TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 10

Evenimente generate de componentele AWT

I. SCOPUL LUCRĂRII

Lucrarea de față are rolul de a prezenta şi familiariza studentul cu modul de tratare a evenimentelor generate de componentele unei interfeței grafice utilizator.

La sfârşitul acestei lucrări, studentul va avea posibilitatea să scrie applet-uri Java cu interfață grafică complet funcțională.

II. NOȚIUNI TEORETICE

1. Clasele Eveniment

Subclasele clasei **java.awt.AWTEvent** reprezintă diferite tipuri de evenimente care pot fi generate de diferitele componente AWT. Aceste subclase conțin toate informațiile necesare cu privire la activitatea care a declanşat evenimentul. Iată care sunt:

- **ActionEvent** – generat de activarea unei componente
- **AdjustmentEvent** – generat prin ajustarea componentelor ajustabile cum ar fi "scroll bars"
- **ContainerEvent** – generat când componentele sunt adăugate sau scoase dintr-un container
- **FocusEvent** – generat când o componentă primeşte sau pierde focus-ul asupra ei
- **ItemEvent** – generat când o opțiune este selectată dintr-o listă (Choice, List) sau Checkbox
- **KeyEvent** – generat de activitatea tastaturii
- **MouseEvent** - generat de activitatea cu mouse-ul
- **PaintEvent** - generat când o componentă este desenată
- **TextEvent** – generat când o componentă text este modificată
- **WindowEvent** – generat de activitatea cu fereastra (iconificarea, deziconificarea)

2. Event Listeners

Există două moduri de a trata evenimentele prezentate mai sus. Primul este de a delega tratarea evenimentului unui obiect care "ascultă" (**listens**). Al doilea este de a permite explicit componentei care generează evenimentele să-şi manipuleze propriile evenimente.

Iată ce presupune prima variantă. Să considerăm următorul exemplu. Fie un buton într-un applet. Când butonul este apăsat, un eveniment **ActionEvent** va fi trimis unei instanțe a clasei **MyActionListener**. Clasa **MyActionListener** implementează interfața **ActionListener**, garantându-se astfel prezența metodei **actionPerformed()**.

```
class MyActionListener implements ActionListener  {
       public void actionPerformed(ActionEvent e)  {
              System.out.println("Action performed.");
        }
 }

public class ListenerTest extends Applet {
          public void init()  {
              Button btn=new Button("OK");
              MyActionListener myAl=new MyActionListener();
              btn.addActionListener(myAl);
              add(btn);
          }
}
```

Putem rezuma tehnica de lucru în patru paşi:

1. Se creează o clasă „listener" care implementează interfața **ActionListener** (pt. exemplul nostru).
2. Se construieşte componenta.
3. Se construieşte o instanţă a clasei „listener".
4. Se apelează metoda **addActionListener()** pentru componentă, transmițându-i ca parametru obiectul „listener".

Există 11 tipuri de "listeners", fiecare reprezentat de o interfață. În continuare vom prezenta numai o parte dintre acestea, împreună cu metodele din interfață şi cu metodele **addXXXListener()** corespunzătoare.
(Pentru celelalte interfețe, anume AdjustmentListener, ComponentListener, ContainerListener, FocusListener, TextListener, WindowListener se vor căuta informații în documentația jsdk).

Interfață	**Metodele din interfață**	**Metoda add**
ActionListener	actionPerformed(ActionEvent e)	addActionListener( )
ItemListener	itemStateChanged(ItemEvent e)	addItemListener( )
KeyListener	keyPressed(keyEvent e) keyReleased(keyEvent e) keyTyped(keyEvent e)	addKeyListener( )
MouseListener	mouseClicked(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e)	addMouseListener( )
MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)	addMouseMotionListener( )

Reversul metodei **addXXXListener()** este metoda **removeXXXListener()** care înlătură pentru o componentă obiectul ataşat ei care "asculta" evenimentele generate de aceasta. Pentru exemplul anterior avem:

```
btn.removeActionListener(myAl);
```

