرنامج.
- فى السطور من 30 إلى 36 نجد أننا أنشأنا تعريف الفصيلة ButtonListener التى تنفذ الـ interface المسمي ActionListener. ونلاحظ وجود تعريف الدالة المسماة actionPerformed() التي تستقبل هدفاً object من نوع الفصيلة ActionEvent الذى يمثل حدث الضغط علي الزر.
- فى هذه الدالة Method وضعنا رد الفعل الذى نريده وهو إغلاق البرنامج تماماً ويتم ذلك عن طريق استدعاء الدالة exit() والموجودة في الفصيلة class المسماة System.
- الدالة exit() تستقبل معاملاً Parameter من نوع int ، والمتعارف عليه أن معامل Parameter الدالة exit() إذا كان رقماً غير الصفر ، فذلك يعني أنك أغلقت البرنامج نتيجة حدوث خطأ ما ، أما إذا كان معامل Parameter الدالة exit() صفراً فذلك يعني إتمام البرنامج بنجاح ودون حدوث خطأ.

قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-1) ويمكنك تجربة الضغط علي الزر للتأكد من إغلاق البرنامج عند الضغط عليه.

الشكل (17-1) حدث الزر JButton

مثال (2): معالجة أحداث أداة الزر JButton:

أولاً: هدف المثال:

نريد أن ننشئ تطبيقاً يحتوي علي أداة نص JTextField وأداة زر JButton ، وعند الضغط على الزر JButton ، فإنه يتم مسح النص الموجود في أداة النص JTextField.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class ButtonEventUsingSeparateClass2 extends
   JFrame
6. {
7.   JButton reset;
8.   JTextField text;
9.   JLabel name;
10.  GridBagLayout grid;
```

```
11. GridBagConstraints gbc;
12.
13. ButtonEventUsingSeparateClass2()
14. {
15.     Container c = getContentPane();
16.
17.     reset = new JButton("reset");
18.
19.     ButtonListener b = new ButtonListener();
20.     reset.addActionListener(b);
21.
22.     text = new JTextField(20);
23.     name = new JLabel("name");
24.
25.     grid = new GridBagLayout();
26.     gbc = new GridBagConstraints();
27.     c.setLayout(grid);
28.
29.     gbc.gridx = 1;
30.     gbc.gridy = 1;
31.     grid.setConstraints(name,gbc);
32.     c.add(name);
33.
34.     gbc.gridx = 3;
35.     gbc.gridy = 1;
36.     grid.setConstraints(text,gbc);
37.     c.add(text);
38.
39.     gbc.gridx = 3;
40.     gbc.gridy = 3;
41.     grid.setConstraints(reset,gbc);
42.     c.add(reset);
43.
44.     setTitle("event");
45.     setSize(350,100);
```

```
46.       setVisible(true);
47.  }
48.
49.  public static void main(String args[])
50.  {
51.       ButtonEventUsingSeparateClass2 event = new
     ButtonEventUsingSeparateClass2();
52.  }
53.}
54.
55.class ButtonListener implements ActionListener
56.{
57.  public void actionPerformed(ActionEvent e)
58.  {
59.       text.setText("");
60.  }
61.}
```

ثالثاً: شرح الكود:

- هذا المثال به خطأ مقصود سنوضحه في الشرح ولذلك فلن يمكنك تنفيذ هذا البرنامج وسنشرح كيفية معالجة هذا الخطأ لاحقاً.
- في السطور من 1 إلي 3 يتم تضمين الحزم packages اللازمة لعمل البرنامج.
- في السطر رقم 5 يتم إنشاء الفصيلة ButtonEventUsingSeparateClass2 التي ترث من الفصيلة JFrame حتي يمكنها إنشاء شاشة البرنامج كما شرحنا سابقاً.
- في السطور من 7 إلي 11 يتم إنشاء جميع المتغيرات التي نحتاجها في البرنامج.
- في السطر رقم 13 يتم إنشاء دالة البناء Constructor.
- في السطر رقم 17 يتم إنشاء هدف Object من نوع JButton والذي يمثل أداة العنوان JButton.
- في السطر رقم 19 يتم إنشاء هدف object من نوع الفصيلة ButtonListener والمسئولة عن معالجة حدث الضغط علي الزر JButton Events كما سنري لاحقاً.

- في السطر رقم 20 يتم إضافة مستمع الحدث Event Listener عن طريق الدالة addActionListener( ).
- في السطور من 49 إلي 52 يتم إنشاء الدالة الرئيسية main() التي يتم تنفيذها عند تنفيذ البرنامج وفيها يتم إنشاء هدف Object من نوع الفصيلة ButtonEventUsingSeparateClass2 وينتج عن ذلك استدعاء دالة البناء Constructor وإظهار نافذة البرنامج.
- فى السطور من 55 إلى 61 نجد أننا أنشأنا تعريف الفصيلة ButtonListener التى تنفذ الـ interface المسمي ActionListener. ونلاحظ وجود تعريف الدالة المسماة actionPerformed() التي تستقبل هدفاً object من نوع الفصيلة ActionEvent الذى يمثل حدث الضغط علي الزر.
- فى هذه الدالة Method وضعنا رد الفعل الذى نريده وهو أن نجعل النص خالياً في أداة النص JTextField ، ويتم ذلك عن طريق الدالة setText().
- الخطأ حدث هنا في السطر رقم 59 حيث أننا قمنا باستدعاء المتغير text والمعرف في السطر رقم 8 وهو يمثل أداة النص JTextField ، والخطأ في استدعاء هذا المتغير أنه معرف في الفصيلة class المسماة ButtonEventUsingSeparateClass2 كمتغير عضو Memeber Variable ولذلك لا يمكن استدعاءه مباشرة من داخل الفصيلة class المسماة ButtonListener ، ولذلك سنقوم بشرح العديد من الطرق لتمكين استدعاء هذا المتغير كما سيتضح لنا من النقاط التالية.
- تذكر أنه لن يمكنك ترجمة هذا البرنامج للخطأ الموجود به والذي سنعالجه في النقاط والأمثلة القادمة.

الاستدعاء الرجعي Callback:

- عملية الاستدعاء الرجعي Callback هي عملية تمكننا من إجراء اتصال بين الأهداف objects المختلفة وهو ما يعرف باسم Interobject Communication ، وهذه العملية يتم تنفيذها عن طريق دالة البناء constructor.
- فمثلاً في المثال السابق احتجنا أن نستدعي المتغير text – والمعرف في الفصيلة

ButtonEventUsingSeparateClass2– من داخل الفصيلة ButtonListener ، ولكي نستطيع تنفيذ هذا الاستدعاء فإننا نمرر إلي الفصيلة ButtonListener هدفاً object من نوع الفصيلة ButtonEventUsingSeparateClass2 ، وباستخدام هذا الهدف object يمكننا استدعاء جميع متغيرات ودوال الفصيلة ButtonEventUsingSeparateClass2.

- السؤال الآن: كيف نمرر هدفاً object من نوع الفصيلة ButtonEventUsingSeparateClass2 إلي الفصيلة ButtonListener؟
- يتم تنفيذ ذلك عن طريق دالة البناء constructor للفصيلة ButtonListener حيث سيتم تعديلها بحيث تستقبل هدفاً object من نوع الفصيلة ButtonEventUsingSeparateClass2 أي سيصبح تعريفها كالتالي:

ButtonListener (ButtonEventUsingSeparateClass2 b3)

- ثم نقوم بتعريف متغير بداخل الفصيلة ButtonListener ويكون من نوع الفصيلة ButtonEventUsingSeparateClass2 بحيث يمكننا استدعاء هذا المتغير من أي مكان في الفصيلة ButtonListener لأنه سيكون متغير عام Global Variable وليس متغيراً محلياً Local Variable.
- إذن عندنا متغير من نوع الفصيلة ButtonEventUsingSeparateClass2 معرف بداخل الفصيلة ButtonListener ومتغير أخر من نوع الفصيلة ButtonEventUsingSeparateClass2 أيضاً تم تمريره كمعامل Parameter لدالة البناء constructor للفصيلة ButtonListener ونريد أن نساوي القيمتين ببعضهما البعض ويتم ذلك في دالة البناء constructor للفصيلة ButtonListener أي كالتالي:

```
class ButtonListener implements ActionListener
{
   ButtonEventUsingSeparateClass2 b3;
```

```
    ButtonListener(ButtonEventUsingSeparateClass2 b3)
    {
        this.b3 = b3;
    }

    //rest of class code
}
```

- لاحظ استخدام this بسبب استخدام نفس اسم المتغير.
- تتبقي خطوة واحدة وهي أنه عند إنشاء هدف object من نوع الفصيلة class المسماة ButtonListener ، فإننا نمرر لدالة البناء construcot هدفاً object من نوع الفصيلة class المسماة ButtonEventUsingSeparateClass2.

المثال التالي يوضح لنا هذه النقطة.

مثال (3): معالجة حدث الضغط علي الزر باستخدام الاستدعاء الرجعي Callback:

أولاً: هدف المثال:

نريد أن نقوم بتعديل المثال السابق لتوضيح عملية الاستدعاء الرجعي Callback وإصلاح الخطأ السابق.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class ButtonEventUsingCallBack extends JFrame
6.  {
7.    JButton reset;
8.    JTextField text;
9.    JLabel name;
10.   GridBagLayout grid;
```

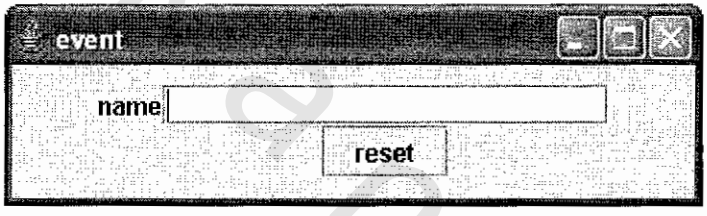
```
11. GridBagConstraints gbc;
12.
13. ButtonEventUsingCallBack()
14. {
15.     Container c = getContentPane();
16.
17.     reset = new JButton("reset");
18.     ButtonListener b = new ButtonListener(this);
19.     reset.addActionListener(b);
20.     text = new JTextField(20);
21.     name = new JLabel("name");
22.
23.     grid = new GridBagLayout();
24.     gbc = new GridBagConstraints();
25.     c.setLayout(grid);
26.
27.     gbc.gridx = 1;
28.     gbc.gridy = 1;
29.     grid.setConstraints(name,gbc);
30.     c.add(name);
31.
32.     gbc.gridx = 3;
33.     gbc.gridy = 1;
34.     grid.setConstraints(text,gbc);
35.     c.add(text);
36.
37.     gbc.gridx = 3;
38.     gbc.gridy = 3;
39.     grid.setConstraints(reset,gbc);
40.     c.add(reset);
41.
42.     setTitle("event");
43.     setSize(350,100);
44.     setVisible(true);
```

```
45.  }
46.
47.  public static void main(String args[])
48.  {
49.       ButtonEventUsingCallBack event = new
     ButtonEventUsingCallBack();
50.  }
51.}
52.
53.class ButtonListener implements ActionListener
54.{
55.  ButtonEventUsingCallBack b3;
56.
57.  public void actionPerformed(ActionEvent e)
58.  {
59.       b3.text.setText("");
60.  }
61.
62.  ButtonListener(ButtonEventUsingCallBack b3)
63.  {
64.       this.b3 = b3;
65.  }
66.}
```

ثالثاً: شرح الكود:

- هذا المثال هو عبارة عن تعديل للمثال السابق ولذلك فلن نعيد شرح المثال كاملاً وسنكتفي بشرح التعديلات فقط.
- في السطر رقم 18 يتم إنشاء هدف object من نوع الفصيلة ButtonListener وتمرير القيمة this لدالة البناء constructor ، وكما تعلم فإن this تشير إلي الهدف object الحالي أي أن this تشير إلي هدف object من نوع الفصيلة ButtonEventUsingCallBack.

- في السطر رقم 55 يتم تعريف متغير من نوع الفصيلة ButtonEventUsingCallBack.
- في السطر رقم 62 يتم تعريف دالة البناء constructor وهي تستقبل معاملاً Parameter من نوع الفصيلة ButtonEventUsingCallBack ويتم استخدام هذه القيمة لتحديد قيمة المتغير b3 والمعرف في السطر رقم 55.
- في السطر رقم 59 يتم استدعاء المتغير text عن طريق المتغير b3 والذي من نوع الفصيلة ButtonEventUsingCallBack وهنا تصبح عملية استدعاء أي متغير موجود في الفصيلة ButtonEventUsingCallBack ممكنة.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-2) ويمكنك تجربة كتابة أي نص ثم الضغط علي الزر للتأكد من مسح النص.



الشكل (17-2) حدث الزر JButton

- واضح أن عملية الاستدعاء الرجعي Callback تسبب تغييراً في الكود وتسبب أيضاً زيادة عدد سطور البرنامج ، ولذلك يفضل استخدام الطريقة الأخري وهي طريقة الفصائل الداخلية Inner Classes.

الفصائل الداخلية Inner Classes:

- الفصائل الداخلية Inner Classes هي الفصائل classes التي يتم تعريفها بداخل فصيلة class أخري.
- كمثال من الحياة العملية ، فإن شقتك تتكون من حجرة معيشة ومطبخ ودورة مياة مثلاً ، إذن فهناك فصيلة كبيرة هي الشقة وهي تحتوي علي عدة فصائل داخلية وهي حجرة المعيشة والمطبخ ودورة المياة ، ولذلك يمكننا كتابة الكود التالي لوصف الشقة:

```
class Apartment
{
    class LivingRoom
    {
        //code
    }

    class BathRoom
    {
        //code
    }

    class Kitchen
    {
        //code
    }

    //code
}
```

- وتلعب الفصائل الداخلية Inner Classes دوراً في غاية الأهمية في معالجة الحدث Event Handling حيث يمكننا إنشاء فصيلة داخلية Inner Class موجودة داخل الفصيلة class المسئولة عن إنشاء نافذة البرنامج (والتي تحتوي بالطبع علي جميع الأدوات التي نريد معالجة أحداثها Event Handling) بحيث يمكننا استدعاء أي متغير من الفصيلة class الكبيرة التي تحتوي هذه الفصيلة الداخلية Inner Class مباشرة دون الحاجة إلي التعديل في دالة البناء constructor وهذا ما سيتضح لنا في المثال التالي:

مثال (4): معالجة حدث الضغط علي الزر باستخدام الفصائل الداخلية Inner Classes:

أولاً: هدف المثال:

نريد أن نعدل المثال السابق لتوضيح كيفية استخدام الفصائل الداخلية Inner Classes لمعالجة الحدث Event Handling بطريقة أفضل.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class ButtonEventUsingInnerClass extends JFrame
6. {
7.  JButton reset;
8.  JTextField text;
9.  JLabel name;
10. GridBagLayout grid;
11. GridBagConstraints gbc;
12.
13. ButtonEventUsingInnerClass()
14. {
15.      Container c = getContentPane();
16.
17.      reset = new JButton("reset");
18.      ButtonListener b = new ButtonListener();
19.      reset.addActionListener(b);
20.      text = new JTextField(20);
21.      name = new JLabel("name");
22.
23.      grid = new GridBagLayout();
24.      gbc = new GridBagConstraints();
25.      c.setLayout(grid);
26.
27.      gbc.gridx = 1;
28.      gbc.gridy = 1;
29.      grid.setConstraints(name,gbc);
30.      c.add(name);
31.
32.      gbc.gridx = 3;
```

