Lucrarea nr. 11 - Evenimente, mouse şi tastatură

Breviar teoretic

Clasa Event (Eveniment) defineşte constante pentru toate evenimentele pentru care o metodă de gestionare a evenimentelor poate răspunde. Principalele câmpuri din clasă:

Câmp	Descriere
Obiect arg	Informație specifică evenimentului. Pentru un buton acest câmp este eticheta butonului
int clickCount	Această variabilă contorizează numărul de evenimenete datorate mouse-ului. Valoarea 1 corespunde unui singur click al mouse-ului, valoarea 2 pentru 2 click-uri, etc.
int id	Identifică tipul de eveniment : MOUSE_DOWN, MOUSE_MOVE, KEY_PRESS, etc.
int key	Această variabilă este legată de evenimentele de tip tastă apăsată. Pentru un eveniment de tip KEY_PRESS această tastă va fi valoarea cheiei ce a fost apăsată.
int modifiers	Aceşti identificatori permit utilizarea cheilor SHIFT şi CTRL. În clasa Event sunt definite constante de tipul SHIFT_MASK şi CTRL_MASK.
Object target	Identifică tipul de obiect ce generează evenimentul buton, câmp de tip text, etc.
long when	Identificarea momentului când a avut loc evenimentul.
int x	Coordonata X asociată unui eveniment, utilizată de obicei în conjuncție cu poziția mouse-ului, în momentul declanşării evenimentului.
int y	Coordonata Y asociată unui eveniment, utilizată de obicei în conjuncție cu poziția mouse-ului, în momentul declanşării evenimentului.

Mouse-ul

Mouse-ul este unul dintre cele mai importante elemente ce poate permite interfațarea directă a utilizatorului cu mediul de programare. Evenimentele ce pot fi asociate mouse-ului sunt:

MOUSE_DOWN	Acest eveniment, declanşat de metoda mouseDown(), este datorat faptului că utiliyatorul a apăsat pe un buton al mouse-ului.
MOUSE_UP	Acest eveniment, declaşat de metoda mouseUp(), este datorat faptului că utiliyatorul a aliberat butonul stâng al mouse-ului.
MOUSE_MOVE	Evenimentul declaşat de metoda mouseMove(), datorată faptului că utiliyatorul a mişcat mouse-ul pe ecran.
MOUSE_DRAG	Eveniment generat de metoda mouseDrag(), constând în mişcarea mouse-ului simultan cu menținerea apăsată a butonului stâng al mouse-ului.
MOUSE_ENTER	Eveniment datorat metodei mouseEnter(), generat de mişcarea mouse-ului în zona activă a applet-ului sau componentei.
MOUSE_EXIT	Metoda mouseExit() generează acest eveniment, în clipa în care mouse-ul părăseşte yona controlată de utilizator sau de applet.

Lucrul cu butoanele mouse-ului

Cea mai des utilizată metodă este MOUSE_DOWN, generată de un utilizator care acționează anumite zone dintr-un applet. Această metodă este cea care permite utilizatorului să declanşeze un eveniment la

activarea unui buton. Spere deosebire de utilizarea separat ă a metodei, acest caz este gestionat separat de java, ca element încorporat în obiectul buton. Cu siguranță că sunt numeroase situații în care se poate răspunde evenimentului MOUSE_DOWN pentru a controla evoluția unei aplicații conform dorinței utilizatorului.

Sunt cunoscute mai multe soluții de tratare a evenimentelor. Cea mai generală este de a capta evenimentul produs de metodă MOUSE_DOWN pentru ca apoi să se rescrie metoda conform specificațiilor utilizatorului. Java apelează automat metoda **mouseDown()** de fiecare dată când este generat un eveniment de tip MOUSE_DOWN. Semnătura unei metode **mouseDown()** are următoarea formă:

```
public boolean mouseDown(Event evt, int x, int y)
```

Argumentele oferite de metodă sunt un obiect de tip eveniment - Event object şi coordonatele X,Y ale evenuimentului generat de mouse. În continuare este prezentat un exemplu care explorează acest tip de eveniment, folosind parametrii generați de metodă pentru a afişa un mesaj.

