Mouse

Mouse este tratat automat de către majoritatea componentelor astfel că nu trebuie să ştiţi de el. Spre exemplu, dacă cineva dă clic pe un buton (**`JButton`**), veţi recepţiona un **`ActionEvent`**, dar nu este nevoie să ştiţi (şi n-ar trebui să vă pese) dacă aceasta s-a datorat unui clic cu mouse pe buton sau a fost cauzat de o apăsare de tastă "rapidă" (shortcut).

Graphica. Dacă desenaţi grafică proprie (d.e., într-un **`JPanel`**) şi aveţi nevoie să ştiţi dacă utilizatorul face clic, atunci trebuie să ştiţi despre evenimentele legate de mouse. Puteţi adăuga cu uşurinţă un "ascultător" pentru mouse la un **`JPanel`**.

1. Clase şi interfeţe importante

Clasele care urmează sunt definite în **java.awt.event**. Primele trei sunt cele mai folosite.

- **`MouseEvent`** – se trimite un obiect de tipul MouseEvent object tuturor ascultătorilor de mouse. Cele mai folositoare informaţii într-un MouseEvent sunt coordonatele x şi y ale cursorului mouse-ului.
- **`MouseListener`** – Interfaţă pentru apăsări şi eliberări de butoane de mouse, clic-uri, intrări şi ieşiri din zone.
- **`MouseMotionListener`** – Interfaţă pentru deplasări şi trageri (drag).
- **`MouseInputListener`** – Combinaţie de interfaţă între MouseListener şi MouseMotionListener.
- **`MouseAdapter`** – Clasă utilă pentru scrierea unui ascultător anonim pentru apăsări de butoane ale mouse, intrări în zone, …
- **`MouseMotionAdapter`** – Clasă utilă pentru scrierea unui ascultător anonim pentru deplasarea mouse.

1.1. MouseListener – tratează apăsări, eliberări, clicuri, intrări şi ieşiri.

Acest tip de ascultător de mouse este pentru evenimente care nu se întâmplă de obicei prea des – se apasă sau eliberează un buton al mouse sau mouse intră sau iese din zona componentei cu ascultător. Iată care sunt acţiunile pe care **MouseListener** le interceptează.

press	Unul dintre butoanele mouse a fost apăsat.
release	Unul dintre butoanele mouse a fost eliberat.
click	Un buton al mouse a fost apăsat şi eliberat fără a mişca mouse. Acesta este poate cel mai folosit.
enter	Cursorul mouse intră pe componentă. Folosit adesea pentru a schimba cursorul.
exit	Cursorul mouse iese de pe componentă. Adesea folosit pentru a restaura cursorul.

Pentru a asculta astfel de evenimente folosiţi **addMouseListener**.

1.1.1. Interfaţa MouseListener

Pentru a implementa interfaţa **MouseListener** trebuie să definiţi următoarele metode. Puteţi copia aceste definiţii în program şi construi un corp cu înţeles pentru metodele care sunt de interes.

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

Metoda este apelată	Atunci când utilizatorul efectuează acţiunea aceasta
`mouseClicked`	Un clic este rezultatul unei apăsări şi eliberări. Probabil cea mai comună

	metodă de scris.
`mousePressed`	A fost apăsat un buton al mouse (oricare dintre cele trei posibile)
`mouseReleased`	A fost eliberat un buton al mouse.
`mouseEntered`	Cursorul mouse intră pe o componentă. Aţi putea scrie aceasta pentru a schimba cursorul.
`mouseExited`	Cursorul mouse iese de pe o componentă. Aţi putea scrie aceasta pentru a restaura cursorul.

Pentru a obţine coordonatele mouse. Toate coordonatele sunt relative le colţul din stânga sus al componentei care are ascultătorul pentru mouse. Folosiţi următoarele metode din **MouseEvent** pentru a obţine coordonatele x şi y ale locului în care a apărut evenimentul mouse.

```
int getX() // returns the x coordinate of the event.
int getY() // returns the y coordinate of the event.
```

Pentru a testa clicuri duble. Folosiţi următoareametodădin **MouseEvent** pentru a obţine numărul de clicuri.

```
int getClickCount() // numarul de clicuri pe mouse
```

1.1.2. MouseMotionListener – tratează deplasările şi tragerile (drags).

Deplasări şi trageri (târâri – drags). La deplasarea mouse, interfaţa generează evenimente foarte repede. Dacă se apasă un buton al mouse la deplasare, aceasta se numeşte târâre (drag). Evenimentele de la o deplasare sau o târâre sunt generate foarte repede; ele pot fi interceptate prin adăugarea unui ascultător pentru mişcarea mouse.