```
33.         gbc.gridy = 1;
34.         grid.setConstraints(text,gbc);
35.         c.add(text);
36.
37.         gbc.gridx = 3;
38.         gbc.gridy = 3;
39.         grid.setConstraints(reset,gbc);
40.         c.add(reset);
41.
42.         setTitle("event");
43.         setSize(350,100);
44.         setVisible(true);
45.   }
46.
47.   public static void main(String args[])
48.   {
49.         ButtonEventUsingInnerClass event = new
      ButtonEventUsingInnerClass();
50.   }
51.
52.   class ButtonListener implements ActionListener
53.   {
54.         public void actionPerformed(ActionEvent e)
55.         {
56.               text.setText("");
57.         }
58.   }
59.}
```

ثالثاً: شرح الكود:

- هذا المثال هو عبارة عن تعديل للمثال رقم 2 ولذلك فلن نعيد شرح المثال كاملاً وسنكتفي بشرح التعديلات فقط.
- بمقارنة هذا المثال مع المثال رقم 2 السابق ، يتضح لنا أن التعديل الذي حدث هو

تعديل بسيط جداً ، فقد تم تغيير مكان الفصيلة class المسماة ButtonListener لتصبح بالكامل موجودة بداخل الفصيلة ButtonEventUsingInnerClass.

- لاحظ أن قوس الفصيلة ButtonEventUsingInnerClass بدأ في السطر رقم 6 وانتهي في السطر رقم 59 ، أي أن الفصيلة ButtonEventUsingInnerClass تحتوي علي كود الفصيلة ButtonListener بالكامل -والمعرفة في السطور من 52 إلي 58- وهذا هو التعديل الوحيد فقط ، ومن هنا يتضح لنا مدي سهولة استخدام الفصائل الداخلية Inner Classes لمعالجة الحدث Event Handling.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (2-17) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

معالجة حدث أكثر من زر JButton:

تعرفنا في الأمثلة السابقة علي كيفية معالجة حدث الضغط علي الزر JButton ، ولكننا لم نتعرف علي كيفية التصرف في حالة وجود أكثر من زر JButton في البرنامج ، وهذا ما سنتعرف عليه في النقاط التالية.

معالجة حدث أكثر من زر JButton باستخدام فصائل Classes متعددة:

تعتمد هذه الطريقة علي إنشاء فصيلة class منفصلة لكل زر JButton موجود في نافذة البرنامج ، وتتم كتابة كود البرمجة لكل زر في الفصيلة class الخاصة به.

المثال التالي يوضح هذه النقطة.

مثال (5): معالجة حدث أكثر من زر JButton باستخدام فصائل Classes متعددة:

أولاً: هدف المثال:

فى هذا المثال نريد أن نقوم بتعديل المثال السابق بحيث نضيف زراً أخر وظيفته وضع النص "Java is excellent" فى أداة النص JTextField عند الضغط عليه.

ثانياً: كود البرمجة:

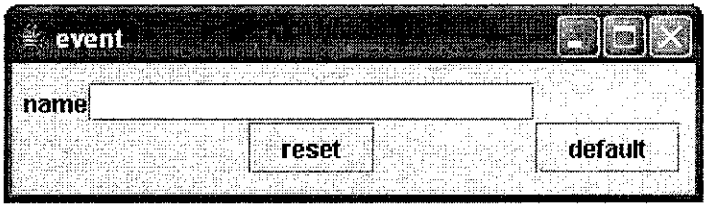
```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class ButtonEventUsingMultipleClasses extends
   JFrame
6. {
7.   JButton reset,defaultButton;
8.   JTextField text;
9.   JLabel name;
10. GridBagLayout grid;
11. GridBagConstraints gbc;
12.
13. ButtonEventUsingMultipleClasses()
14. {
15.       Container c = getContentPane();
16.
17.       reset = new JButton("reset");
18.       ButtonListener b = new ButtonListener();
19.       reset.addActionListener(b);
20.       defaultButton = new JButton("default");
21.       DefaultButton gg = new DefaultButton();
22.       defaultButton.addActionListener(gg);
23.       text = new JTextField(20);
24.       name = new JLabel("name");
25.
26.       grid = new GridBagLayout();
27.       gbc = new GridBagConstraints();
28.       c.setLayout(grid);
29.
30.       gbc.gridx = 1;
31.       gbc.gridy = 1;
32.       grid.setConstraints(name,gbc);
33.       c.add(name);
```

```
34.
35.         gbc.gridx = 3;
36.         gbc.gridy = 1;
37.         grid.setConstraints(text,gbc);
38.         c.add(text);
39.
40.         gbc.gridx = 3;
41.         gbc.gridy = 3;
42.         grid.setConstraints(reset,gbc);
43.         c.add(reset);
44.
45.         gbc.gridx = 4;
46.         gbc.gridy = 3;
47.         grid.setConstraints(defaultButton,gbc);
48.         c.add(defaultButton);
49.
50.         setTitle("event");
51.         setSize(350,100);
52.         setVisible(true);
53.  }
54.
55.  public static void main(String args[])
56.  {
57.         ButtonEventUsingMultipleClasses event = new
     ButtonEventUsingMultipleClasses();
58.  }
59.
60.  class ButtonListener implements ActionListener
61.  {
62.         public void actionPerformed(ActionEvent e)
63.         {
64.              text.setText("");
65.              text.requestFocus();
66.         }
67.  }
68.
```

```
69. class DefaultButton implements ActionListener
70. {
71.     public void actionPerformed(ActionEvent e)
72.     {
73.         text.setText("Java is excellent");
74.         text.selectAll();
75.         text.requestFocus();
76.     }
77. }
78.}
```

ثالثاً: شرح الكود:

- في السطر رقم 7 و 20 يتم إنشاء الزر الجديد ونسميه defaultButton.
- في السطر رقم 21 يتم إنشاء هدف object من نوع الفصيلة DefaultButton حيث نستخدم هذا الهدف object كمستمع لحدث الضغط علي الزر.
- في السطر رقم 22 يتم إضافة مستمع الحدث Event Listener عن طريق الدالة addActionListener( ).
- في السطور من 69 إلى 77 يتم تعريف الفصيلة DefaultButton التي بالطبع تنفذ الـ interface المسمي ActionListener والذي يحتوي علي الدالة actionPerformed() التي تستقبل هدفاً object من نوع الفصيلة ActionEvent ، وفيها يتم تغيير النص في أداة النص JTextField إلي النص Java is excellent كما يتم تظليل Select النص بالكامل حتي يستطيع المستخدم أن يقوم بالكتابة مباشرة في أداة النص JTextField دون الحاجة إلي مسح النص المكتوب ، ويتم ذلك عن طريق الدالة selectAll( ). كما يتم وضع مؤشر الفارة Mouse في أداة النص JTextField عن طريق الدالة requestFocus( ) وهذه العملية معروفة باسم Focusing.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (3-17) ويمكنك تجربة الضغط علي الزر default للتأكد من كتابة النص Java is excellent في أداة النص JTextField.



الشكل (17-3) حدث الزر JButton

- ولكن لنفترض أنه يوجد عدد كبير من الأزرار في نافذة البرنامج ، ففي هذه الحالة نجد أن عملية إنشاء فصيلة class لكل زر عملية طويلة ومعقدة وغير عملية وتسبب تخبطاً بين الفصائل classes الموجودة. إذن ما الحل؟
- الحل هو كتابة فصيلة class واحدة فقط لكل الأزرار كما يلي في النقطة التالية.

معالجة حدث أكثر من زر JButton باستخدام فصيلة Class واحدة:

تعتمد هذه الطريقة علي إنشاء فصيلة class واحدة لجميع الأزرار JButton الموجودة في نافذة البرنامج ، وبالتالي يظهر سؤال هو كيف يتم التفريق بين كل زر وأخر؟ يتم ذلك عن طريق الدالة ()getSource.

المثال التالي يوضح هذه النقطة.

مثال (6): معالجة حدث أكثر من زر JButton باستخدام فصيلة Class واحدة:

أولاً: هدف المثال:

توضيح كيفية إنشاء فصيلة class واحدة لمعالجة حدث الضغط علي أكثر من زر.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class ButtonEventUsingOneClass extends JFrame
6. {
7.   JButton reset,defaultbutton;
8.   JTextField text;
```

```
9.  JLabel name;
10. GridBagLayout grid;
11. GridBagConstraints gbc;
12.
13. ButtonEventUsingOneClass()
14. {
15.     Container c = getContentPane();
16.
17.     reset = new JButton("reset");
18.     ButtonListener b = new ButtonListener();
19.     reset.addActionListener(b);
20.     defaultbutton = new JButton("default");
21.     defaultbutton.addActionListener(b);
22.
23.     text = new JTextField(20);
24.     name = new JLabel("name");
25.
26.     grid = new GridBagLayout();
27.     gbc = new GridBagConstraints();
28.     c.setLayout(grid);
29.
30.     gbc.gridx = 1;
31.     gbc.gridy = 1;
32.     grid.setConstraints(name,gbc);
33.     c.add(name);
34.
35.     gbc.gridx = 3;
36.     gbc.gridy = 1;
37.     grid.setConstraints(text,gbc);
38.     c.add(text);
39.
40.     gbc.gridx = 3;
41.     gbc.gridy = 3;
42.     grid.setConstraints(reset,gbc);
```

```
43.         c.add(reset);
44.
45.         gbc.gridx = 4;
46.         gbc.gridy = 3;
47.         grid.setConstraints(defaultbutton,gbc);
48.         c.add(defaultbutton);
49.
50.         setTitle("event");
51.         setSize(350,100);
52.         setVisible(true);
53.  }
54.
55.  public static void main(String args[])
56.  {
57.         ButtonEventUsingOneClass event = new
     ButtonEventUsingOneClass();
58.  }
59.
60.  class ButtonListener implements ActionListener
61.  {
62.         public void actionPerformed(ActionEvent e)
63.         {
64.              Object obj = e.getSource();
65.
66.              if ( obj == reset )
67.              {
68.                   text.setText("");
69.                   text.requestFocus();
70.              }
71.
72.              else if ( obj == defaultbutton )
73.              {
74.                   text.setText("Java is excellent");
75.                   text.selectAll();
```

```
76.                 text.requestFocus();
77.             }
78.         }
79.  }
80.}
```

ثالثاً: شرح الكود:

- في السطر رقم 18 يتم إنشاء هدف object من نوع الفصيلة ButtonListener حيث نستخدم هذا الهدف object كمستمع لحدث الضغط علي الزر.
- في السطرين 19 و 21 يتم إضافة مستمع الحدث Event Listener للزرين defaultbutton, reset معاً عن طريق الدالة ( )addActionListener.
- فى بناء الفصيلة ButtonListener كما في السطور من 60 إلى 79 نجد أننا في الدالة actionPerformed() أنشأنا هدفاً object اسمه obj من نوع الفصيلة Object – وهى الفصيلة class الأم لجميع الفصائل classes الموجودة فى لغة Java – واستخدمنا الدالة getSource() التي نستطيع من خلالها أن نعرف مصدر الحدث Event Source ، أي نستطيع معرفة الزر الذى أطلق الحدث Event والذي ضغط عليه المستخدم.
- في السطور من 66 إلى 77 نختبر الهدف obj ، فإذا كان هو الزر reset فإننا ننفذ كود البرمجة السابق شرحه والذي يقوم بمسح النص ، وإذا كان هو الزر defaultButton فإننا ننفذ كود البرمجة السابق شرحه والذي يقوم بكتابة النص Java is excellent.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-3) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

معالجة حدث أكثر من زر JButton باستخدام الفصائل الداخلية اللااسمية Inner Anonymous Classes:

تعتمد هذه الطريقة علي إنشاء فصيلة class داخلية لكل زر JButton من الأزرار الموجودة في نافذة البرنامج ولكن يتم كتابة كود الفصيلة class التي تعالج الحدث

Event بداخل الزر نفسه ودون تسمية الفصيلة class بأي اسم (ومن هنا جاء اسم لااسمية) ، وبالتالي تعتبر هذه الطريقة وسطاً بين الطريقتين السابقتين.

المثال التالي يوضح هذه النقطة.

مثال (7): معالجة حدث أكثر من زر JButton باستخدام الفصائل الداخلية اللااسمية Inner Anonymous Classes:

أولاً: هدف المثال:

توضيح كيفية إنشاء فصيلة داخلية لااسمية inner anonymous class لمعالجة حدث الضغط علي أكثر من زر.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class ButtonEventUsingInnerAnonymousClasses
    extends JFrame
6.  {
7.   JButton reset,defaultbutton;
8.   JTextField text;
9.   JLabel name;
10.  GridBagLayout grid;
11.  GridBagConstraints gbc;
12.
13.  ButtonEventUsingInnerAnonymousClasses()
14.  {
15.      Container c = getContentPane();
16.      reset = new JButton("reset");
17.
18.      reset.addActionListener(new ActionListener()
19.          {
20.              public void actionPerformed(ActionEvent e)
```

```
21.                 {
22.                     text.setText("");
23.                     text.requestFocus();
24.                 }
25.             }
26.         );
27.
28.         defaultbutton = new JButton("default");
29.
30.         defaultbutton.addActionListener(new
    ActionListener()
31.             {
32.                 public void actionPerformed(ActionEvent
    e)
33.                 {
34.                     text.setText("Java is excellent");
35.                     text.selectAll();
36.                     text.requestFocus();
37.                 }
38.             }
39.         );
40.
41.         text = new JTextField(20);
42.         name = new JLabel("name");
43.
44.         grid = new GridBagLayout();
45.         gbc = new GridBagConstraints();
46.         c.setLayout(grid);
47.
48.         gbc.gridx = 1;
49.         gbc.gridy = 1;
50.         grid.setConstraints(name,gbc);
51.         c.add(name);
52.
53.         gbc.gridx = 3;
```

