Poo Laboratoare[1]

Contents

[1] http://www.google.ro/

Laborator9

Programare Orientată pe Obiecte: Laborator 9

1 Programare cu evenimente JFC

1.1 Overview

Folosirea interfețelor grafice implica și o abordare specială în programare, numită programare orientată pe evenimente (**Event Oriented Programming**) sau programare ghidată de evenimente (**Event Driven Programming**). În această concepție, obiectele din program pot fi surse de evenimente sau consumatoare (ascultatoare) de evenimente (**Event Listeners**). Evenimentele înseși sunt obiecte (instanțe ale unor clase de evenimente) generate de surse și interceptate de consumatori. În limbajului Java, evenimentele sunt instanțe ale claselor derivate din **AWTEvent**. Clasele consumatoare de evenimente sunt cele care conțin metodele prin care aplicația reacționează la evenimentele respective.

În JDK se aplică modelul de evenimente bazat pe delegare (**Delegation Event Model**). Conform acestuia, clasele de evenimente sunt organizate într-o ierarhie, fiind specializate pe diferite tipuri de evenimente. Se disting evenimente de nivel inferior (cele generate de diferite componente) și evenimente abstracte (care reprezintă diferite evenimente conceptuale, independente de componenta care le-a generat). Fiecarei clase de evenimente îi corespunde o interfață pe care trebuie să o implementeze clasele concepute de programator, care intercepteză și tratează evenimentele respective. Interfețele sunt, de asemenea organizate într-o ierarhie, similară celei a claselor. Pentru unele interfețe s-au realizat și clase prototip care le implementează, numite adaptoare. Clasele de evenimente, interfețele și adaptoarele se găsesc în pachetul **java.awt.event**.

Conform cu modelul delegării, fiecare instanță a unei clase consumatoare de evenimente trebuie înregistrată la obiectul care generează evenimentele respective (sursa de evenimente). Înregistrarea se face prin apelul metodei **addNumeInterfataAscultatoare()**, unde numele interfteței poate fi oricare din cele prezentate mai jos. Sursa transmite evenimentele numai consumatorilor (ascultatorilor) înregistrați. La recepționarea unui eveniment, în instanța consumatoare este invocata metoda de tratare a evenimentului respectiv.

1.2 Evenimente pentru componente și containere AWT

ComponentEvent este clasa de bază. Această clasă include evenimente de tip notificare în caz că o componentă își schimbă dimensiunea sau vizibilitatea sau dacă se întamplă evenimente cu mouse-ul sau tastatura care afectează componenta respectivă. Apar evenimente **ContainerEvent** dacă se adaugă, respectiv se șterg componente dintr-un container.

1.2.1 ComponentEvent

- Produs de: **toate componentele**
- Interfețe Listener: **ComponentListener**
- Metode de tratare:
  - **componentResized()**
  - **componentMoved()**
  - **componentShown()**
  - **componentHidden()**

1.2.2 FocusEvent

- Produs de: **toate componentele**
- Interfețe Listener: **FocusListener**
- Metode de tratare:
  - **focusGained()**
  - **focusLost()**

1.2.3 KeyEvent

- Produs de: **toate componentele**
- Interfețe Listener: **KeyListener**
- Metode de tratare:
  - **keyTyped()**
  - **keyPressed()**
  - **keyReleased()**

1.2.4 MouseEvent

- Produs de: **toate componentele**
- Interfețe Listener:
  - **MouseListener**
  - **MouseMotionListener**
- Metode de tratare:
  - **mouseClicked()**
  - **mousePressed()**
  - **mouseReleased()**
  - **mouseEntered()**
  - **mouseExited()**
  - **mouseDragged()**
  - **mouseMoved()**

1.2.5 ContainerEvent

- Produs de: **toți containerii**
- Interfețe Listener: **ContainerListener**
- Metode de tratare:
  - **componentAdded()**
  - **componentRemoved()**

1.3 Evenimente specifice componentelor AWT

1.3.1 ActionEvent

- Produs de:
  - **TextField**
  - **MenuItem**
  - **List**
  - **Button**
- Interfețe Listener: **ActionListener**
- Metode de tratare: **actionPerformed()**