Aşa cum deja am menţionat, o a doua tehnică de lucru permite explicit componentei care generează evenimentele să-şi manipuleze propriile evenimente. În continuare vom prezenta câteva exemple sugestive.

```
// App4.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class App4 extends Applet implements ActionListener
{
   Button but1,but2;
   int x=0,y=0;
   public void init()
   {
     but1=new Button("Deseneaza");
     add(but1);
     but2=new Button("Sterge");
     add(but2);
     but1.addActionListener(this);
     but2.addActionListener(this);
   }

   public void actionPerformed(ActionEvent e)
   {
        Button b=(Button)e.getSource();
        if (b==but1)
          {
             x=150;y=150;
             repaint();
          }
        else if(b==but2)
          {
             x=y=0;
             repaint();
          }
   }
   public void paint(Graphics g)
   {
       setBackground(Color.blue);
       setForeground(Color.magenta);
       if(x>0)
           g.fillOval(x,y,100,100);
   }
}
```

```
// App4.html
<html>
<body>
<applet code=App4.class width=500 height=400>
</applet>
</body>
</html>

//App5.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;

public class App5 extends Applet
                  implements ActionListener, MouseListener
{
   Button but;
   Vector points;
   public void init()
   {
     but=new Button("Clear all");
     add(but);
     but.addActionListener(this);
     addMouseListener(this);
     points=new Vector();
   }
   public void actionPerformed(ActionEvent e)
   {
        Button b=(Button)e.getSource();
        if (b==but)
          {
             points.setSize(0);
             repaint();
          }

   }
   public void mouseClicked(MouseEvent e)
   {
      e.consume();
      points.addElement(new Point(e.getPoint()));
      repaint();
   }
   public void mouseEntered(MouseEvent e)
   {
   }
   public void mouseExited(MouseEvent e)
   {
   }
   public void mousePressed(MouseEvent e)
   {
   }
   public void mouseReleased(MouseEvent e)
   {
   }
```

```
   public void paint(Graphics g)
   {
       setBackground(Color.orange);
       setForeground(Color.black);
       for(int i=0;i<points.size();i++)
       {

           Point p=(Point)points.elementAt(i);
           g.fillRect(p.x,p.y,70,30);
       }
   }
}

//App5.html
<html>
<body>
<applet code=App5.class width=500 height=500>
</applet>
</body>
</html>

//App6.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;

public class App6 extends Applet
                  implements ActionListener, MouseListener
{
   Button but;
   Vector lines;
   int x1,y1,x2,y2;
   public void init()
   {
     but=new Button("Clear all");
     add(but);
     but.addActionListener(this);
     addMouseListener(this);
     lines=new Vector();
   }
   public void actionPerformed(ActionEvent e)
   {
        Button b=(Button)e.getSource();
        if (b==but)
          {
             lines.setSize(0);
             repaint();
          }
   }
   public void mouseClicked(MouseEvent e)
   {
   }
```

```
    public void mouseEntered(MouseEvent e)
    {
    }
    public void mouseExited(MouseEvent e)
    {
    }
    public void mousePressed(MouseEvent e)
    {
        e.consume();
        x1=e.getX();
        y1=e.getY();
    }
    public void mouseReleased(MouseEvent e)
    {
       e.consume();
       x2=e.getX();
       y2=e.getY();
       lines.addElement(new Rectangle(x1,y1,x2,y2));
       repaint();
    }

    public void paint(Graphics g)
    {
        for(int i=0;i<lines.size();i++)
        {

            Rectangle r=(Rectangle)lines.elementAt(i);
            g.drawLine(r.x,r.y,r.width,r.height);
        }
    }
}

//App6.html
<html>
<body>
<applet code=App6.class width=500 height=500>
</applet>
</body>
</html>

//App7.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class App7 extends Applet
                  implements ActionListener, ItemListener
{
    Button but;
    Choice ch;
    String color;

    public void init()
    {
      but=new Button("Draw");
```

```
      add(but);
      but.addActionListener(this);
      ch=new Choice();
      ch.addItem("red");
      ch.addItem("yellow");
      ch.addItem("pink");
      add(ch);
      ch.addItemListener(this);
      color=new String();
      setFont(new Font("Arial",Font.PLAIN,14));
   }

   public void actionPerformed(ActionEvent e)
   {
        Button b=(Button)e.getSource();
        if (b==but)
          {
             repaint();
          }
   }

   public void itemStateChanged(ItemEvent e)
   {
      color=ch.getSelectedItem();
   }

   public void paint(Graphics g)
   {
       setBackground(Color.black);
       if(color.compareTo("red")==0)
             g.setColor(Color.red);
       else if(color.compareTo("yellow")==0)
             g.setColor(Color.yellow);
       else if(color.compareTo("pink")==0)
             g.setColor(Color.pink);

       g.drawString(color,70,150);
   }
}

//App7.html
<html>
<body>
<applet code=App7.class width=200 height=300>
</applet>
</body>
</html>

//App8.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
```

```
public class App8 extends Applet implements ActionListener
{
   Button but;
   TextField tx;
   myFrame f;

   public void init()
   {
     setFont(new Font("Arial",Font.PLAIN,14));
     tx=new TextField("",10);
     add(tx);
     but=new Button("Open window");
     add(but);
     but.addActionListener(this);
   }

   public void actionPerformed(ActionEvent e)
   {
        Button b=(Button)e.getSource();
        if (b==but)
          {
             f=new myFrame("Little window",tx.getText());
             f.setSize(300,300);
             f.setVisible(true);
          }
   }
}

class myFrame extends Frame implements ActionListener
{
   String text;
   Button buton;
   public myFrame(String s,String tx)
   {
      super(s);
      text=tx;
      setLayout(new FlowLayout());
      buton=new Button("Close");
      add(buton);
      buton.addActionListener(this);
   }
   public void actionPerformed(ActionEvent e)
   {
        Button b=(Button)e.getSource();
        if (b==buton)
          {
            dispose();
          }
   }
   public void paint(Graphics g)
   {
      g.drawString(text,50,100);
   }
}
```

```
//App8.html
<html>
<body>
<applet code=App8.class width=300 height=300>
</applet>
</body>
</html>
```