```
54.         gbc.gridy = 1;
55.         grid.setConstraints(text,gbc);
56.         c.add(text);
57.
58.         gbc.gridx = 3;
59.         gbc.gridy = 3;
60.         grid.setConstraints(reset,gbc);
61.         c.add(reset);
62.
63.         gbc.gridx = 4;
64.         gbc.gridy = 3;
65.         grid.setConstraints(defaultbutton,gbc);
66.         c.add(defaultbutton);
67.
68.         setTitle("event");
69.         setSize(350,100);
70.         setVisible(true);
71.   }
72.
73.   public static void main(String args[])
74.   {
75.         ButtonEventUsingInnerAnonymousClasses event =
      new ButtonEventUsingInnerAnonymousClasses();
76.   }
77.}
```

ثالثاً: شرح الكود:

- في السطر رقم 18 يتم إنشاء مستمع لحدث الضغط علي الزر reset عن طريق الدالة addActionListener( ) مع ملاحظة أنه تم كتابة كود معالجة الحدث داخل معامل الدالة addActionListener( ) وبدون كتابة اسم للفصيلة class Parameter التي تعالج الحدث Event.
- قد تكون طريقة الكتابة غريبة ولكنها طريقة صحيحة تماماً ومستخدمة كثيراً في لغة Java وستعتاد عليها.

- بالمثل يتم إنشاء مستمع لحدث الضغط علي الزر defaultbutton في السطر رقم 30.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (3-17) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

ملخص معالجة الحدث Event Handling:

يمكننا معالجة الحدث Event Handling بثلاث طرق مختلفة وهي:

1. إنشاء فصيلة class مستقلة لمعالجة الحدث Event Handling. نستطيع استخدام هذه الطريقة إذا لم نكن بحاجة إلي استدعاء أي متغير أو دالة Method من فصيلة class أخري.
2. استخدام الاستدعاء الرجعي Callback. نستطيع استخدام هذه الطريقة إذا كنا بحاجة إلي استدعاء أي متغير أو دالة Method من فصيلة class أخري ، ولكن عيوب هذه الطريقة أنها تحتاج منك إلي إجراء عدة تغييرات في الكود وتسبب زيادة عدد سطور البرمجة.
3. استخدام الفصائل الداخلية Inner Classes أو الفصائل الداخلية اللاسمية Inner Anonymous Classes والتي تمكننا من استدعاء أي متغير أو دالة Method من الفصيلة class الكبري التي تحتويها ، وتتميز هذه الطريقة بعدم زيادة سطور البرمجة وبعدم إجراء أي تغيير إلا وضع الفصيلة class التي تعالج الحدث Event بداخل الفصيلة class الحاوية.

نستنتج من هنا أن استخدام الفصائل الداخلية Inner Classes لمعالجة الحدث Event Handling تعتبر أفضل طريقة ، ولكن هذا لا يمنع أن عملية الاستدعاء الرجعي Callback من أهم العمليات في البرمجة.

أحداث أداة الإطار JFrame Events:

تتوافر الأحداث التالية لأداة الإطار JFrame:

- حدث إغلاق النافذة.
- حدث إخفاء النافذة عن طريق الدالة setVisible (false) أو dispose().

- حدث فتح النافذة لأول مرة.
- حدث الضغط على مفتاح التصغير للنافذة (Minimize).
- حدث تكبير النافذة Restore بعد تصغيرها.
- حدث تنشيط Activate الإطار Frame.
- حدث عدم تنشيط Deactivate الإطار Frame.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء فصيلة class ترث من أداة الإطار JFrame بحيث أن النافذة الجديدة هي التي نريد معالجة أحداثها.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي WindowListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
windowClosing()	تنفذ هذه الدالة Method عند إغلاق النافذة
WindowClosed()	تنفذ هذه الدالة عند إخفاء النافذة Method عن طريق الدالة setVisible (false) أو dispose()
windowOpened()	تنفذ هذه الدالة Method عند فتح النافذة لأول مرة
windowIconified()	تنفذ هذه الدالة Method عند الضغط على مفتاح التصغير للنافذة (Minimize)
windowDeiconified()	تنفذ هذه الدالة Method عند تكبير النافذة Restore بعد تصغيرها
windowActivated()	تنفذ هذه الدالة Method عندما يصبح الإطار Frame هو النشط Active
windowDeactivated()	تنفذ هذه الدالة Method عندما يصبح الإطار Frame غير نشط

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي أداة الإطار JFrame عن طريق الدالة ( )addWindowListener.

ملحوظة:

يجب أن تعيد تعريف جميع هذه الدوال Methods حتى لو لم تكتب فيها شيئاً لأن كلها دوال مجردة abstract methods ولذلك إذا نسيت كتابة تعريف دالة Method أو أخطأت فى كتابة الدالة Method بطريقة صحيحة - مع ملاحظة الحرف الصغير والكبير - ، فإنك تحصل على رسالة خطأ عند ترجمة Compile البرنامج لأنه فى هذه الحالة تكون الفصيلة class التى أنشأتها -باسم MyWindowListener مثلاً- فصيلة مجردة abstract class ولا يمكن أن تنشئ منها هدفاً object ، وستتعرف في الأمثلة القادمة علي كيفية التغلب علي هذه المشكلة.

المثال التالي يوضح النقاط السابقة.

مثال (8): معالجة حدث أداة الإطار: Handling JFrame Event:

أولاً: هدف المثال:

نريد تعديل المثال السابق بحيث أنه عند إغلاق نافذة التطبيق فإن ذلك يغلق أيضاً Java Virtual Machine (JVM) أي يتم إنهاء البرنامج تماماً.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class WindowEventUsingListener extends JFrame
6. {
7.  JButton reset,defaultbutton;
```

```
8.  JTextField text;
9.  JLabel name;
10. GridBagLayout grid;
11. GridBagConstraints gbc;
12.
13. WindowEventUsingListener()
14. {
15.     MyWindowListener w = new MyWindowListener();
16.     addWindowListener(w);
17.
18.     Container c = getContentPane();
19.
20.     reset = new JButton("reset");
21.     ButtonListener b = new ButtonListener();
22.     reset.addActionListener(b);
23.     defaultbutton = new JButton("default");
24.     defaultbutton.addActionListener(b);
25.
26.     text = new JTextField(20);
27.     name = new JLabel("name");
28.
29.     grid = new GridBagLayout();
30.     gbc = new GridBagConstraints();
31.     c.setLayout(grid);
32.
33.     gbc.gridx = 1;
34.     gbc.gridy = 1;
35.     grid.setConstraints(name,gbc);
36.     c.add(name);
37.
38.     gbc.gridx = 3;
39.     gbc.gridy = 1;
40.     grid.setConstraints(text,gbc);
41.     c.add(text);
```

```
42.
43.         gbc.gridx = 3;
44.         gbc.gridy = 3;
45.         grid.setConstraints(reset,gbc);
46.         c.add(reset);
47.
48.         gbc.gridx = 4;
49.         gbc.gridy = 3;
50.         grid.setConstraints(defaultbutton,gbc);
51.         c.add(defaultbutton);
52.
53.         setTitle("event");
54.         setSize(350,100);
55.         setVisible(true);
56.  }
57.
58.  public static void main(String args[])
59.  {
60.         WindowEventUsingListener event = new
     WindowEventUsingListener();
61.  }
62.
63.  class ButtonListener implements ActionListener
64.  {
65.         public void actionPerformed(ActionEvent e)
66.         {
67.             Object obj = e.getSource();
68.
69.             if ( obj == reset )
70.             {
71.                 text.setText("");
72.                 text.requestFocus();
73.             }
74.
```

```
75.             else if ( obj == defaultbutton )
76.             {
77.                 text.setText("Java is excellent");
78.                 text.selectAll();
79.                 text.requestFocus();
80.             }
81.         }
82.     }
83.
84.     class MyWindowListener implements WindowListener
85.     {
86.         public void windowActivated(WindowEvent e)
87.         {
88.             System.out.println("activated");
89.         }
90.
91.         public void windowClosed(WindowEvent e)
92.         {
93.             System.out.println("byebye");
94.         }
95.
96.         public void windowClosing(WindowEvent e)
97.         {
98.             System.out.println("closing");
99.             dispose();
100.            }
101.
102.            public void windowDeactivated(WindowEvent e)
103.            {
104.                System.out.println("deactivated");
105.            }
106.
107.            public void windowIconified(WindowEvent e)
108.            {
```

```
109.            System.out.println("minimized");
110.        }
111.
112.        public void windowDeiconified(WindowEvent e)
113.        {
114.            System.out.println("maximized");
115.        }
116.
117.        public void windowOpened(WindowEvent e)
118.        {
119.            System.out.println("WELCOME");

120.        }
121.    }
122.}
```

ثالثاً: شرح الكود:

- في السطر رقم 15 يتم إنشاء هدف object من نوع الفصيلة MyWindowListener حيث نستخدم هذا الهدف object كمستمع لحدث الضغط علي الزر.
- في السطر رقم 16 يتم إضافة مستمع الحدث Event Listener لأداة الإطار JFrame عن طريق الدالة ( )addWindowListener.
- فى بناء الفصيلة MyWindowListener -كما في السطور من 84 إلى 121- نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث أداة الإطار JFrame بحيث أن جميع الدوال Methods تطبع على شاشة الدوس Dos كلمة تفيد الحدث الذى أطلقته النافذة ، وقمنا في الدالة ()windowClosing باستدعاء الدالة ( )dispose التي تتولي إغلاق نافذة البرنامج وإخلاء الذاكرة منها.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-3) السابق وتأكد من عمل البرنامج بالشكل المطلوب.
- لاحظ أنه عند فتح البرنامج لأول مرة ستلاحظ طباعة كلمة Activated وذلك لأن

النافذة أصبحت نشطة Active بالإضافة إلي كلمة Opened وذلك لأن النافذة تم فتحها لأول مرة.

- الآن قم بعمل تصغير Minimize للنافذة وستلاحظ طباعة كلمة Iconified وذلك لأنك صغرت النافذة بالإضافة إلي كلمة Dectivated وذلك لأن النافذة أصبحت غير نشطة Deactivated.
- الآن اضغط على زر تكبير النافذة من شريط المهام (Task Bar) وستلاحظ طباعة كلمة Deiconified وذلك لأنك كبرت النافذة بالإضافة إلي كلمة Activated وذلك لأن النافذة أصبحت نشطة Active.
- الآن اضغط على زر إغلاق النافذة ولاحظ طباعة كلمة Closing وذلك لأنك أغلقت النافذة بالإضافة إلي كلمة Closed وذلك لأنك أخفيت النافذة عن طريق الدالة ()dispose.
- لاحظ أنه يمكنك إغلاق البرنامج تماماً عن طريق الدالة exit(0) حيث أن الدالة ( )dispose تقوم فقط بإغلاق النافذة وليس البرنامج ككل.

معالجة أحداث الإطار JFrame Events عن طريق Adapter:

- عند كتابة الفصيلة MyWindowListener في المثال السابق ، فلا بد أنك لاحظت أنها تنفذ الـ interface المسمي WindowListener ، ولذلك فلا بد من إعادة كتابة تعريف الدوال Methods السبعة برغم أنه قد تكون بعض الدوال Methods بلا فائدة بالنسبة إليك وتريد فقط كتابة تعريف الدالة Method التى تريدها ، وفى هذه الحالة فإنك تستخدم ما يسمي بـ WindowAdapter.
- إن WindowAdapter ببساطة هو عبارة عن فصيلة class تقوم بتنفيذ الـ interface المسمي WindowListener عن طريق تنفيذ الدوال Methods السبعة السابق شرحهم ، وبالتالي يمكنك الوراثة من هذه الفصيلة class وتنفيذ الدوال Methods التي تحتاجها فقط دون الحاجة إلي تنفيذ الدوال Methods السبعة ، ولكن يعيبه أنه إذا كانت الفصيلة class التي تنشئها تقوم بالوراثة من

فصيلة class أخري ، فعندئذ لا يمكنك استخدام WindowAdapter لأن لغة Java - كما تعلم - لا تدعم خاصية الوراثة المتعددة Multiple Inheritance.

الجدول التالي يقوم بعقد مقارنة بين WindowListener و WindowAdapter.

وجه المقارنة	WindowListener	WindowAdapter
الاستخدام	يجب تنفيذه implement	يجب الوراثة Inheritance منه
تعريف الدوال Methods	يجب تعريف جميع الدوال Methods	يكفي أن نقوم بتعريف الدوال Methods التي نحتاجها فقط
قيود الاستخدام	لا توجد	إذا كانت الفصيلة class التي تنشئها تقوم بالوراثة من فصيلة class أخري ، فعندئذ لا يمكنك استخدام WindowAdapter

مثال (9): معالجة أحداث الإطار JFrame Events عن طريق Adapter:

أولاً: هدف المثال:

توضيح كيفية معالجة أحداث أداة الإطار JFrame عن طريق WindowAdapter بدلاً من WindowListener وذلك لعدم الحاجة إلي كتابة تعريف الدوال السبعة السابق شرحهم خاصة أن بعض الدوال Methods قد نكون في غير حاجة إليها في تطبيقنا.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class WindowEventUsingAdapter extends JFrame
6. {
7.   JButton reset,defaultbutton;
8.   JTextField text;
```

```
9.  JLabel name;
10. GridBagLayout grid;
11. GridBagConstraints gbc;
12.
13. WindowEventUsingAdapter()
14. {
15.     MyWindowListener w = new MyWindowListener();
16.     addWindowListener(w);
17.
18.     Container c = getContentPane();
19.
20.     reset = new JButton("reset");
21.     ButtonListener b = new ButtonListener();
22.     reset.addActionListener(b);
23.     defaultbutton = new JButton("default");
24.     defaultbutton.addActionListener(b);
25.
26.     text = new JTextField(20);
27.     name = new JLabel("name");
28.
29.     grid = new GridBagLayout();
30.     gbc = new GridBagConstraints();
31.     c.setLayout(grid);
32.
33.     gbc.gridx = 1;
34.     gbc.gridy = 1;
35.     grid.setConstraints(name,gbc);
36.     c.add(name);
37.
38.     gbc.gridx = 3;
39.     gbc.gridy = 1;
40.     grid.setConstraints(text,gbc);
41.     c.add(text);
42.
```

```
43.        gbc.gridx = 3;
44.        gbc.gridy = 3;
45.        grid.setConstraints(reset,gbc);
46.        c.add(reset);
47.
48.        gbc.gridx = 4;
49.        gbc.gridy = 3;
50.        grid.setConstraints(defaultbutton,gbc);
51.        c.add(defaultbutton);
52.
53.        setTitle("event");
54.        setSize(350,100);
55.        setVisible(true);
56.  }
57.
58.  public static void main(String args[])
59.  {
60.        WindowEventUsingAdapter event = new
     WindowEventUsingAdapter();
61.  }
62.
63.  class ButtonListener implements ActionListener
64.  {
65.        public void actionPerformed(ActionEvent e)
66.        {
67.             Object obj = e.getSource();
68.
69.             if ( obj == reset )
70.             {
71.                  text.setText("");
72.                  text.requestFocus();
73.             }
74.
75.             else if ( obj == defaultbutton )
```