Aplicatie privind lucrul cu mouse-ul si evenimente generate de acesta - afisarea coordonatelor curente ale mouse-ului:

```
import java.awt.*;
//sunt importate pachetul grafic şi pachetul de lucru cu //applet-ul
import java.applet.*;


public class MouseApplet extends Applet

{
int coordX, coordY;
//sunt  declarate  variabilele  de  clasă

public void init()
//este  creată  metoda  de  iniţializare

{
 coordX = -1; coordY = -1;
 Font font = new Font("TimesRoman", Font.BOLD, 24);
 setFont(font);
 resize(400,  300);
}

public  void  paint(Graphics  g)
//este  creată  metoda  de  desenare
{
if(coordX != -1) g.drawString("Click!", coordX, coordY);
}

public boolean mouseDown(Event evt, int x, int y)
//este construită metoda de tratare a evenimentului
 {

coordX = x; coordY = y;
repaint();
return true;
 }
}
```

Tratarea mişcărilor mouse-ului

Chiar dacă acțiunile mouse-ului sunt extrem de utile , mişcarea acestuia poate fi folosită pentru a trasa spre exemplu a unor desene. Evenimentele ce apar la mişcarea mouse-ului sunt MOUSE_MOVE. Acestea sunt gestionate de metoda **mouseMove()** ce are următoarele argumente:

```
public boolean mouseMove(Event evt, int x, int  y)
```

Cu excepția numelui, această metodă este identică cu metoda anterioară **mouseDown()** recepționând un eveniment de tip eveniment Event şi generând coordinatele X,Y la apariția evenimentului. Un exemplu al utilizării metodei **mouseMove** este prezentat în continuare.

Aplicatie privind lucrul cu mouse-ul si evenimente generate de acesta - desenarea unui segment de maxim 1000 de puncte intre 2 pozitii consecutive ale mouse-ului:

```
import java.awt.*;
import java.applet.*;

public class Mouse2 extends Applet {
//sunt declarate variabilele de clasă

 Point punctStart;
 Point puncte[];
 int numPuncte;
 boolean desenare;

public  void  init()
 {//este rescrisă metoda de inițializare
punctStart = new Point(0, 0);
puncte = new Point[1000];

//este construit un vector de 1000 de componente numPuncte = 0;

drawing  =  false;
resize(400,  300);
  }

public void paint(Graphics g) {
//este rescrisă metoda de desenare

int oldX = punctStart.x;
int oldY = punctStart.y;

for(int x=0; x<numPuncte; ++x)
{
 //se construişte segmentul ce uneşte ultima locație a mouse-ului cu locația
curentă a acestuia
 g.drawLine(oldX, oldY, puncte[x].x, puncte[x].y);
 oldX = puncte[x].x;
 oldY = puncte[x].y;
 //se  reține  ultima  locație  a  mouse-ului
 }
}

public boolean mouseDown(Event evt, int x, int y)
{
```

```
//dacă butonul stânga este apăsat se validează desenarea şi se declară ca
punct curent de desenare punctul curent al mouse-lui

desenare = true;
punctStart.x = x;
punctStart.y = y;
return true;
}


public boolean mouseMove(Event evt, int x, int y)
{//dacă validarea s-a realizat această metodă validează desenarea a maxim
1000 de puncte

if ((desenare) && (numPuncte < 1000))
  {
puncte[numPuncte] = new Point(x, y);
++numPuncte;
repaint();
  }
return  true;
 }
}
```

Probleme propuse

1. Folosiți metoda **mouseUp** pentru a construi un applet pe care atunci când veți elibera butonul stâng al mouse-ului i se va schimba imaginea din background.
2. Scrieți un applet care desenează un dreptunghi ori de câte ori utilizatorul accesează mouse-ul prin butonul stâng.
3. Scrieți un applet care afişează coordonatele curente ale mişcării mouse-ului.
4. Modificați appletul Mouse astfel încât primul eveniment de tip MOUSE_DOWN să selecteze punctul de start, după care appletul să afişeze fiecare coordinate datorate evenimentului de tip MOUSE_MOVE . Appletul va trasa linia numai la activarea tastei f2. Activarea tastei f3 va şterge ultima linie din cadrul appletului permițând trasarea unei noi linii de la capătul ultimei linii rămase pe ecran.

Tastatura

Alături de mouse utilizarea tastaturii reprezintă mijlocul cel mai des utilizat pentru realizarea comunicării calculator- utilizator. Chiar dacă unele evenimente care implică utilizarea tastaturii sunt integrate în anumite obiecte Java) un bun exemplu în acest sens sunt obiectele **Text** şi **TextField** , integrarea accesului la taste, în cadrul unui program se realizează prin utilizarea a două evenimente **KEY_PRESS** şi **KEY_RELEASE** .