Metode ale MouseMotionListener

Pentru a implementa un MouseMotionListener, trebuie definite următoarele metode:

```
public void mouseMoved(MouseEvent e) {}
public void mouseDragged(MouseEvent e) {}
```

Acesată metodă este apelată	**Atunci când utilizatorul**
`mouseMoved(...)`	Deplasează mouse în timp ce indicatorul sau este pe o componentă.
`mouseDragged(...)`	Ţârăşte mouse (deplasează mouse cu un buton apăsat).

1.2. Ascultători pentru mouse – Cum şi unde să scriem ascultători pentru mouse

Există câteva stiluri de folosire a ascultătorilor pentru mouse. Ascultătorii sunt de obicei adăugaţi la panouri grafice care au o metodă `paintComponent`.

Ascultarea din panoul însuşi

Este uzual ca panoul să asculte propriile evenimente. Spre exemplu,

```
class DrawingPanel extends JPanel implements MouseListener
{

   public DrawingPanel()
   { // Constructor
      this.addMouseListener(this);
      . . .
   }

   public void paintComponent(Graphics g)
   {
      . . .
   }
   . . .
   public void mousePressed(MouseEvent e) {. . .}
```

```
    public void mouseReleased(MouseEvent e) {. . .}
    public void mouseClicked(MouseEvent e) {. . .}
    . . .
}
```

Panoul poate comunica schimbările în exterior prin (1) construirea unei subclase, (2) furnizarea de metode *getter*, sau (3) furnizarea unui obiect "model" pentru constructor.

Ascultarea din afara panoului

Puteti crea un panou şi să doriţi ca ascultătorii să fie în afara lui, pentru că aşa este mai convenabil să interacţionăm cu ei. Dacă aveţi un singur asemenea panou, atunci puteţi implementa interfeţele ascultător de mouse în clasa care nu este panou şi scrie toate metodele ascultator. Spre exemplu,

```
public class MyClass implements MouseListener {
   . . .
   DrawingPanel drawing = new DrawingPanel();
   drawing.addMouseListener(this);
   . . .

   public void mousePressed(MouseEvent e) {. . .}
   public void mouseReleased(MouseEvent e) {. . .}
   public void mouseClicked(MouseEvent e) {. . .}
   . . .
}

class DrawingPanel extends JPanel {
   public void paintComponent(Graphics g) {
      . . .
   }
   . . .
}
```

Aceasta necesită metode *setter* în clasa DrawingPanel astfel încât să poată fi modificat ceea cese desenează. Sau se poate transmite unui constructor pentru DrawingPanel un obiect pentru "model" care să-i permită să obtina valorile necesare lui `paintComponent`.

Ca mai sus cu ascultători anonimi

Dacă doriţi să ascultaţi un singur fel de eveniment, atunci este uşor să folosiţi clasele MouseAdapter sau MouseMotionAdapter pentru a crea un ascultător anonim. Spre exemplu, pentru a asculta clic-uri pe mouse,

```
p.addMouseListener(new MouseAdapter()
{
   public void mouseClicked(MouseEvent e)
   {
      x = e.getX();
      y = e.getY();
   }
});
```

1.3. Butoane ale mouse şi taste modificatoare – Cum să verificăm care butoane ale mouse a fost apăsate.

Butoanele mouse. Java suportă până la trei butoane pentru mouse. Chiar dacă mouse nu are trei butoane separate, se poate simula unul dintre butoane apăsând o tastă modificatoare când se apasă butonul mouse.

Obiectul MouseEvent care este transmis ascultătorului conţine informaţii care vă permit să întrebaţi care combinaţie de butoane a fost apăsată pentru apariţia evenimentului. *Controalele de defilare ale mouse (scroll controls)* au fost prima dată suportate în Java 2 SDK 1.4.

Există două modalităţi de testare a butoanelor mouse şi a tastelor modificatoare:

- Să folosiţi *metode* pentru a afla starea butoanelor mouse şi a tastelor modificatoare.
- Să folosiţi *măştile pe biţi* definite în clasa InputEvent pentru a vedea biţii modificatori din MouseEvent. Aceasta este o cale *foarte rapidă* de verificare, în special pentru combinaţii complexe de modificatori şi butoane; dar trebuie să înţelegeţi operatorii pe biţi (| & ^ ~ >> >>> <<).