```
76.             {
77.                 text.setText("java is good");
78.                 text.selectAll();
79.                 text.requestFocus();
80.             }
81.         }
82.  }
83.
84.  class MyWindowListener extends WindowAdapter
85.  {
86.      public void windowClosing(WindowEvent e)
87.      {
88.          System.out.println("closing");
89.          dispose();
90.      }
91.  }
92. }
```

ثالثاً: شرح الكود:

- لاحظ أن الاختلاف الوحيد في هذا المثال يتمثل في الفصيلة MyWindowListener (سطر 84) ، فبدلاً من تنفيذ WindowListener interface فإنك ترث من الفصيلة WindowAdapter ، وفي هذه الحالة لا نحتاج لإعادة كتابة تعريف الدوال Methods السبعة بل نكتب فقط تعريف الدالة Method التي نريدها وهي هنا windowClosing( ).
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-3) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

معالجة الحدث Event Handling عن طريق استخدام نفس الفصيلة class:

- في جميع الأمثلة السابقة قمنا بإنشاء فصيلة class جديدة لمعالجة الحدث Event Handling ، والحقيقة أن هذا ليس ضرورياً ، فمن الممكن استخدام نفس الفصيلة class لمعالجة الحدث Event Handling.

- ويتم ذلك عن طريق تعديل الفصيلة class بحيث تنفذ الـ interface المطلوب ، وفي هذه الحالة يتم تعريف جميع الدوال Methods التي نحتاجها داخل الفصيلة class مباشرة.
- المثال التالي يوضح هذه النقطة.

مثال (10): معالجة الحدث Event Handling عن طريق استخدام نفس الفصيلة class:

أولاً: هدف المثال:

نريد تعديل المثال السابق بحيث تتم معالجة الحدث Event Handling عن طريق استخدام نفس الفصيلة class ودون الحاجة إلي إنشاء فصيلة class جديدة لمعالجة الحدث Event Handling.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class EventHandlerUsingTheSameClass extends
   JFrame
6. implements WindowListener,ActionListener
7. {
8.   JButton reset,defaultbutton;
9.   JTextField text;
10. JLabel name;
11. GridBagLayout grid;
12. GridBagConstraints gbc;
13.
14. EventHandlerUsingTheSameClass()
15. {
16.       addWindowListener(this);
17.
```

```
18.     Container c = getContentPane();
19.
20.     reset = new JButton("reset");
21.     reset.addActionListener(this);
22.     defaultbutton = new JButton("default");
23.     defaultbutton.addActionListener(this);
24.
25.     text = new JTextField(20);
26.     name = new JLabel("name");
27.
28.     grid = new GridBagLayout();
29.     gbc = new GridBagConstraints();
30.     c.setLayout(grid);
31.
32.     gbc.gridx = 1;
33.     gbc.gridy = 1;
34.     grid.setConstraints(name,gbc);
35.     c.add(name);
36.
37.     gbc.gridx = 3;
38.     gbc.gridy = 1;
39.     grid.setConstraints(text,gbc);
40.     c.add(text);
41.
42.     gbc.gridx = 3;
43.     gbc.gridy = 3;
44.     grid.setConstraints(reset,gbc);
45.     c.add(reset);
46.
47.     gbc.gridx = 4;
48.     gbc.gridy = 3;
49.     grid.setConstraints(defaultbutton,gbc);
50.     c.add(defaultbutton);
51.
```

```
52.       setTitle("event");
53.       setSize(350,100);
54.       setVisible(true);
55.  }
56.
57.  public static void main(String args[])
58.  {
59.       EventHandlerUsingTheSameClass event = new
     EventHandlerUsingTheSameClass();
60.  }
61.
62.  public void actionPerformed(ActionEvent e)
63.  {
64.            Object obj = e.getSource();
65.
66.            if ( obj == reset )
67.            {
68.                 text.setText("");
69.                 text.requestFocus();
70.            }
71.
72.            else if ( obj == defaultbutton )
73.            {
74.                 text.setText("Java is excellent");
75.                 text.selectAll();
76.                 text.requestFocus();
77.            }
78.  }
79.
80.  public void windowActivated(WindowEvent e)
81.  {
82.            System.out.println("activated");
83.  }
84.
```

```
85.  public void windowClosed(WindowEvent e)
86.  {
87.          System.out.println("byebye");
88.  }
89.
90.  public void windowClosing(WindowEvent e)
91.  {
92.          System.out.println("closing");
93.          dispose();
94.  }
95.
96.  public void windowDeactivated(WindowEvent e)
97.  {
98.          System.out.println("deactivated");
99.  }
100.
101.     public void windowIconified(WindowEvent e)
102.     {
103.             System.out.println("minimized");
104.     }
105.
106.     public void windowDeiconified(WindowEvent e)
107.     {
108.             System.out.println("maximized");
109.     }
110.
111.     public void windowOpened(WindowEvent e)
112.     {
113.             System.out.println("WELCOME");

114.     }
115.}
```

ثالثاً: شرح الكود:

- تم تعديل الفصيلة EventHandlerUsingTheSameClass بحيث تقوم بتنفيذ الـ interface المسمي WindowListener و ActionListener في نفس الوقت مع ملاحظة الفصل بينهما بعلامة الفصلة.
- في السطور من 62 إلي 114 تم تعريف جميع الدوال Methods التي نحتاجها لعمل البرنامج.
- تتبقي ملحوظة صغيرة حيث أنه عند استخدام الدالة addWindowListener( ) أو الدالة addActionListener( ) ، فقد قمنا باستخدام this كمعامل Parameter لكلا الدالتين Methods وذلك لأن نفس الفصيلة class هي التي تقوم بمعالجة الحدث Event Handling.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (3-17) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

أحداث الضغط علي زر من لوحة المفاتيح Key Events:

تتوافر الأحداث التالية عند التعامل مع لوحة المفاتيح Keyboard:

- حدث الضغط علي مفتاح من لوحة المفاتيح Keyboard.
- حدث إطلاق Release مفتاح من لوحة المفاتيح Keyboard.
- حدث كتابة حرف بعد إطلاق مفتاح من لوحة المفاتيح Keyboard.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع الأداة التي نريد معالجة حدث التعامل مع لوحة المفاتيح Keyboard عليها.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي KeyListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
keyPressed()	تنفذ هذه الدالة Method عند الضغط علي مفتاح من لوحة المفاتيح Keyboard دون إطلاقه Release أي أننا لا زلنا ضاغطين علي الزر
keyReleased()	تنفذ هذه الدالة عند إطلاق Release مفتاح من لوحة المفاتيح Keyboard
keyTyped()	تنفذ هذه الدالة Method عند كتابة حرف بعد إطلاق مفتاح من لوحة المفاتيح Keyboard

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة ( )addKeyListener.

ملحوظة:

يجب أن تعيد تعريف جميع هذه الدوال Methods حتى لو لم تكتب فيها شيئاً لأن كلها دوال مجردة abstract methods ولذلك إذا نسيت كتابة تعريف دالة Method أو أخطأت فى كتابة الدالة Method بطريقة صحيحة مع ملاحظة الحرف الصغير والكبير ، فإنك تحصل على رسالة خطأ عند ترجمة Compile البرنامج لأنه فى هذه الحالة تكون الفصيلة class التى أنشأتها - باسم MyWindowListener مثلاً - فصيلة مجردة abstract class ولا يمكن أن تنشئ منها هدفاً object.

في الدوال Methods الثلاثة الموجودين في الجدول السابق ، فإن حرف الـ k حرف صغير small.

المثال التالي يوضح النقاط السابقة.

مثال (11): معالجة حدث الضغط علي زر من لوحة المفاتيح Handling Key Event:

أولاً: هدف المثال:

نريد تعديل البرنامج السابق بحيث أن الضغط على زر الإدخال Enter يجعل النص فى أداة النص JTextField هو "You pressed enter".

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class KeyEventHandler extends JFrame
6. implements WindowListener,ActionListener
7. {
8.  JButton reset,defaultbutton;
9.  JTextField text;
10. JLabel name;
11. GridBagLayout grid;
12. GridBagConstraints gbc;
13.
14. KeyEventHandler()
15. {
16.     addWindowListener(this);
17.     MyKeyListener k = new MyKeyListener();
18.
19.     Container c = getContentPane();
20.
21.     reset = new JButton("reset");
22.     reset.addActionListener(this);
23.
24.     defaultbutton = new JButton("default");
25.     defaultbutton.addActionListener(this);
26.
27.     text = new JTextField(20);
```

```
28.         text.addKeyListener(k);
29.
30.         name = new JLabel("name");
31.
32.         grid = new GridBagLayout();
33.         gbc = new GridBagConstraints();
34.         c.setLayout(grid);
35.
36.         gbc.gridx = 1;
37.         gbc.gridy = 1;
38.         grid.setConstraints(name,gbc);
39.         c.add(name);
40.
41.         gbc.gridx = 3;
42.         gbc.gridy = 1;
43.         grid.setConstraints(text,gbc);
44.         c.add(text);
45.
46.         gbc.gridx = 3;
47.         gbc.gridy = 3;
48.         grid.setConstraints(reset,gbc);
49.         c.add(reset);
50.
51.         gbc.gridx = 4;
52.         gbc.gridy = 3;
53.         grid.setConstraints(defaultbutton,gbc);
54.         c.add(defaultbutton);
55.
56.         setTitle("event");
57.         setSize(350,100);
58.         setVisible(true);
59.   }
60.
61.  public static void main(String args[])
```

```
62. {
63.     KeyEventHandler event = new KeyEventHandler();
64. }
65.
66. public void actionPerformed(ActionEvent e)
67. {
68.         Object obj = e.getSource();
69.
70.         if ( obj == reset )
71.         {
72.             text.setText("");
73.             text.requestFocus();
74.         }
75.
76.         else if ( obj == defaultbutton )
77.         {
78.             text.setText("Java is excellent");
79.             text.selectAll();
80.             text.requestFocus();
81.         }
82. }
83.
84. public void windowActivated(WindowEvent e)
85. {
86.         System.out.println("activated");
87. }
88.
89. public void windowClosed(WindowEvent e)
90. {
91.         System.out.println("byebye");
92. }
93.
94. public void windowClosing(WindowEvent e)
95. {
```

```
96.             System.out.println("closing");
97.             dispose();
98.  }
99.
100.    public void windowDeactivated(WindowEvent e)
101.    {
102.            System.out.println("deactivated");
103.    }
104.
105.    public void windowIconified(WindowEvent e)
106.    {
107.            System.out.println("minimized");
108.    }
109.
110.    public void windowDeiconified(WindowEvent e)
111.    {
112.            System.out.println("maximized");
113.    }
114.
115.    public void windowOpened(WindowEvent e)
116.    {
117.            System.out.println("WELCOME");

118.    }
119.
120.    class MyKeyListener implements KeyListener
121.    {
122.            public void keyPressed(KeyEvent e)
123.            {
124.                System.out.println("pressed");
125.            }
126.
127.            public void keyTyped(KeyEvent e)
128.            {
```

```
129.                System.out.println("typed");
130.            }
131.
132.            public void keyReleased(KeyEvent e)
133.            {
134.                System.out.println("released");
135.                char c = e.getKeyChar();
136.
137.                if ( c == '\n' )
138.                {
139.                    text.setText("You pressed enter");
140.                }
141.            }
142.        }
143.}
```

ثالثاً: شرح الكود:

- في السطر رقم 17 يتم إنشاء هدف object من نوع الفصيلة MyKeyListener حيث نستخدم هذا الهدف object كمستمع لحدث التعامل مع لوحة المفاتيح Keyboard.
- في السطر رقم 28 يتم إضافة مستمع الحدث Event Listener لأداة النص JTextField عن طريق الدالة ( )addKeyListener.
- فى بناء الفصيلة MyWindowListener كما في السطور من 120 إلى 142 نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع لوحة المفاتيح Keyboard بحيث أن جميع الدوال Methods تطبع على شاشة الدوس Dos كلمة تفيد الحدث الذى أطلقته لوحة المفاتيح Keyboard وقمنا في الدالة ( )keyReleased بمعرفة الحرف الذى ضغطناه عن طريق استخدام الدالة getKeyChar().
- ولاختبار ما إذا كان الحرف الذي ضغطناه هو Enter فإننا نكتب if (c= = '\n') حيث '\n' يمثل زر الإدخال Enter ، وفي هذه الحالة فإننا نضع الرسالة You pressed enter فى أداة النص JTextField.

- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (3-17) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

ملحوظات:

بالطبع لابد أن تعيد تعريف الثلاثة دوال Method فى الفصيلة MykeyListener ، وإذا لم ترد ذلك ، فقم بعمل التعديل التالى على الفصيلة MyKeyListener:

```
class MyKeyListener extends KeyAdapter
{
   public void keyReleased (keyEvent e)
      {
            System.out.println ("Released");
            char c=e.getkeyChar ();
             if (c=='\n')
            {
               textMessage.setText ("You pressed enter");
            }
      }
}
```

أحداث تغيير النص في أداة النص JTextField:

تتوافر الأحداث التالية عند تغيير النص في أداة النص JTextField:

- حدث كتابة نص في أداة النص JTextField.
- حدث مسح نص في أداة النص JTextField.
- حدث تغيير خصائص النص في أداة النص JTextField مثل تغيير شكل الخط أو لونه ... إلخ.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع الأداة التي نريد معالجة حدث التعامل مع تغيير النص فيها.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي DocumentListener.

- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
insertUpdate()	تنفذ هذه الدالة Method عند كتابة نص في أداة النص JTextField
removeUpdate()	تنفذ هذه الدالة عند مسح نص في أداة النص JTextField
changedUpdate()	تنفذ هذه الدالة Method عند تغيير خصائص النص في أداة النص JTextField مثل تغيير شكل الخط أو لونه ... إلخ

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلى الأداة المطلوب معالجة أحداثها عن طريق الدالة getDocument().addDocumentListener( ).

ملحوظة:

يجب أن تعيد تعريف جميع هذه الدوال Methods حتى لو لم تكتب فيها شيئاً لأن كلها دوال مجردة abstract methods ولذلك إذا نسيت كتابة تعريف دالة Method أو أخطأت فى كتابة الدالة Method بطريقة صحيحة مع ملاحظة الحرف الصغير والكبير ، فإنك تحصل على رسالة خطأ عند ترجمة Compile البرنامج لأنه فى هذه الحالة تكون الفصيلة class التى أنشأتها -باسم MyTextListener مثلاً- فصيلة مجردة abstract class ولا يمكن أن تنشئ منها هدفاً object.

المثال التالي يوضح النقاط السابقة.

مثال (12): معالجة حدث تغيير النص في أداة النص JTextField:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي أداة نص JTextField وأداة عنوان JLabel بحيث أنه عند التعديل في النص الموجود في أداة النص JTextField ، فإن أداة العنوان تظهر الرسالة Your Name is: بالإضافة إلي النص الموجود فى أداة النص JTextField.

ثانياً: كود البرمجة:

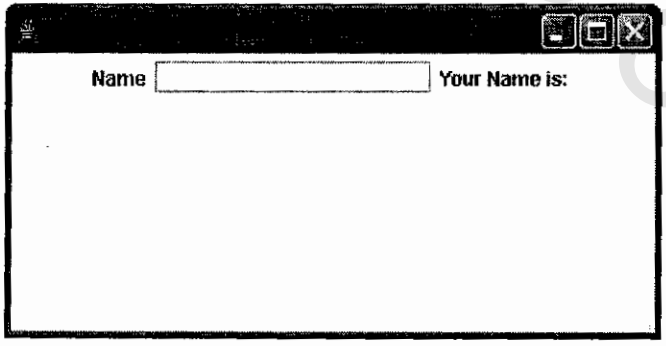
```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4. import javax.swing.event.*;
5.
6. public class TextHandler2 extends JFrame
7. {
8.   Container c;
9.   JLabel labelName,labelMessage;
10. JTextField name;
11.
12. TextHandler2()
13. {
14.       c = getContentPane();
15.       c.setLayout(new FlowLayout());
16.
17.       labelName = new JLabel("Name");
18.       name = new JTextField(15);
19.       labelMessage = new JLabel("Your Name is: ");
20.
21.       MyTextListener textListener = new
    MyTextListener();
22.
      name.getDocument().addDocumentListener(textListener);
23.
24.       c.add(labelName);
```

```
25.       c.add(name);
26.       c.add(labelMessage);
27.
28.       setSize(400,200);
29.       setVisible(true);
30.  }
31.
32.  public static void main(String[]args)
33.  {
34.       new TextHandler2();
35.  }
36.
37.  class MyTextListener implements DocumentListener
38.  {
39.       public void changedUpdate(DocumentEvent e)
40.       {
41.            System.out.println("change");
42.            updateText();
43.       }
44.
45.       public void insertUpdate(DocumentEvent e)
46.       {
47.            System.out.println("insert");
48.            updateText();
49.       }
50.
51.       public void removeUpdate(DocumentEvent e)
52.       {
53.            System.out.println("remove");
54.            updateText();
55.       }
56.  }
57.
58.  public void updateText()
59.  {
60.       String yourName = name.getText();
```