1.3.2 ItemEvent

- Produs de:
  - **List**
  - **CheckBox**
  - **Choice**
  - **CheckboxMenuItem**
- Interfețe Listener: **ItemListener**
- Metode de tratare: **itemStateChanged()**

1.3.3 AdjustmentEvent

- Produs de:
  - **ScrollPane**
  - **Scrollbar**
- Interfețe Listener: **AdjustmentListener**
- Metode de tratare: **adjustmentValueChanged()**

1.3.4 TextEvent

- Produs de:
  - **TextArea**
  - **TextField**
- Interfețe Listener: **TextListener**
- Metode de tratare: **textValueChanged()**

1.3.5 WindowEvent

- Produs de:
  - **Frame**
  - **Dialog**
- Interfețe Listener: **WindowListener**

- Metode de tratare:
  - **windowOpened()**
  - **windowClosing()**
  - **windowClosed()**
  - **windowIconified()**
  - **windowDeiconified()**
  - **windowActivated()**
  - **windowDeactivated()**

1.4 Clase Adapter

Multe din interfețele de tip Listener conțin mai multe metode de tratare a evenimentelor. Din nefericire acesta înseamnă că, clasa care este interesată în aceste tipuri de evenimente **trebuie să implementeze toate metodele** declarate în interfață. Spre ajutorul programatorilor au fost create niște **clase de tip Adapter** care implementează aceste metode. Deci dacă este nevoie de o anumită interfață, se va deriva din aceasta clasa de tip Adapter implementată special pentru interfața dorită de tip Listener. De exemplu, pentru evenimentele cu mouse-ul există clasa **MouseAdapter** care implementeză interfața **MouseListener**.

```
public void mouseClicked( MouseEvent e ) {};
public void mousePressed( MouseEvent e ) {};
public void mouseReleased( MouseEvent e ) {};
public void mouseEntered( MouseEvent e ) {};
public void mouseExited( MouseEvent e ) {};
```

Exemplu de utilizare:

```
oComponenta.addMouseListener(
       new MouseAdapter() {
              public void MousePressed( MouseEvent e ) {
                     obiect.show();
                     ...
              }
       }
);
```

În construcția de mai sus s-a utilizat o clasă încuibată.

1.5 Exemplu

Aplicația de mai jos afișează o fereastră având o casetă text și trei butoane cu etichetele 1, 2, 3. La apasărea butoanelor în caseta text apare eticheta butonului apăsat.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class MyFrame extends JFrame implements ActionListener
{
       private JButton butoane[];
       private JTextField tf;
       public MyFrame()
       {
                     setLayout( new FlowLayout() );
                     tf = new JTextField( 10 );
```

```
                    add( tf );
                    butoane = new JButton[ 3 ];
                    for( int i = 0; i < butoane.length; i ++ )
                    {
                          butoane[ i ] = new JButton( Integer.toString( i ) );
                          butoane[ i ].addActionListener( this );
                          add( butoane[ i ] );
                    }
                    // Tratarea evenimentelor cu fereastra
                    addWindowListener( new WindowAdapter()
                    {
                          public void windowClosing( WindowEvent e ){
                          setVisible( false );
                          System.exit( 0 );
                          }
                    });
      }
      public void actionPerformed( ActionEvent arg0 )
      {
            String s = arg0.getActionCommand();
            tf.setText( s );
      }
      public static void main( String[] args )
      {
            MyFrame f = new MyFrame();
            f.setBounds( 1, 1, 200, 100 );
            f.setVisible( true );
      }
}
```

2 Probleme de laborator

Atentie! Se vor utiliza obiecte grafice Swing (JFrame, JButton, JTextField, etc)

2.1 Problema 1

(2.1 puncte) Program pentru afișarea unui buton cu inscripția *Click Me* și afișarea unei casete de dialog cu titlul *Event Fired* la fiecare clic pe buton (cu mouse-ul). Afișarea casetei de dialog se face astfel:

```
JOptionPane.showMessageDialog( new JFrame(), ``'',
            ``Event Fired !'', JOptionPane.PLAIN_MESSAGE );
```

Se vor examina pe rând următoarele variante de definire a clasei ascultător la evenimente generate de buton:

- *(0.3 puncte)* Cu trei clase separate:

1. clasa ascultator
2. clasa derivată din JFrame care conține și butonul
3. clasa cu **main()** (care afișează fereastra)

- *(0.3 puncte)* Cu două clase:

1. clasa ascultător
2. clasa derivată din JFrame și care conține metoda **main()**

- *(0.3 puncte)* Cu o singură clasă: clasa ascultător cu nume inclusă în clasa ce conține metoda **main()**
- *(0.3 puncte)* Cu o singură clasă: clasa ascultător anonimă, inclusă într-un bloc (metoda **addActionListener()**) din clasa ce conține metoda **main()**
- *(0.3 puncte)* Cu două clase:

1. o subclasă a clasei JButton care conține și metoda **actionPerformed()**
2. clasa care conține metoda **main()**

- *(0.3 puncte)* Cu două clase:

1. clasa ascultător inclusă într-o subclasă a clasei JFrame
2. clasa ce conține metoda **main()**

- *(0.3 puncte)* O singură clasă care extinde JFrame și implementează interfața ActionListener (clasa este și generator și ascultător la evenimente)

2.2 Problema 2

(1.9 puncte) Program pentru afișarea unui câmp text modificabil, cu textul inițial *black*. La introducerea unui nume de culoare acceptat (*red*, *blue*, etc.) se modifică culoarea textului afișat. Pentru un alt nume de culoare se revine la culoarea negru pentru textul afișat. Metoda **actionPerformed()** apelează metoda **setForeground()** cu parametrul corespunzător textului preluat din câmp (cu metoda **getText()**). Se poate folosi un vector de constante:

```
Color cc[] = { Color.red, Color.blue, Color.cyan, Color.orange, Color.magenta };
```

2.3 Problema 3

(3.5 puncte) Program pentru afișarea a două câmpuri text nemodificabile și a trei butoane radio JRadioButton notate cu A, B, C și grupate într-un grup de butoane ButtonGroup. În primul câmp se afișează întrebarea *Ce alegeți?*. După apăsarea unui buton se afișează în al doilea câmp numele selectat (A, B, C). Toate cele trei butoane radio au un singur obiect ascultător de tip ActionListener care modifică conținutul câmpului rezultat (cu răspunsul la întrebare). Programul corespunde unei întrebări cu un singur răspuns posibil dintr-un test grilă.

2.4 Problema 4

(2.5 puncte) Modificați programul anterior pentru afișarea unor casete cu bifare JCheckBox și afișarea numelor casetelor selectate în câmpul text. Programul corespunde unei întrebări cu mai multe răspunsuri posibile dintr-un test grilă. Cum se poate interzice bifarea repetată a unor casete? Toate cele trei casete au un singur ascultător de tip ItemListener, cu o metodă **itemStateChanged()** și cu argument de tip ItemEvent. Metoda **getSource()** apelată pentru un obiect eveniment are ca rezultat numele casetei care a generat evenimentul (care diferă de textul afișat lângă casetă și transmis la construirea obiectului JCheckBox).

2.5 Problema 5 (bonus)

(3 puncte) Program care afișează patru câmpuri text și un buton *Open*. În primele trei câmpuri text operatorul introduce (în orice ordine) numele discului suport (A:, C:, D:), calea (secvența de directoare) și respectiv numele unui fișier. La apăsarea pe buton se transferă în cel de-al patrulea câmp text cele trei componente pentru a se afișa calea completă la fișier. Variante:

1. La buton se înregistrează trei obiecte ascultător, care transferă fiecare textul dintr-un camp text (plus caracterul '\') în câmpul rezultat.

2. La buton se înregistrează un singur obiect ascultător, care compune datele din primele 3 câmpuri text și le afișează în câmpul rezultat.

Trecerea de la un câmp text la altul se poate face și cu tasta *Tab*.