1.1.3. Folosirea metodelor pentru a verifica butoanele mouse

Pentru a verifica ce buton a fost apăsat folosiţi una dintre metodele statice din SwingUtilities. Aceste metode întorc true dacă s-a folosit butonul corespunzător. Remarcaţi că mai mult de una va fi true dacă a fost folosit mai mult de un buton simultan.

- `boolean SwingUtilities.isLeftMouseButton(MouseEvent anEvent)`
- `boolean SwingUtilities.isMiddleMouseButton(MouseEvent anEvent)`
- `boolean SwingUtilities.isRightMouseButton(MouseEvent anEvent)`

1.1.4. Folosirea metodelor pentru a verifica tastele modificatoare

Pentru a verifica care atstă modificatoare a fost apăsată, folosiţi aceste metode dinclasa MouseEvent:

```
boolean isAltDown()     // true daca s-a apasat Alt – butonul mijlociu al mouse
boolean isControlDown() // true daca s-a apasat Control
boolean isShiftDown()   // true daca s-a apasat Shift
boolean isAltGraphDown()// true daca s-a apasat Alt Graphics
boolean isMetaDown()    // true daca s-a apasat Meta sau butonul dreapta
```

Spre exemplu, în interiorul ascultătorului de mouse am putea testa ca mai jos ca să vedem dacă s-a apăsat butonul dreapta în timp de era apăsată tasta Shift. Presupunem că e este un obiect MouseEvent.

```
if (SwingUtilities.isRightMouseButton(e) && e.isShiftDown())
   ...
```

1.1.5. Folosirea măştilor pe biţi pentru a verifica butoanele mouse şi tastele modificatoare

Folosiţi metoda getModifiers() din MouseEvent pentru a obţine o mască pe biţi care spune ce butoane au fost apăsate la eveniment. Măstile pentru fiecare buton şi tastele modificatoare sunt:

Masca	Semnificaţie
`InputEvent.BUTTON1_MASK`	Mouse buton1
`InputEvent.BUTTON2_MASK`	Mouse buton2
`InputEvent.BUTTON3_MASK`	Mouse buton3
`InputEvent.ALT_MASK`	Tasta alt
`InputEvent.CTRL_MASK`	Tasta control
`InputEvent.SHIFT_MASK`	Tasta shift
`InputEvent.META_MASK`	Tasta meta
`InputEvent.ALT_GRAPH_MASK`	Tasta alt-graph

Pentru a rescrie exemplul anterior folosind măşti pe biţi pentru a verifica dacă s-a apăsat butonul dreapta în timp ce era apăsată, am putea scrie următoarele:

```
int RIGHT_SHIFT_MASK = InputEvent.BUTTON3_MASK + InputEvent.SHIFT_MASK;
. . .
if ((e.getModifiers() & RIGHT_SHIFT_MASK) == RIGHT_SHIFT_MASK) {
   ...
```

1.4. Exemplu - MouseTest.java – Exemplul arată coordonatele indicatorului mouse.

Aceasta este o demonstraţie simplă de ascultare a evenimentelor mouse pe un panou. Aplicaţia afişează doua panouri pentru a ilustra cum depinde ascultătorul de mouse de panou.

Programul principal

```
// File:   mousedemo/MouseTest.java
// Description: Program principal/applet pentru demonstrarea ascultatorilor mouse
// Author: Fred Swartz
// Date:   2005-02-03, 2000-11-29...2002-11-21
// Imbunatatiri posibile: de aratat alte evenimente mouse.


package mousedemo;
import javax.swing.*;

////////////////////////////////////////////////////// clasa MouseTest
public class MouseTest extends JApplet{
    //=================================================== constructor
    public MouseTest() {
        add(new DualMousePanel());
    }

    //=================================================== method main
    public static void main(String[] args) {
        JFrame window = new JFrame("Mouse Demo");
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setContentPane(new DualMousePanel());
        window.pack();
        window.setVisible(true);
    }
}
```

Panoul de conţinut al GUI

```
// File:   mousedemo/MousePanel.java
// Description: Panou care contine doua MousePanels.
// Author: Fred Swartz
// Date:   2005-02-03, 2000-11-29...2002-11-21

package mousedemo;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;

////////////////////////////////////////////////// class DualMousePanel
class DualMousePanel extends JPanel {
    //=================================================== constructor
    public DualMousePanel() {
        //--- Creaza doua MousePanels
        MousePanel mp1 = new MousePanel();
        MousePanel mp2 = new MousePanel();

        //--- Adauga margini (nota: marginile sunt inauntrul panoului)
        Border etched = BorderFactory.createEtchedBorder();
        mp1.setBorder(BorderFactory.createTitledBorder(etched, "Panel 1"));
        mp2.setBorder(BorderFactory.createTitledBorder(etched, "Panel 2"));

        //--- Aranjeaza panourile
        this.setLayout(new GridLayout(1, 2));
        this.add(mp1);
        this.add(mp2);
    }//end constructor
}
```