```
61.       labelMessage.setText("Your Name is: " +
    yourName);
62.  }
63.}
```

ثالثاً: شرح الكود:

- في السطر رقم 21 يتم إنشاء هدف object من نوع الفصيلة MyTextListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير النص في أداة النص JTextField.
- في السطر رقم 28 يتم إضافة مستمع الحدث Event Listener لأداة النص JTextField عن طريق الدالة getDocument().addDocumentListener().
- فى بناء الفصيلة MyTextListener كما في السطور من 37 إلى 56 نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع تغيير النص في أداة النص JTextField بحيث أن جميع الدوال Methods تطبع على شاشة الدوس Dos كلمة تفيد الحدث الذى أطلقه تغيير النص في أداة النص JTextField وقمنا في جميع الدوال Methods باستدعاء الدالة updateText( ) المعرفة في السطر رقم 58 وفيها يتم معرفة النص المكتوب في أداة النص JTextField ثم يتم تغيير النص في أداة العنوان JLabel إلي الرسالة المطلوبة.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-4) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-4) حدث تغيير النص في أداة النص JTextField

أحداث تغيير موضع نقطة الإدخال في أداة النص JTextField:

تتوافر الأحداث التالية عند تغيير موضع نقطة الإدخال في أداة النص JTextField:

- حدث تغيير موضع نقطة الإدخال في أداة النص JTextField.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع الأداة التي نريد معالجة حدث التعامل مع تغيير تغيير موضع نقطة إدخال النص فيها.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي CaretListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
caretUpdate()	تنفذ هذه الدالة Method عند تغيير موضع نقطة الإدخال في أداة النص JTextField

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addCaretListener( ).

المثال التالي يوضح النقاط السابقة.

مثال (13): معالجة حدث تغيير موضع نقطة الإدخال في أداة النص JTextField:

أولاً: هدف المثال:

نريد تعديل المثال السابق بحيث يقوم بنفس الوظيفة ولكن يتم التنفيذ عند تغيير موضع نقطة الإدخال.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.  import javax.swing.event.*;
5.
6.  public class TextHandler extends JFrame
7.  {
8.    Container c;
9.    JLabel labelName,labelMessage;
10.  JTextField name;
11.
12.  TextHandler()
13.  {
14.       c = getContentPane();
15.       c.setLayout(new FlowLayout());
16.
17.       labelName = new JLabel("Name");
18.       name = new JTextField(15);
19.       labelMessage = new JLabel("Your Name is: ");
20.
21.       MyTextListener textListener = new
     MyTextListener();
22.       name.addCaretListener(textListener);
23.
24.       c.add(labelName);
25.       c.add(name);
26.       c.add(labelMessage);
27.
28.       setSize(400,200);
29.       setVisible(true);
30.  }
31.
32.  public static void main(String[]args)
33.  {
```

```
34.       new TextHandler();
35.  }
36.
37.  class MyTextListener implements CaretListener
38.  {
39.       public void caretUpdate(CaretEvent e)
40.       {
41.            Object source = e.getSource();
42.
43.            if ( source == name )
44.            {
45.                 String yourName = name.getText();
46.                 labelMessage.setText("Your Name is: " +
     yourName);
47.            }
48.       }
49.  }
50.}
```

ثالثاً: شرح الكود:

- في السطر رقم 21 يتم إنشاء هدف object من نوع الفصيلة MyTextListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير موضع نقطة الإدخال في أداة النص JTextField.
- في السطر رقم 22 يتم إضافة مستمع الحدث Event Listener لحدث تغيير موضع نقطة الإدخال في أداة النص JTextField عن طريق الدالة addCaretListener( ).
- فى بناء الفصيلة MyTextListener كما في السطور من 37 إلى 49 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع تغيير موضع نقطة الإدخال وقمنا بتنفيذ المطلوب كما سبق لنا شرحه في المثال السابق.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (4-17) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

أحداث أداة شريط التمرير JScrollBar:

تتوافر الأحداث التالية لأداة شريط التمرير JScrollBar:

- حدث تغيير قيمة أداة شريط التمرير JScrollBar.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JScrollBar.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي AdjustmentListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
adjustmentValueChanged()	تنفذ هذه الدالة Method عند تغيير قيمة أداة شريط التمرير JScrollBar

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addAdjustmentListener().

المثال التالي يوضح النقاط السابقة.

مثال (14): معالجة حدث تغيير قيمة أداة شريط التمرير JScrollBar:

أولاً: هدف المثال:

نريد تعديل المثال رقم 11 السابق بحيث يتم إضافة شريط تمرير JScrollBar إلى التطبيق ، وعند تغيير قيمة شريط الزالقة JScrollBar تتغير الرسالة فى أداة النص JTextField.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class ScrollEventHandler extends JFrame
6.  implements WindowListener,ActionListener
7.  {
8.    JButton reset,defaultbutton;
9.    JTextField text;
10.  JLabel name;
11.  GridBagLayout grid;
12.  GridBagConstraints gbc;
13.  JScrollBar scroll;
14.
15.  ScrollEventHandler()
16.  {
17.       addWindowListener(this);
18.       MyKeyListener k = new MyKeyListener();
19.
20.       Container c = getContentPane();
21.
22.       scroll = new
     JScrollBar(JScrollBar.HORIZONTAL,1,10,1,15);
23.       MyScrollListener slisten = new MyScrollListener();
24.       scroll.addAdjustmentListener(slisten);
25.
26.       reset = new JButton("reset");
27.       reset.addActionListener(this);
28.
29.       defaultbutton = new JButton("default");
30.       defaultbutton.addActionListener(this);
31.
32.       text = new JTextField(20);
33.       text.addKeyListener(k);
```

```
34.
35.         name = new JLabel("name");
36.
37.         grid = new GridBagLayout();
38.         gbc = new GridBagConstraints();
39.         c.setLayout(grid);
40.
41.         gbc.gridx = 1;
42.         gbc.gridy = 1;
43.         grid.setConstraints(name,gbc);
44.         c.add(name);
45.
46.         gbc.gridx = 3;
47.         gbc.gridy = 1;
48.         grid.setConstraints(text,gbc);
49.         c.add(text);
50.
51.         gbc.gridx = 3;
52.         gbc.gridy = 3;
53.         grid.setConstraints(reset,gbc);
54.         c.add(reset);
55.
56.         gbc.gridx = 4;
57.         gbc.gridy = 3;
58.         grid.setConstraints(defaultbutton,gbc);
59.         c.add(defaultbutton);
60.
61.         gbc.gridx = 1;
62.         gbc.gridy = 4;
63.         grid.setConstraints(scroll,gbc);
64.         c.add(scroll);
65.
66.         setTitle("event");
67.         setSize(500,300);
68.         setVisible(true);
```

```
69.  }
70.
71.  public static void main(String args[])
72.  {
73.       ScrollEventHandler event = new
     ScrollEventHandler();
74.  }
75.
76.  public void actionPerformed(ActionEvent e)
77.  {
78.          Object obj = e.getSource();
79.
80.          if ( obj == reset )
81.          {
82.               text.setText("");
83.               text.requestFocus();
84.          }
85.
86.          else if ( obj == defaultbutton )
87.          {
88.               text.setText("Java is excellent");
89.               text.selectAll();
90.               text.requestFocus();
91.          }
92.  }
93.
94.  public void windowActivated(WindowEvent e)
95.  {
96.          System.out.println("activated");
97.  }
98.
99.  public void windowClosed(WindowEvent e)
100.     {
101.             System.out.println("byebye");
102.     }
```

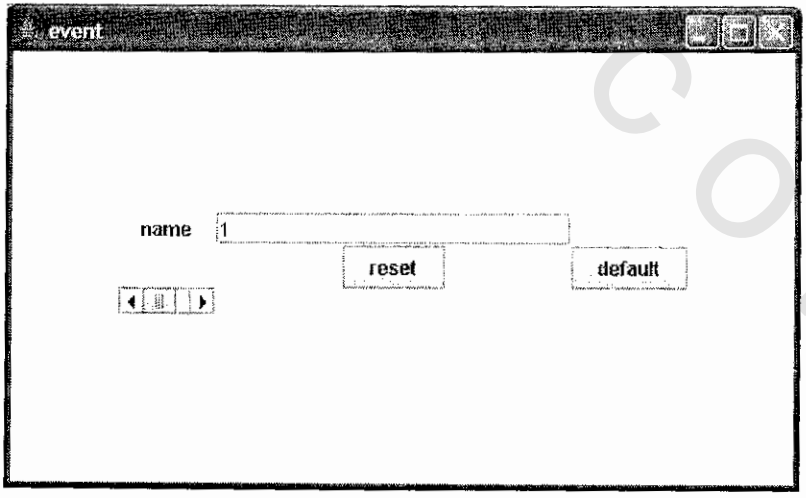
```
103.
104.    public void windowClosing(WindowEvent e)
105.    {
106.            System.out.println("closing");
107.            dispose();
108.    }
109.
110.    public void windowDeactivated(WindowEvent e)
111.    {
112.            System.out.println("deactivated");
113.    }
114.
115.    public void windowIconified(WindowEvent e)
116.    {
117.            System.out.println("minimized");
118.    }
119.
120.    public void windowDeiconified(WindowEvent e)
121.    {
122.            System.out.println("maximized");
123.    }
124.
125.    public void windowOpened(WindowEvent e)
126.    {
127.            System.out.println("WELCOME");
128.
129.            int i = scroll.getValue();
130.            String s = "" + i;
131.            text.setText(s);
132.    }
133.
134.    class MyKeyListener implements KeyListener
135.    {
136.            public void keyPressed(KeyEvent e)
137.            {
```

```
138.                System.out.println("pressed");
139.            }
140.
141.            public void keyTyped(KeyEvent e)
142.            {
143.                System.out.println("typed");
144.            }
145.
146.            public void keyReleased(KeyEvent e)
147.            {
148.                System.out.println("released");
149.                char c = e.getKeyChar();
150.
151.                if(c == '\n')
152.                {
153.                    text.setText("you pressed enter");
154.                }
155.            }
156.        }
157.
158.        class MyScrollListener implements
    AdjustmentListener
159.        {
160.            public void
    adjustmentValueChanged(AdjustmentEvent e)
161.            {
162.                Object obj = e.getSource();
163.
164.                if ( obj == scroll )
165.                {
166.                    int i = scroll.getValue();
167.                    String s = "" + i;
168.                    text.setText(s);
169.                }
170.            }
```

```
171.        }
172.}
```

ثالثاً: شرح الكود:

- في السطر رقم 23 يتم إنشاء هدف object من نوع الفصيلة MyScrollListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير قيمة أداة شريط التمرير JScrollBar.
- في السطر رقم 24 يتم إضافة مستمع الحدث Event Listener لحدث تغيير قيمة أداة شريط التمرير JScrollBar عن طريق الدالة ()addAdjustmentListener.
- فى بناء الفصيلة MyScrollListener كما في السطور من 158 إلى 171 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع تغيير قيمة أداة شريط التمرير JScrollBar وقمنا باستدعاء الدالة ( )getValue لمعرفة قيمة أداة شريط التمرير JScrollBar ثم قمنا بالتعديل في النص الموجود في أداة النص JTextField ليعكس قيمة أداة شريط التمرير JScrollBar.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-5) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-5) حدث تغيير قيمة أداة شريط التمرير JScrollBar

مثال (15): معالجة حدث تغيير قيمة أداة شريط التمرير JScrollBar:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي أداة عنوان JLabel وثلاثة أشرطة تمرير JScrollBar وزر JButton بحيث أنه عند تغيير قيمة أي من أشرطة التمرير JScrollBar ، فإن لون النص في أداة العنوان JLabel يتغير ، بالإضافة إلي ظهور القيمة الحالية لأشرطة التمرير JScrollBar ، وعند الضغط علي الزر JButton فإن لون النص يتحول إلي الرمادي وتتغير قيم أشرطة التمرير JScrollBar لتعكس القيم الجديدة التي ستصبح 128 لأشرطة التمرير JScrollBar.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class ColorScroll extends JFrame
6.  {
7.   JLabel
    text,redlabel,greenlabel,bluelabel,redvalue,greenvalue,blue
    value;
8.   JScrollBar redscroll,bluescroll,greenscroll;
9.   JPanel redpanel,bluepanel,greenpanel;
10. int r = 128,g = 128,b = 128;
11. JButton reset;
12.
13. ColorScroll()
14. {
15.       Container c = getContentPane();
16.       c.setLayout(new GridLayout(5,1));
17.
18.       redlabel = new JLabel("red");
19.       greenlabel = new JLabel("green");
```

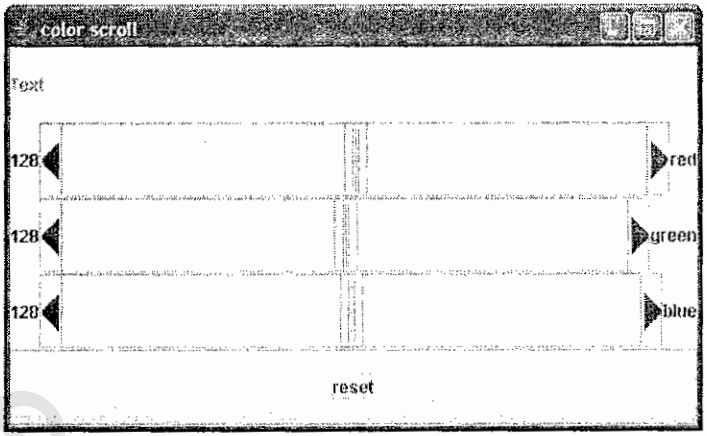
```
20.       bluelabel = new JLabel("blue");
21.       text = new JLabel("Text");
22.       redvalue = new JLabel("128");
23.       greenvalue = new JLabel("128");
24.       bluevalue = new JLabel("128");
25.       reset = new JButton("reset");
26.
27.       MyButtonListener b = new MyButtonListener();
28.       reset.addActionListener(b);
29.
30.       redscroll = new
    JScrollBar(JScrollBar.HORIZONTAL,128,10,0,265);
31.       greenscroll = new
    JScrollBar(JScrollBar.HORIZONTAL,128,10,0,265);
32.       bluescroll = new
    JScrollBar(JScrollBar.HORIZONTAL,128,10,0,265);
33.
34.       MyScrolllListener l = new MyScrolllListener();
35.       redscroll.addAdjustmentListener(l);
36.       bluescroll.addAdjustmentListener(l);
37.       greenscroll.addAdjustmentListener(l);
38.
39.       redpanel = new JPanel();
40.       bluepanel = new JPanel();
41.       greenpanel = new JPanel();
42.
43.       c.add(text);
44.       c.add(redpanel);
45.       c.add(greenpanel);
46.       c.add(bluepanel);
47.       c.add(reset);
48.
49.       redpanel.setLayout(new BorderLayout());
50.       redpanel.add("East",redlabel);
```

```
51.       redpanel.add("Center",redscroll);
52.       redpanel.add("West",redvalue);
53.
54.       greenpanel.setLayout(new BorderLayout());
55.       greenpanel.add("East",greenlabel);
56.       greenpanel.add("Center",greenscroll);
57.       greenpanel.add("West",greenvalue);
58.
59.       bluepanel.setLayout(new BorderLayout());
60.       bluepanel.add("East",bluelabel);
61.       bluepanel.add("Center",bluescroll);
62.       bluepanel.add("West",bluevalue);
63.
64.       setTitle("color scroll");
65.       setSize(500,300);
66.       setVisible(true);
67.  }
68.
69.  public static void main(String args[])
70.  {
71.       ColorScroll color = new ColorScroll();
72.  }
73.
74.  class MyScrolllListener implements AdjustmentListener
75.  {
76.       public void
     adjustmentValueChanged(AdjustmentEvent e)
77.       {
78.            Object obj = e.getSource();
79.
80.            if ( obj == redscroll )
81.            {
82.                 r = redscroll.getValue();
83.                 redvalue.setText("" + r);
```

```
84.          }
85.
86.          else if ( obj == greenscroll )
87.          {
88.              g = greenscroll.getValue();
89.              greenvalue.setText(""+g);
90.          }
91.
92.          else if ( obj == bluescroll )
93.          {
94.              b = bluescroll.getValue();
95.              bluevalue.setText(""+b);
96.          }
97.
98.          text.setForeground(new Color(r,g,b));
99.      }
100.     }
101.
102.     class MyButtonListener implements ActionListener
103.     {
104.         public void actionPerformed(ActionEvent e)
105.         {
106.             redvalue.setText("128");
107.             greenvalue.setText("128");
108.             bluevalue.setText("128");
109.             text.setForeground (new Color(128,128,128));
110.             redscroll.setValue(128);
111.             greenscroll.setValue(128);
112.             bluescroll.setValue(128);
113.         }
114.     }
115.}
```