III. MODUL DE LUCRU

1. Se editează codul sursă al programului Java folosind un editor de text disponibil(de ex., se poate utiliza Notepad).
2. Se salvează fişierul cu extensia **.java**. Fişierul trebuie salvat la următoarea locație: **c:\JBulider7\jdk1.3.1\bin**
3. Compilarea mini-aplicației Java se va face din linia de comandă. Se poate proceda astfel. Se deschide o fereastră MS-Dos: Start ->Run, se tipăreşte **command** în căsuța de text şi se apasă butonul OK. Printr-o schimbare de directoare şi subdirectoare se trece la locația: **c:\JBulider7\jdk1.3.1\bin.** Sau, se lansează WindowsCommander. Se trece la locația **c:\JBulider7\jdk1.3.1\bin.** Se deschide o fereastră MS-Dos: Commander ->Run Dos.
4. Pentru compilare, se tipăreşte la prompter **javac nume_fişier_sursă.java** şi se apasă Enter. De ex., dacă fişierul se numeşte Test.java, se va scrie **javac Test.java**. În cazul în care programul conține erori acestea vor fi semnalate şi afişate în fereastra MS-Dos, după care va apare iar prompter-ul. Dacă programul nu conține erori şi compilarea se face cu succes, atunci va apare numai prompter-ul.
5. Se editează fişierul .**html**. Se salvează în acelaşi director cu fişierul **.class** rezultat în urma compilării cu succes a fişierului sursă java.
6. Se încarcă fişierul **.html** într-un browser Web (ex., Internet Explorer). Alternativ se poate utiliza comanda **appletviewer nume.html**.

IV. TEMĂ

1. Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic.
2. Scrieți un applet Java care să conțină un grup de 2 componente CheckBox etichetate “red” şi “blue”, şi un buton etichetat “Clear all”. Utilizatorul va bifa una din opțiunile “red” sau “blue”. La efectuarea unui click cu mouse-ul pe suprafața aapplet-ului se va desena un cerc colorat roşu sau albastru în funcție de opțiunea aleasă. La apăsarea butonului “Clear all” se vor şterge toate cercurile desenate până în acel moment.