Panoul care ascultă şi arată evenimentele mouse

```
// File:   mousedemo/MousePanel.java
// Description: Panoul crea asculta evenimentele mouse si afiseaza informatie
//              despre unele dintre ele.
// Author: Fred Swartz
// Date:   2005-02-03, 2000-11-29...2002-11-21

package mousedemo;

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

////////////////////////////////////////////////////// clasa MousePanel
class MousePanel extends JPanel implements MouseListener, MouseMotionListener {

   //================================================ variabile instanta
    private int m_lastClickedX = 0;  // coordonata x a clic-ului
    private int m_lastClickedY = 0;  // coordonata y a clic-ului
    private int m_lastMovedX   = 0;  // coordonata x a deplasarii mouse
    private int m_lastMovedY   = 0;  // coordonata y a deplasarii mouse


    //=================================================== constructor
    public MousePanel() {
        this.setBackground(Color.white);
        this.setPreferredSize(new Dimension(200, 200));
        //--- Adauga ascultatorii mouse.
        this.addMouseListener(this);       // asculta evenimente mouse
        this.addMouseMotionListener(this); // asculta deplasari si tiriri
    }


    //============================================== metoda paintComponent
    public void paintComponent(Graphics g) {
        super.paintComponent(g);  // deseneaza fundalul si marginile
        g.drawString("Last click: x=" + m_lastClickedX
                     + ", y=" + m_lastClickedY , 10, 30);
        g.drawString("x=" + m_lastMovedX   + ", y=" + m_lastMovedY
                     , m_lastMovedX, m_lastMovedY);
    }


    //============================================== ascultator mouseClicked
    public void mouseClicked(MouseEvent e) {
        m_lastClickedX = e.getX(); // Salveaza coordonata x a clic-ului
        m_lastClickedY = e.getY(); // Salveaza coordonata y a clic-ului
        this.repaint();            // Deseneaza panoul cu noile valori.
    }


    //============================================== ascultator mouseMoved
    public void mouseMoved(MouseEvent e) {
        m_lastMovedX = e.getX();
        m_lastMovedY = e.getY();
        this.repaint();
    }


    //=========================================================== ignorate
    //==== celelalte evenimente mouse trebuie sa fie aici, dar nu fac nimic.
    public void mouseDragged (MouseEvent e) {}  // ignora
    //==== aceste evenimente mouse "lente" sunt ignorate.
    public void mouseEntered (MouseEvent e) {}  // ignora
```

```
    public void mouseExited  (MouseEvent e) {}  // ignora
    public void mousePressed (MouseEvent e) {}  // ignora
    public void mouseReleased(MouseEvent e) {}  // ignora
}
```

1.5. Exemplu - DragDemo.java – Exemplul ilustreaz a târârea cu mouse.

Târâţi mingea pe applet-ul din stânga. Sursa acestui program este dată mai jos. Programul este scris atât ca applet (subclasa `JApplet`) cât şi ca aplicatie (are o metodă `main`). Cele două fişiere sursă sunt date mai jos.
Imaginea este desenată cu metoda din Graphics (`fillOval`), dar se poate afişa cu uşurinţă o imagine.

Cadrul principal main/applet

```
/** DragDemo.java – Exemplu de tirire cu mouse; duala - aplicatie/applet
    @author Fred Swartz
    @version 2004-04-15
*/
// "appletviewer DragDemo.java" functioneaza cu linia urmatoare.
// <applet code="DragDemo.class" height="200" width="200"></applet>
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
//////////////////////////////////////////////////////////// class DragDemo
/** Aceasta este o aplicatie pentru ca are o metoda main.
    Este si applet pentru ca extinde JApplet.
*/
public class DragDemo extends JApplet {
    //============================================================ metoda main
    public static void main(String[] args) {
        JFrame window = new JFrame();
        window.setTitle("Drag Demo");
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setContentPane(new DragBallPanel());
        window.pack();
        window.show();
    }//end main

    //================================================== applet constructor
    public DragDemo() {
        this.setContentPane(new DragBallPanel());
    }
}//endclass DragDemo
```