ثالثاً: شرح الكود:

- في السطر رقم 27 يتم إنشاء هدف object من نوع الفصيلة MyButtonListener حيث نستخدم هذا الهدف object كمستمع لحدث الضغط علي أداة الزر JButton.
- في السطر رقم 28 يتم إضافة مستمع الحدث Event Listener لحدث الضغط علي أداة الزر JButton عن طريق الدالة addActionListener().
- في السطر رقم 34 يتم إنشاء هدف object من نوع الفصيلة MyScrollListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير قيمة أداة شريط التمرير JScrollBar.
- في السطور من 35 إلي 37 يتم إضافة مستمع الحدث Event Listener لحدث تغيير قيمة أداة شريط التمرير JScrollBar عن طريق الدالة addAdjustmentListener().
- فى بناء الفصيلة MyScrollListener كما في السطور من 74 إلى 100 نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع تغيير قيمة أداة شريط التمرير JScrollBar وقمنا باستدعاء الدالة getValue( ) لمعرفة قيمة أداة شريط التمرير JScrollBar ثم قمنا بالتعديل في لون النص الموجود في أداة العنوان JLabel حسب قيمة أداة شريط التمرير JScrollBar.
- فى بناء الفصيلة MyButtonListener كما في السطور من 102 إلى 114 نجد أننا قمنا بالتعديل في قيم أشرطة التمرير JScrollBar لتصبح 128 باستخدام الدالة setValue( ) كما تم تغيير لون النص الموجود في أداة العنوان JLabel حسب قيمة أداة شريط التمرير JScrollBar وتم إظهار القيم الجديدة لأشرطة التمرير JScrollBar.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-6) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-6) حدث تغيير قيمة أداة شريط التمرير JScrollBar

أحداث أداة قائمة الاختيارات JComboBox:

تتوافر الأحداث التالية لأداة قائمة الاختيارات JComboBox:

- حدث تغيير الاختيار في أداة قائمة الاختيارات JComboBox.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JComboBox.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ItemListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
itemStateChanged()	تنفذ هذه الدالة Method عند تغيير الاختيار في أداة قائمة الاختيارات JComboBox

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addItemListener().

المثال التالي يوضح النقاط السابقة.

مثال (16): معالجة حدث تغيير الاختيار في أداة قائمة الاختيارات JComboBox:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي قائمة باختيارات الدول ، وعند تغيير الدولة تظهر رسالة للمستخدم تبين اسم الدولة التي اختارها.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class ComboBoxHandler extends JFrame
6. {
7.   JComboBox country;
8.   Container c;
9.
10. ComboBoxHandler()
11. {
12.       c = getContentPane();
13.       c.setLayout(new FlowLayout());
14.
15.
    String[]countries={"Egypt","Lebanon","Morocco","KSA"}
    ;
16.       country = new JComboBox(countries);
17.       c.add(country);
18.
19.       country.addItemListener(new ItemListener()
20.       {
21.           public void itemStateChanged(ItemEvent e)
22.           {
```

```
23.                 String message = "You are from " +
    (String)country.getSelectedItem();
24.
    JOptionPane.showMessageDialog(c,message);
25.             }
26.         });
27.
28.         setSize(200,200);
29.         setVisible(true);
30.  }
31.
32.  public static void main(String[]args)
33.  {
34.         new ComboBoxHandler();
35.  }
36.}
```

ثالثاً: شرح الكود:

- في السطر رقم 19 يتم إضافة مستمع الحدث Event Listener لحدث تغيير الاختيار في أداة قائمة الاختيارات JComboBox عن طريق الدالة addItemListener().
- استخدمنا هنا طريقة الفصائل الداخلية اللاسمية inner anonymous classes حيث قمنا بتعريف الدالة itemStateChanged( ) وفيها تم معرفة القيمة المختارة عن طريق الدالة getSelectedItem( ) وتم إظهار رسالة للمستخدم باستخدام أداة صناديق الرسائل JOptionPane.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-7) وتأكد من عمل البرنامج بالشكل المطلوب.

الشكل (17-7) حدث تغيير الاختيار في أداة قائمة الاختيارات JComboBox

أحداث أداة السرد JList:

تتوافر الأحداث التالية لأداة السرد JList:

- حدث تغيير الاختيار في أداة السرد JList.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JList.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ListSelectionListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
valueChanged()	تنفذ هذه الدالة Method عند تغيير الاختيار في أداة السرد JList

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addListSelectionListener().

المثال التالي يوضح النقاط السابقة.

مثال (17): معالجة حدث تغيير الاختيار في أداة السرد JList:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي قائمة تحتوي علي الحالة الاجتماعية إما أعزب وإما متزوج بالإضافة إلي أداة نص JTextField لإدخال عدد الأولاد ، إذا اختار المستخدم الاختيار أعزب فتصبح أداة النص JTextField غير فعالة Disabled وإذا اختار المستخدم الاختيار متزوج فتصبح أداة النص JTextField فعالة Enabled.

ثانياً: كود البرمجة:

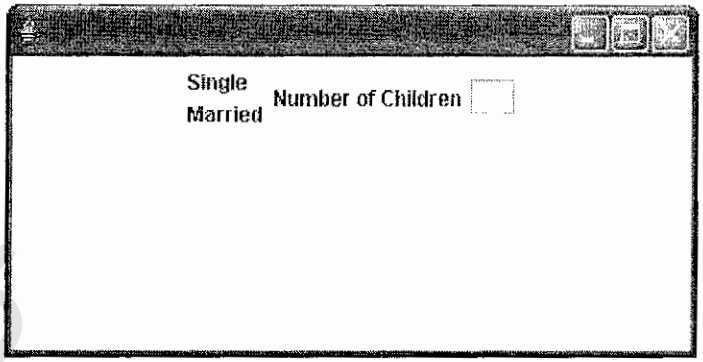
```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.  import javax.swing.event.*;
5.
6.  public class ListHandler extends JFrame
7.  {
8.    Container c;
9.    JList marital;
10.  JLabel labelChildren;
11.  JTextField numberOfChildren;
12.
13.  ListHandler()
14.  {
15.       c = getContentPane();
16.       c.setLayout(new FlowLayout());
17.
18.       String maritalOptions[] = {"Single","Married"};
19.       marital = new JList(maritalOptions);
20.
21.       ListListener listListener = new ListListener();
22.       marital.addListSelectionListener(listListener);
23.
```

```
24.       c.add(marital);
25.
26.       labelChildren = new JLabel("Number of Children");
27.       numberOfChildren = new JTextField(2);
28.       c.add(labelChildren);
29.       c.add(numberOfChildren);
30.
31.       setSize(400,200);
32.       setVisible(true);
33.  }
34.
35.  public static void main(String[]args)
36.  {
37.       new ListHandler();
38.  }
39.
40.  class ListListener implements ListSelectionListener
41.  {
42.       public void valueChanged(ListSelectionEvent e)
43.       {
44.            Object source = e.getSource();
45.
46.            if ( source == marital )
47.            {
48.                 String selectedValue =
     (String)marital.getSelectedValue();
49.
50.                 if ( selectedValue != null &&
     selectedValue.equalsIgnoreCase("single") )
51.                 {
52.                      numberOfChildren.setText("0");
53.                      numberOfChildren.setEnabled(false);
54.                 }
```

```
55.
56.                 else if ( selectedValue != null &&
    selectedValue.equalsIgnoreCase("married") )
57.                 {
58.                     numberOfChildren.setText("");
59.                     numberOfChildren.setEnabled(true);
60.                     numberOfChildren.requestFocus();
61.                 }
62.             }
63.         }
64.     }
65. }
```

ثالثاً: شرح الكود:

- في السطر رقم 21 يتم إنشاء هدف object من نوع الفصيلة ListListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير الاختيار في أداة السرد JList.
- في السطر رقم 22 يتم إضافة مستمع الحدث Event Listener لحدث تغيير الاختيار في أداة السرد JList عن طريق الدالة addListSelectionListener().
- فى بناء الفصيلة ListListener كما في السطور من 40 إلى 64 نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث تغيير الاختيار في أداة السرد JList وقمنا باستدعاء الدالة getSelectedValue( ) لمعرفة القيمة المختارة من أداة السرد JList ثم قمنا بالتحقق من القيمة المختارة ، فإذا اختار المستخدم الاختيار أعزب فإن البرنامج يقوم بتغيير عدد الأولاد إلي صفر (بالطبع لا بد أن تكون متزوجاً حتي يكون لك أولاد) ثم يقوم بجعل أداة النص JTextField غير فعالة Disabled ، أما إذا اختار المستخدم الاختيار متزوج فإن البرنامج يقوم بمسح عدد الأولاد ثم يقوم بجعل أداة النص JTextField فعالة Enabled ثم يتم وضع مؤشر الفارة Mouse في أداة النص JTextField.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-8) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-8) حدث تغيير الاختيار في أداة السرد JList

أحداث أداة زر الراديو JRadioButton:

تتوافر الأحداث التالية لأداة زر الراديو JRadioButton:

- حدث تغيير الاختيار في أداة زر الراديو JRadioButton.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JRadioButton.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ActionListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
actionPerformed()	تنفذ هذه الدالة Method عند تغيير الاختيار في أداة زر الراديو JRadioButton

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addActionListener().

المثال التالي يوضح النقاط السابقة.

مثال (18): معالجة حدث تغيير الاختيار في أداة زر الراديو JRadioButton:

أولاً: هدف المثال:

نريد تنفيذ نفس البرنامج السابق ولكن باستخدام أداة زر الراديو JRadioButton.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class RadioButtonHandler extends JFrame
6.  {
7.    Container c;
8.    ButtonGroup marital;
9.    JRadioButton single;
10.  JRadioButton married;
11.  JLabel labelChildren;
12.  JTextField numberOfChildren;
13.
14.  RadioButtonHandler()
15.  {
16.       c = getContentPane();
17.       c.setLayout(new FlowLayout());
18.
19.       marital = new ButtonGroup();
20.       single = new JRadioButton("Single");
21.       married = new JRadioButton("Married");
22.       marital.add(single);
23.       marital.add(married);
24.
25.       RadioListener radioListener = new RadioListener();
26.       single.addActionListener(radioListener);
27.       married.addActionListener(radioListener);
28.
```

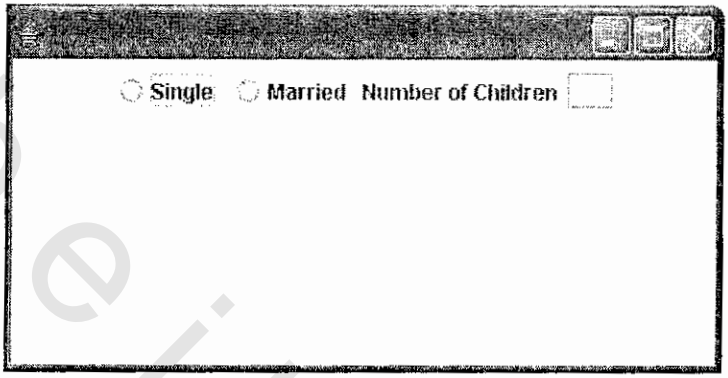
```
29.         c.add(single);
30.         c.add(married);
31.
32.         labelChildren = new JLabel("Number of Children");
33.         numberOfChildren = new JTextField(2);
34.         c.add(labelChildren);
35.         c.add(numberOfChildren);
36.
37.         setSize(400,200);
38.         setVisible(true);
39.     }
40.
41.     public static void main(String[]args)
42.     {
43.         new RadioButtonHandler();
44.     }
45.
46.     class RadioListener implements ActionListener
47.     {
48.         public void actionPerformed(ActionEvent e)
49.         {
50.             Object source = e.getSource();
51.
52.             if ( source == single )
53.             {
54.                 boolean isSingle = single.isSelected();
55.
56.                 if ( isSingle == true )
57.                 {
58.                     numberOfChildren.setText("0");
59.                     numberOfChildren.setEnabled(false);
60.                 }
61.             }
62.
```

```
63.          else if ( source == married )
64.          {
65.              boolean isMarried = married.isSelected();
66.
67.              if ( isMarried == true )
68.              {
69.                  numberOfChildren.setText("");
70.                  numberOfChildren.setEnabled(true);
71.                  numberOfChildren.requestFocus();
72.              }
73.          }
74.      }
75.  }
76.}
```

ثالثاً: شرح الكود:

- في السطر رقم 25 يتم إنشاء هدف object من نوع الفصيلة RadioListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير الاختيار في أداة زر الراديو JRadioButton.
- في السطرين 26 و 27 يتم إضافة مستمع الحدث Event Listener لحدث تغيير الاختيار في أداة زر الراديو JRadioButton عن طريق الدالة addActionListener().
- فى بناء الفصيلة RadioListener كما في السطور من 46 إلى 75 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث تغيير الاختيار في أداة زر الراديو JRadioButton وقمنا باستدعاء الدالة ( )isSelected لمعرفة ما إذا كانت القيمة المختارة من أداة زر الراديو JRadioButton هي الاختيار أعزب أم لا (راجع السطر رقم 54) ثم قمنا بتنفيذ المطلوب مثل المثال السابق ، وبالمثل للاختيار متزوج.

- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-9) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-9) حدث تغيير الاختيار في أداة زر الراديو JRadioButton

ملحوظة:

تم استخدام الفصيلة ButtonGroup لإنشاء مجموعة اختيار تحتوي علي الاختيار أعزب ومتزوج بحيث يتم اختيار قيمة واحدة فقط منهما ولا يمكن للمستخدم اختيار القيمتين معاً ، فاختيار قيمة من الاختيارات تقوم بإلغاء العلامة من باقي الاختيارات.

أحداث أداة صندوق التحقق JCheckBox:

تتوافر الأحداث التالية لأداة صندوق التحقق JCheckBox:

- حدث تغيير الاختيار في أداة صندوق التحقق JCheckBox.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JCheckBox.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ActionListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

Method الدالة	الاستخدام
actionPerformed()	تنفذ هذه الدالة Method عند تغيير الاختيار في أداة صندوق التحقق JCheckBox

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلى الأداة المطلوب معالجة أحداثها عن طريق الدالة addActionListener().

المثال التالي يوضح النقاط السابقة.

مثال (19): معالجة حدث تغيير الاختيار في أداة صندوق التحقق JCheckBox:

أولاً: هدف المثال:

نريد تنفيذ نفس البرنامج السابق ولكن باستخدام أداة صندوق التحقق JCheckBox.

ثانياً: كود البرمجة:

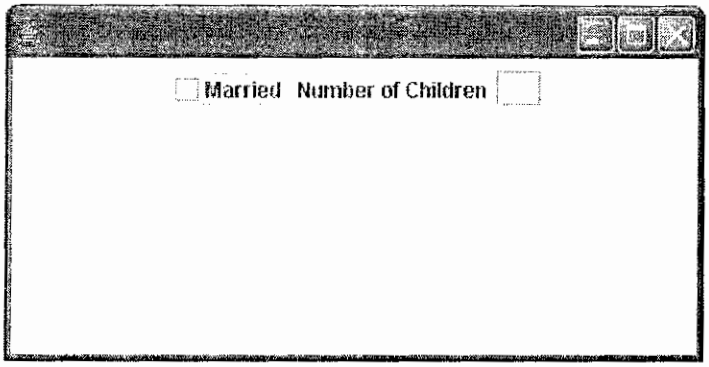
```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class CheckBoxHandler extends JFrame
6.  {
7.    Container c;
8.    JCheckBox married;
9.    JLabel labelChildren;
10.   JTextField numberOfChildren;
11.
12.   CheckBoxHandler()
13.   {
14.       c = getContentPane();
15.       c.setLayout(new FlowLayout());
```

```
16.
17.       married = new JCheckBox("Married");
18.
19.       CheckBoxListener checkBoxListener = new
    CheckBoxListener();
20.       married.addActionListener(checkBoxListener);
21.
22.       c.add(married);
23.
24.       labelChildren = new JLabel("Number of Children");
25.       numberOfChildren = new JTextField(2);
26.       c.add(labelChildren);
27.       c.add(numberOfChildren);
28.
29.       setSize(400,200);
30.       setVisible(true);
31.  }
32.
33.  public static void main(String[]args)
34.  {
35.       new CheckBoxHandler();
36.  }
37.
38.  class CheckBoxListener implements ActionListener
39.  {
40.       public void actionPerformed(ActionEvent e)
41.       {
42.            Object source = e.getSource();
43.
44.            if ( source == married )
45.            {
46.                 boolean isMarried = married.isSelected();
47.
48.                 if ( isMarried == true )
```

```
49.                 {
50.                     numberOfChildren.setText("");
51.                     numberOfChildren.setEnabled(true);
52.                     numberOfChildren.requestFocus();
53.                 }
54.
55.                 else
56.                 {
57.                     numberOfChildren.setText("0");
58.                     numberOfChildren.setEnabled(false);
59.                 }
60.             }
61.         }
62.  }
63.}
```

ثالثاً: شرح الكود:

- في السطر رقم 19 يتم إنشاء هدف object من نوع الفصيلة CheckBoxListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير الاختيار في أداة صندوق التحقق JCheckBox.
- في السطر رقم 20 يتم إضافة مستمع الحدث Event Listener لحدث تغيير الاختيار في أداة صندوق التحقق JCheckBox عن طريق الدالة addActionListener().
- فى بناء الفصيلة CheckBoxListener كما في السطور من 38 إلى 62 نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث تغيير الاختيار في أداة صندوق التحقق JCheckBox وقمنا باستدعاء الدالة ( )isSelected لمعرفة ما إذا كانت القيمة مختارة من أداة صندوق التحقق JCheckBox أم لا (راجع السطر رقم 46) ثم قمنا بتنفيذ المطلوب مثل المثال السابق.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-10) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-10) حدث تغيير الاختيار في أداة صندوق التحقق JCheckBox

أحداث أداة اللوحة المبوبة JTabbedPane:

تتوافر الأحداث التالية لأداة اللوحة المبوبة JTabbedPane:

- حدث تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JTabbedPane.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ChangeListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
stateChanged()	تنفذ هذه الدالة Method عند تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addChangeListener().

المثال التالي يوضح النقاط السابقة.

مثال (20): معالجة حدث تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane:

أولاً: هدف المثال:

نريد تغيير لون خلفية البرنامج عندما يتم تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import javax.swing.event.*;
4.
5.  public class TabHandler extends JFrame
6.  {
7.   JButton submit, cancel;
8.   JPanel p1, p2;
9.   JTabbedPane tab;
10.
11.  TabHandler()
12.  {
13.       Container c = getContentPane();
14.
15.       submit = new JButton("Submit");
16.       cancel = new JButton("Cancel");
17.
18.       p1 = new JPanel();
19.       p2 = new JPanel();
20.
21.       tab = new JTabbedPane();
22.
23.       createPanel1();
24.       createPanel2();
25.
26.       c.add(tab);
27.
```

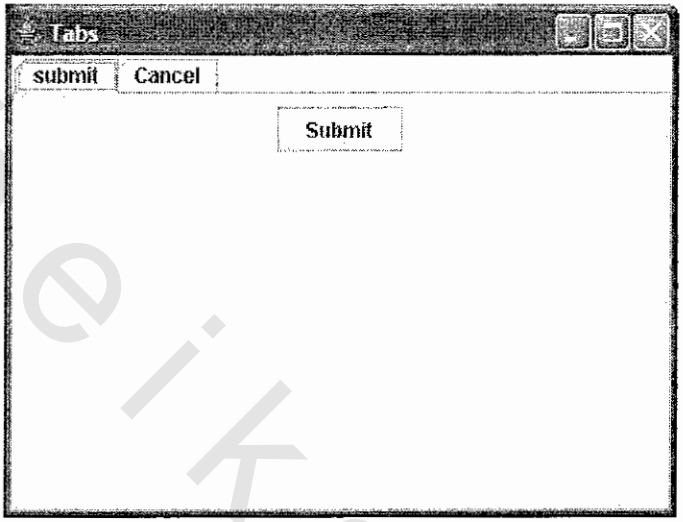
```
28.       tab.addTab("submit",null,p1,"this is panel1");
29.       tab.addTab("Cancel",null,p2,"this is panel2");
30.
31.       TabListener tabListener = new TabListener();
32.       tab.addChangeListener(tabListener);
33.
34.       setTitle("Tabs");
35.       setSize(400,300);
36.       setVisible(true);
37.  }
38.
39.  void createPanel1()
40.  {
41.       p1.add(submit);
42.  }
43.
44.  void createPanel2()
45.  {
46.       p2.add(cancel);
47.  }
48.
49.  public static void main(String args[])
50.  {
51.       TabHandler tab=new TabHandler();
52.  }
53.
54.  class TabListener implements ChangeListener
55.  {
56.       public void stateChanged(ChangeEvent e)
57.       {
58.            Object source = e.getSource();
59.
60.            if ( source == tab )
61.            {
```

```
62.                 JPanel selectedPanel =
    (JPanel)tab.getSelectedComponent();
63.
64.                 if ( selectedPanel == p1 )
65.                 {
66.                     p1.setBackground(Color.yellow);
67.                 }
68.
69.                 else if ( selectedPanel == p2 )
70.                 {
71.                     p2.setBackground(Color.blue);
72.                 }
73.             }
74.         }
75.  }
76.}
```

ثالثاً: شرح الكود:

- في السطر رقم 31 يتم إنشاء هدف object من نوع الفصيلة TabListener حيث نستخدم هذا الهدف object كمستمع لحدث تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane.
- في السطر رقم 32 يتم إضافة مستمع الحدث Event Listener لحدث تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane عن طريق الدالة addChangeListener().
- في بناء الفصيلة TabListener كما في السطور من 54 إلى 75 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane وقمنا باستدعاء الدالة getSelectedComponent( ) لمعرفة التبويب Tab المختار من أداة اللوحة المبوبة JTabbedPane (راجع السطر رقم 62) ثم قمنا بالتحقق من التبويب Tab المختار ، فإذا كان اللوحة p1 فإننا نقوم بتغيير لون الخلفية وبالمثل بالنسبة للوحة p2.

- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (17-11) وتأكد من عمل البرنامج بالشكل المطلوب.



الشكل (17-11) حدث تغيير التبويب Tab في أداة اللوحة المبوبة JTabbedPane

أحداث أداة القوائم JMenu:

تتوافر الأحداث التالية لأداة القوائم JMenu:

- حدث الضغط علي أداة القوائم JMenu.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع JMenuItem. تذكر أن الفصيلة JMenu تستخدم لإنشاء القوائم الرئيسية أما الفصيلة JMenuItem فتستخدم لإنشاء القوائم الفرعية.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي ActionListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
actionPerformed( )	تنفذ هذه الدالة Method عند الضغط علي أداة القوائم JMenu

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addActionListener().

المثال التالي يوضح النقاط السابقة.

مثال (21): معالجة حدث الضغط علي أداة القوائم JMenu:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي قائمتين رئيسيتين هما File و Edit بالإضافة إلي أداة نص JTextField.

تحتوي القائمة File علي قائمتين فرعيتين هما New و Close بحيث أن الضغط علي القائمة New يقوم بمسح النص الموجود في أداة النص JTextField والضغط علي القائمة Close يقوم بإغلاق البرنامج.

تحتوي القائمة Edit علي قائمتين فرعيتين هما Copy و Paste بحيث أن الضغط علي القائمة Copy يقوم بنسخ النص الموجود في أداة النص JTextField إلي الحافظة Clip Board والضغط علي القائمة Paste يقوم بلصق النص الموجود في الحافظة Clip Board في أداة النص JTextField.

ثانياً: كود البرمجة:

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

```
5.  public class MenuHandler extends JFrame
6.  {
7.    JMenuBar bar;
8.    JMenu file, edit;
9.    JMenuItem mnew, close, copy, paste;
10.  JTextField text;
11.
12.  MenuHandler()
13.  {
14.       Container c = getContentPane();
15.       c.setLayout(new BorderLayout());
16.
17.       bar = new JMenuBar();
18.       setJMenuBar(bar);
19.
20.       file = new JMenu("File");
21.       edit = new JMenu("Edit");
22.
23.       bar.add(file);
24.       bar.add(edit);
25.
26.       MyMenuListener myMenuListener = new
    MyMenuListener();
27.
28.       mnew = new JMenuItem("New");
29.       mnew.addActionListener(myMenuListener);
30.       close = new JMenuItem("Close");
31.       close.addActionListener(myMenuListener);
32.       copy = new JMenuItem("Copy");
33.       copy.addActionListener(myMenuListener);
34.       paste = new JMenuItem("Paste");
35.       paste.addActionListener(myMenuListener);
36.
```

```
37.         file.add(mnew);
38.         file.addSeparator();
39.         file.add(close);
40.         edit.add(copy);
41.         edit.addSeparator();
42.         edit.add(paste);
43.
44.         text = new JTextField();
45.         c.add("Center",text);
46.
47.         setTitle("Menus");
48.         setSize(400,300);
49.         setVisible(true);
50.   }
51.
52.   public static void main(String args[])
53.   {
54.         MenuHandler menus = new MenuHandler();
55.   }
56.
57.   class MyMenuListener implements ActionListener
58.   {
59.         public void actionPerformed(ActionEvent e)
60.         {
61.               Object source = e.getSource();
62.
63.               if ( source == mnew )
64.               {
65.                     text.setText("");
66.                     text.requestFocus();
67.               }
68.
69.               else if ( source == close )
```

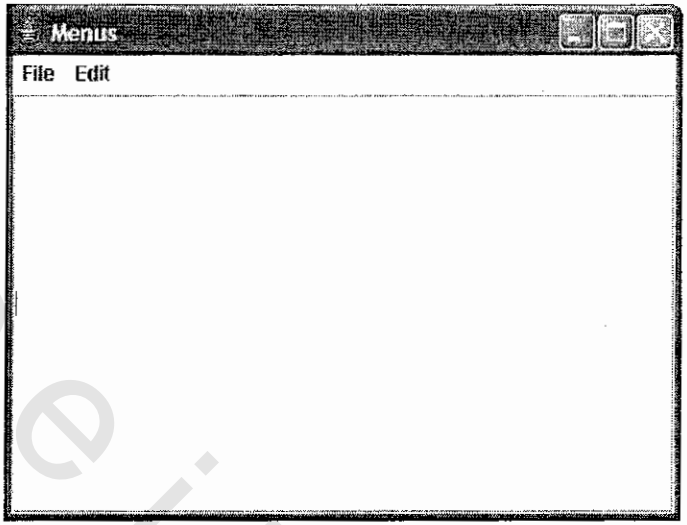
```
70.             {
71.                 dispose();
72.                 System.exit(0);
73.             }
74.
75.             else if ( source == copy )
76.             {
77.                 text.copy();
78.             }
79.
80.             else if ( source == paste )
81.             {
82.                 text.paste();
83.             }
84.         }
85.  }
86.}
```

ثالثاً: شرح الكود:

- في السطر رقم 26 يتم إنشاء هدف object من نوع الفصيلة MyMenuListener حيث نستخدم هذا الهدف object كمستمع لحدث الضغط علي أداة القوائم JMenu.
- في السطر رقم 29 و 31 و 33 و 35 يتم إضافة مستمع الحدث Event Listener لحدث الضغط علي أداة القوائم الفرعية JMenuItem عن طريق الدالة addActionListener().
- فى بناء الفصيلة MyMenuListener كما في السطور من 57 إلى 85 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث الضغط علي أداة القوائم الفرعية JMenuItem وقمنا بتنفيذ المطلوب لكل قائمة مع ملاحظة استخدام الدالة copy( ) كما هو واضح في السطر رقم 77 ، حيث تقوم هذه

الدالة Method بنسخ النص الموجود في أداة النص JTextField إلي الحافظة Clip Board ، وبالمثل تم استخدام الدالة paste( ) للصق النص الموجود في الحافظة Clip Board في أداة النص JTextField.

- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (12-17) وتأكد من عمل البرنامج بالشكل المطلوب.
- يمكنك تجربة الآتي للتأكد من عمل البرنامج بشكل صحيح.
- أولاً قم بكتابة أي نص في أداة النص JTextField ثم قم بفتح القائمة File ثم اضغط New وتأكد من مسح النص الموجود في أداة النص JTextField.
- ثانياً قم بكتابة أي نص في أداة النص JTextField ثم قم بالتظليل Select علي النص الذي تريد نسخه ثم قم بفتح القائمة Edit ثم اضغط Copy. يمكنك فتح أي برنامج أخر وليكن برنامج المفكرة Notepad الموجود في نظام النوافذ Windows وتجربة لصق Paste النص للتأكد من أن البرنامج قام فعلاً بنسخ النص إلي الحافظة Clip Board وبالتالي أصبح متاحاً لأي برنامج أخر.
- ثالثاً قم بالضغط في أداة النص JTextField ثم قم بوضع نقطة إدخال النص في الموضع الذي تريد لصق Paste النص فيه ثم قم بفتح القائمة Edit ثم اضغط Paste وتأكد من كتابة النص السابق نسخه إلي الحافظة Clip Board. ليس هذا فحسب ، فيمكنك فتح أي برنامج أخر وليكن برنامج المفكرة Notepad الموجود في نظام النوافذ Windows وتجربة كتابة أي نص ثم نسخه وبعد ذلك قم بتجربة لصق Paste النص في أداة النص JTextField الموجودة في برنامجنا للتأكد من أن البرنامج قام فعلاً بلصق Paste النص من الحافظة Clip Board وبالتالي يمكننا اللصق Paste من أي برنامج أخر.



الشكل (17-12) حدث الضغط علي أداة القوائم الفرعية JMenuItem

أحداث الفارة Mouse Events:

تتوافر الأحداث التالية عند التعامل مع الفارة Mouse:

- حدث الضغط علي زر الفارة Mouse.
- حدث إطلاق Release زر الفارة Mouse.
- حدث إكمال الضغط علي زر الفارة Mouse بعد الضغط علي زر الفارة Mouse ثم إطلاق زر الفارة Mouse.
- حدث دخول مؤشر الفارة Mouse داخل مساحة الأداة.
- حدث خروج مؤشر الفارة Mouse من مساحة الأداة.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع الأداة التي نريد معالجة حدث التعامل مع الفارة Mouse عليها.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي MouseListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
mousePressed()	تنفذ هذه الدالة Method عند الضغط علي زر الفارة Mouse دون إطلاقه Release أي أننا لا زلنا ضاغطين علي الزر
mouseReleased()	تنفذ هذه الدالة عند إطلاق Release زر الفارة Mouse
mouseClicked()	تنفذ هذه الدالة Method عند إكمال الضغط علي الفارة Mouse بعد الضغط علي زر الفارة Mouse ثم إطلاق زر الفارة Mouse
mouseEntered()	تنفذ هذه الدالة Method عند دخول مؤشر الفارة Mouse داخل مساحة الأداة
mouseExited()	تنفذ هذه الدالة Method عند خروج مؤشر الفارة Mouse من مساحة الأداة

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة addMouseListener( ).

ملحوظة:

يجب أن تعيد تعريف جميع هذه الدوال Methods حتى لو لم تكتب فيها شيئاً لأن كلها دوال مجردة abstract methods ، ولذلك إذا نسيت كتابة تعريف دالة Method أو أخطأت فى كتابة الدالة Method بطريقة صحيحة - مع ملاحظة الحرف الصغير والكبير- ، فإنك تحصل على رسالة خطأ عند ترجمة Compile البرنامج لأنه فى هذه الحالة تكون الفصيلة class التى أنشأتها باسم MyMouseListener مثلاً ، فصيلة مجردة abstract class ولا يمكن أن تنشئ منها هدفاً object.

المثال التالي يوضح النقاط السابقة.

مثال (22): معالجة حدث الضغط علي الفارة Handling Mouse Event:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي أداة عنوان JLabel بحيث أن الضغط علي زر الفارة Mouse يؤدي إلي تغيير لون خلفية أداة العنوان JLabel إلي اللون الأصفر ، أما إطلاق Release زر الفارة Mouse فيؤدي إلي تغيير لون خلفية أداة العنوان JLabel إلي اللون الأزرق ، أما دخول مؤشر الفارة Mouse داخل مساحة أداة العنوان JLabel فيؤدي إلي تغيير لون خلفية أداة العنوان JLabel إلي اللون الأحمر ، أما خروج مؤشر الفارة Mouse من مساحة أداة العنوان JLabel فيؤدي إلي تغيير لون خلفية أداة العنوان JLabel إلي اللون الأخضر.

ثانياً: كود البرمجة:

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.
5.  public class MouseHandler extends JFrame
6.  {
7.   JLabel label;
8.
9.   MouseHandler()
10.  {
11.      Container c = getContentPane();
12.      c.setLayout(new BorderLayout());
13.
14.      label = new JLabel("Mouse");
15.      MyMouseListener listen = new MyMouseListener();
16.      label.addMouseListener(listen);
17.      label.setOpaque(true);
18.      c.add("Center",label);
```

```
19.
20.         setTitle("event");
21.         setSize(500,500);
22.         setVisible(true);
23.   }
24.
25.   public static void main(String args[])
26.   {
27.         MouseHandler event = new MouseHandler();
28.   }
29.
30.   class MyMouseListener implements MouseListener
31.   {
32.         public void mouseClicked(MouseEvent e)
33.         {
34.               label.setBackground(Color.BLUE);
35.         }
36.
37.         public void mouseEntered(MouseEvent e)
38.         {
39.               label.setBackground(Color.RED);
40.         }
41.
42.         public void mouseExited(MouseEvent e)
43.         {
44.               label.setBackground(Color.GREEN);
45.         }
46.
47.         public void mousePressed(MouseEvent e)
48.         {
49.               label.setBackground(Color.YELLOW);
50.         }
51.
52.         public void mouseReleased(MouseEvent e)
```

```
53.     {
54.         label.setBackground(Color.BLUE);
55.     }
56. }
57.}
```