Panoul folosit pentru grafica

```
/** DragBallPanel.java - Panel care permite tirirea mingii.
    @author Fred Swartz
    @version 2004-04-15
*/

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

//////////////////////////////////////////////////////// class DragBallPanel
/** La invocarea ascultatorului mousePressed lse testeaza pozitia
    pentru a vedea daca este in zona mingii.  Daca este, atunci
    (1) seteaza _canDrag true insemnind fii atent la evenimentele MouseDragged.
    (2) inregistreaza unde in minge (relativ la coordonatele stinga sus)
        s-a dat clic, deoarece arata cel mai bine daca tragem de acolo.
```

```
*/
public class DragBallPanel extends JPanel implements MouseListener,
MouseMotionListener {

    private static final int BALL_DIAMETER = 40; // Diametrul mingii
    //--- variabile instanta
    /** Coord. mingii.  Modificate de ascultatorii mouse.
        Folosite de paintComponent. */
    private int _ballX     = 50;   // coord x - setata din drag
    private int _ballY     = 50;   // coord y - setata din drag

    /** Pozitia in minge a apasarii mouse pentru a face tirirea
        sa arate mai bine. */
    private int _dragFromX = 0;    // apasat atit de departe in cutia
    private int _dragFromY = 0;    // care margineste mingea.

    /** true inseamna ca s-a apasat mouse in minge si este inca in panou.*/
    private boolean _canDrag  = false;

    //============================================================ constructor
    /** Constructorul seteza marimea, culorile si adauga ascultatori de mouse.*/
    public DragBallPanel() {
        setPreferredSize(new Dimension(300, 300));
        setBackground(Color.blue);
        setForeground(Color.yellow);
        //--- Adauaga ascultatori de mouse.
        this.addMouseListener(this);
        this.addMouseMotionListener(this);
    }//endconstructor

    //=================================================== method paintComponent
    /** Mingea este desenata la ultimele coordonate inregistrate de ascultatorul
        mouse. */
    public void paintComponent(Graphics g) {
        super.paintComponent(g);   // Necesar pentru fundal.
        g.fillOval(_ballX, _ballY, BALL_DIAMETER, BALL_DIAMETER);
    }//end paintComponent

    //=================================================== method mousePressed
    /** Seteaza _canDrag daca clicul este in minge (sau in cutia care o
        margineste, ceea ce arata lene, dar e destul de aproape pentru acest
        program).
        Retine deplasamentul (dragFromX and Y) in minge
        Pentru a-l folosi ca punct relativ de afisat in timpul tiririi.
    */
    public void mousePressed(MouseEvent e) {
        int x = e.getX();   // Salveaza coordonata x  a clicului
        int y = e.getY();   // Salveaza coordonata y  a clicului

        if (x >= _ballX && x <= (_ballX + BALL_DIAMETER)
                && y >= _ballY && y <= (_ballY + BALL_DIAMETER)) {
            _canDrag = true;
            _dragFromX = x - _ballX;  // cit de departe de stinga
            _dragFromY = y - _ballY;  // cit de departe de partea de sus
        } else {
            _canDrag = false;
        }
    }//end mousePressed

    //============================================================ mouseDragged
    /** Seteaza x,y  la pozitia mouse si redeseneaza. */
    public void mouseDragged(MouseEvent e) {
        if (_canDrag) {   // True doar daca s-a apasat un buton in minge.
            //--- Pozitia mingii de la mouse si deplasamentul clicului original
            _ballX = e.getX() - _dragFromX;
            _ballY = e.getY() - _dragFromY;
```

```
            //--- Nu muta mingea in afara ecranului marginilor stinga-dreapta
            _ballX = Math.max(_ballX, 0);
            _ballX = Math.min(_ballX, getWidth() - BALL_DIAMETER);

            //--- Nu muta mingea in afara ecranului marginilor sus-jos
            _ballY = Math.max(_ballY, 0);
            _ballY = Math.min(_ballY, getHeight() - BALL_DIAMETER);

            this.repaint(); // Redeseneaza pentru ca s-a schimbat pozitia.
        }
    }//end mouseDragged

    //================================================== method mouseExited
    /** Opreste tirirea daca mouse iese din panou. */
    public void mouseExited(MouseEvent e) {
        _canDrag = false;
    }//end mouseExited

    //============================================ Ignora alte evenimente mouse.
    public void mouseMoved   (MouseEvent e) {}  // ignora aceste evenimente
    public void mouseEntered (MouseEvent e) {}  // ignora aceste evenimente
    public void mouseClicked (MouseEvent e) {}  // ignora aceste evenimente
    public void mouseReleased(MouseEvent e) {}  // ignora aceste evenimente
}//endclass DragBallPanel
```