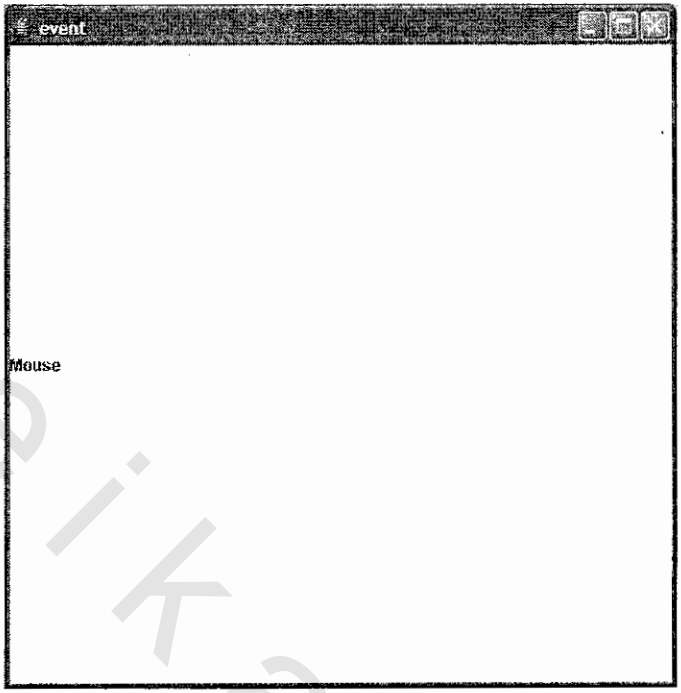
ثالثاً: شرح الكود:

- في السطر رقم 15 يتم إنشاء هدف object من نوع الفصيلة MyMouseListener حيث نستخدم هذا الهدف object كمستمع لحدث التعامل مع الفارة Mouse.
- في السطر رقم 16 يتم إضافة مستمع الحدث Event Listener لأحداث الفارة Mouse عن طريق الدالة ( )addMouseListener.
- فى بناء الفصيلة MyMouseListener كما في السطور من 30 إلى 56 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع الفارة Mouse بحيث أن جميع الدوال Methods تقوم بتغيير لون خلفية أداة العنوان JLabel إلي اللون المطلوب.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (13-17) وتأكد من عمل البرنامج بالشكل المطلوب.

ملحوظات:

بالطبع لابد أن تعيد تعريف الخمسة دوال Method فى الفصيلة MyMouseListener ، وإذا لم ترد ذلك فقم بعمل التعديل التالى على الفصيلة MyMouseListener:

```
class MykeyListener extends MouseAdapter
{
   public void mouseClicked(MouseEvent e)
        {
            label.setBackground(Color.BLUE);
        }
}
```



الشكل (17-13) حدث الفارة Mouse

أحداث تحريك الفارة Mouse Motion Events:

تتوافر الأحداث التالية عند تحريك الفارة Mouse:

- حدث تحريك الفارة Mouse.
- حدث سحب Drag الفارة Mouse.

خطوات معالجة الحدث Event Handling:

- لا بد من إنشاء هدف object من نوع الأداة التي نريد معالجة حدث التعامل مع تحريك الفارة Mouse عليها.
- لا بد من إنشاء فصيلة class جديدة تنفذ الـ interface المسمي MouseMotionListener.
- لا بد لهذه الفصيلة class الجديدة أن تقوم بتنفيذ الدوال Method الموجودة في الجدول التالي:

الدالة Method	الاستخدام
mouseMoved()	تنفذ هذه الدالة Method عند تحريك الفارة Mouse
mouseDragged()	تنفذ هذه الدالة عند سحب Drag الفارة Mouse

- وبداخل كل دالة Method تتم كتابة كود البرمجة الذي نريد تنفيذه.
- لا بد من إنشاء هدف object من نوع الفصيلة class التي أنشأناها في الخطوة الثانية.
- لا بد من إضافة هذا الهدف object كمستمع للحدث Event Listener إلي الأداة المطلوب معالجة أحداثها عن طريق الدالة ( )addMouseMotionListener.

ملحوظة:

يجب أن تعيد تعريف جميع هذه الدوال Methods حتى لو لم تكتب فيها شيئاً لأن كلها دوال مجردة abstract methods ، ولذلك إذا نسيت كتابة تعريف دالة Method أو أخطأت فى كتابة الدالة Method بطريقة صحيحة - مع ملاحظة الحرف الصغير والكبير- ، فإنك تحصل على رسالة خطأ عند ترجمة Compile البرنامج لأنه فى هذه الحالة تكون الفصيلة class التى أنشأتها باسم MyMouseListener مثلاً ، فصيلة مجردة abstract class ولا يمكن أن تنشئ منها هدفاً object.

المثال التالي يوضح النقاط السابقة.

مثال (24): معالجة حدث تحريك الفارة Handling Mouse Motion Event:

أولاً: هدف المثال:

نريد إنشاء برنامج يحتوي علي أداة عنوان JLabel بحيث أن سحب Drag الفارة Mouse يؤدي إلي تغيير لون خلفية أداة العنوان JLabel إلي اللون الأصفر ، أما تحريك الفارة Mouse فيؤدي إلي تغيير لون خلفية أداة العنوان JLabel حسب موضع الفارة Mouse في نافذة البرنامج.

ثانياً: كود البرمجة:

```
1. import javax.swing.*;
2. import java.awt.*;
3. import java.awt.event.*;
4.
5. public class MouseMotionHandler extends JFrame
6. {
7.  JLabel label;
8.
9.  MouseMotionHandler()
10. {
11.      Container c = getContentPane();
12.      c.setLayout(new BorderLayout());
13.
14.      label = new JLabel("Mouse");
15.      MyMouseListener listen = new MyMouseListener();
16.      label.addMouseMotionListener(listen);
17.      label.setOpaque(true);
18.      c.add("Center",label);
19.
20.      setTitle("event");
21.      setSize(500,500);
22.      setVisible(true);
23. }
24.
25. public static void main(String args[])
26. {
27.      MouseMotionHandler event = new
    MouseMotionHandler();
28. }
29.
30. class MyMouseListener implements
    MouseMotionListener
31. {
```

```
32.     public void mouseMoved(MouseEvent e)
33.     {
34.         int x = e.getX();
35.         int y = e.getY();
36.
37.         if ( x < 0 )
38.         {
39.             x = 0;
40.         }
41.
42.         if ( x > 255 )
43.         {
44.             x = 255;
45.         }
46.
47.         if ( y < 0 )
48.         {
49.             y = 0;
50.         }
51.
52.         if ( y > 255 )
53.         {
54.             y = 255;
55.         }
56.
57.         label.setBackground(new Color(x,y,0));
58.     }
59.
60.     public void mouseDragged(MouseEvent e)
61.     {
62.         label.setBackground(Color.YELLOW);
63.     }
64. }
65.}
```

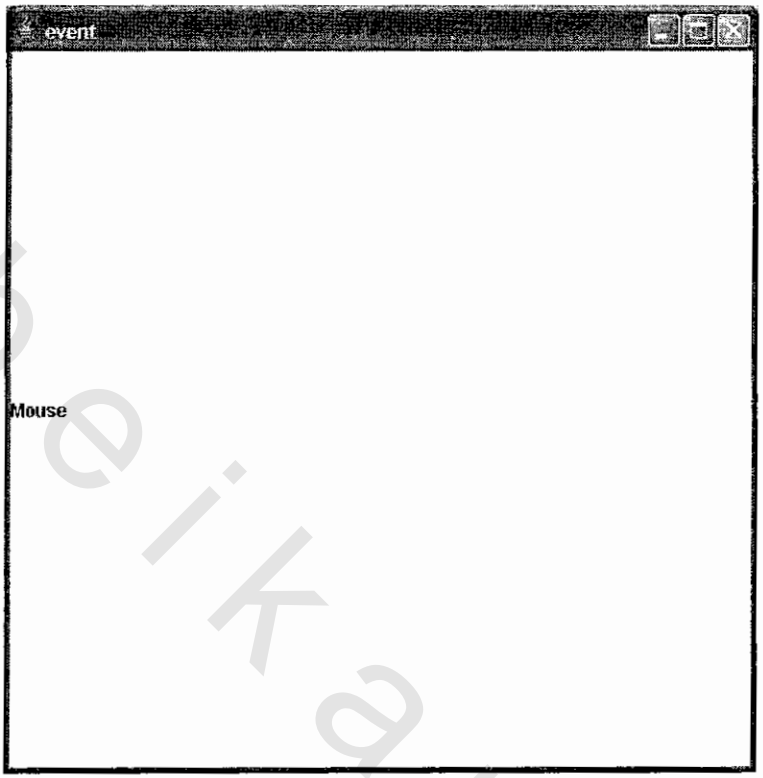
ثالثاً: شرح الكود:

- في السطر رقم 15 يتم إنشاء هدف object من نوع الفصيلة MyMouseListener حيث نستخدم هذا الهدف object كمستمع لحدث تحريك الفارة Mouse.
- في السطر رقم 16 يتم إضافة مستمع الحدث Event Listener لأحداث تحريك الفارة Mouse عن طريق الدالة addMouseMotionListener( ).
- فى بناء الفصيلة MyMouseListener كما في السطور من 30 إلى 64 ، نجد أننا قمنا بتعريف جميع الدوال Methods السابق شرحها لمعالجة أحداث التعامل مع تحريك الفارة Mouse بحيث أن جميع الدوال Methods تقوم بتغيير لون خلفية أداة العنوان JLabel إلي اللون المطلوب. لاحظ أننا في الدالة mouseMoved( ) قمنا بمعرفة الإحداثي الأفقي للمؤشر عن طريق الدالة getX( ) وبالمثل تم معرفة الإحداثي الرأسي للمؤشر عن طريق الدالة getY( ) ثم يتم التحقق من أن قيم المتغيرين x و y لا بد أن يكونوا في المدي من صفر إلي 255 ، ثم يتم تغيير لون خلفية أداة العنوان JLabel بالشكل المطلوب.
- قم بترجمة البرنامج وتنفيذه لتظهر لك شاشة البرنامج كما هو واضح في الشكل (14-17) السابق وتأكد من عمل البرنامج بالشكل المطلوب.

ملحوظات:

بالطبع لابد أن تعيد تعريف الدالتين Methods فى الفصيلة MyMouseListener وإذا لم ترد ذلك فقم بعمل التعديل التالى على الفصيلة MyMouseListener:

```
class MykeyListener extends MouseMotionAdapter
{
   public void mouseDragged(MouseEvent e)
        {
            label.setBackground(Color. YELLOW);
        }
}
```



الشكل (17-14) حدث تحريك الفارة Mouse

ملخص الفصل:

- تعلمنا في هذا الفصل كيفية معالجة أحداث Event Handling العديد من أدوات مكتبة Swing.
- في الفصل القادم نتعلم - بإذن الله - كيفية تصميم وتنفيذ العديد من التطبيقات بحيث تعتبر مثل تمرين لك للتدرب علي جميع الأوامر التي شرحناها في الفصول السابقة ، فتابع معنا الفصل القادم.