Evenimentele de tip tastatură – KEY_PRESS

Atunci când o tastă este apăsată, Java trimite către aplicație un eveniment de tip **KEY_PRESS**. Răspunsul la acest eveniment este permis prin metoda **keyDown()**, a cărei structură inițială este:

```
public boolean keyDown(Event evt, int key)
```

Se recunoaşte structura specifică metodelor ce răspund la evenimente, alături de Event, fiind şi un întreg generat de tasta ce a fost apăsată. Întregul reprezintă codul ASCII al caracterului asociat tastei apăsate. Cum în majoritatea a cazurilor, în cadrul unui program este utilizată o variabilă de tip caracter, conversia explicită a acestei variabile permite lucrul cu caractere:

```
char c = (char)key;
```

Taste predefinite

Unele taste au nume predefinit,, motiv pentru care şi Java le-a alocat o serie de constante, pentru identificarea lor. Aceste taste sunt : tastele speciale F1 la F12, Shift, Ctrl, Page Up, Page Down. Constantele asociate lor sunt :

Constanta	**Tasta**
DOWN	Săgeata în jos
END	Tasta End
F1	Tasta F1
F2	Tasta F2
F3	Tasta F3
F4	Tasta F4
F5	Tasta F5
F6	Tasta F6
F7	Tasta F7
F8	Tasta F8
F9	Tasta F9
F10	Tasta F10
F11	Tasta F11
F12	Tasta F12
HOME	Tasta Home
LEFT	Săgeata stânga
PGDN	Tasta Page Down
PGUP	Tasta Page Up
RIGHT	Săgeata dreapta
UP	Săgeata în sus

Taste combinate

În cadrul clasei **Event** se pot defini şi o serie de taste combinate prin utilizarea tastei Shift, a tastei Alt sau a tastei Ctrl. Modificatorii asoiați acestor taste sunt **ALT_MASK**, **SHIFT_MASK**, şi **CTRL_MASK**. Pentru tastele Shift şi Ctrl sunt utilizate metodele **shiftDown()** şi **controlDown(),** fiecare metodă furnizând o variabilă logică care indică dacă respectiva tastă a fost sau nu apăsată. Pentru tasta Alt, nefiind asociată nici o metodă, se poate apela la un artificiu: se inspectează câmpul al clasei, pentru a determina dacă o tastă a fost sau nu apăsată. Apoi printr-o funcție de tip **ŞI** se realizează intersecția cu evenimentul tastei Alt. Implementarea acestui artificiu va conduce la un indicator logic de forma:

```
boolean altPressed=(evt.modifiers & Event.ALT_MASK)!=0;
```

Aplicatie privind lucrul cu tastatura si evenimente generate de aceasta:

```
import java.awt.*;
import java.applet.*;

public  class  KeyApplet  extends  Applet
{
int  keyPressed;

public  void  init()
{
//initializarea appletului prin stabilirea tipului de font

 keyPressed = -1;
 Font font =new Font("TimesRoman", Font.BOLD, 144);
 setFont(font);
 resize(200, 200);
 }

public  void  paint(Graphics  g)
 {
//metoda de desenare va şterge caracterul anterior "", după care va prelua
caracterul introdus de la tastatură
 String str = "";
 if (keyPressed != -1)
    {

str += (char)keyPressed;
g.drawString(str, 40, 150);
    }
 }

public  boolean  keyDown(Event  evt,  int  key)
{
//metda de tratare a evenimentului va furniza codul int al tastei apăsate,
după care va accesa metoda paint. Se observă ca de fapt are loc rescrierea
metodei

KeyDown keyPressed = key;

repaint();
return true;
//se confirmă programului Java că s-a răspuns evenimentului şi implicit
metodei generată de acesta
}
}
```

Lucrul direct cu evenimentele generate de mouse si tastatura

Toate evenimentele ce au loc în cadrul unui applet sunt procesate de metoda handleEvent(), moştenită din clasa Component. Atunci când metoda nu este rescrisă, implementarea implicită gestionează toate metodele ce răspund unui eveniment.În cele ce urmează este listată forma implicită a acestei metode `handleEvent( ).`

```
public boolean handleEvent(Event evt)
```

```
{ switch (evt.id) {

case Event.MOUSE_ENTER:return  mouseEnter(evt, evt.x, evt.y);

case Event.MOUSE_EXIT:return mouseExit(evt, evt.x, evt.y);

case Event.MOUSE_MOVE:return mouseMove(evt, evt.x, evt.y);

case Event.MOUSE_DOWN:return mouseDown(evt, evt.x, evt.y);

case  Event.MOUSE_DRAG:return mouseDrag(evt, evt.x, evt.y);

case  Event.MOUSE_UP:return mouseUp(evt, evt.x, evt.y);

case Event.KEY_PRESS:
case Event.KEY_ACTION:return keyDown(evt, evt.key);

case Event.KEY_RELEASE:

case Event.KEY_ACTION_RELEASE: return keyUp(evt, evt.key);

case Event.ACTION_EVENT: return action(evt, evt.arg);

case Event.GOT_FOCUS:return  gotFocus(evt,  evt.arg);

case  Event.LOST_FOCUS:return  lostFocus(evt,  evt.arg);

}

return false;

}
```

Pentru o configurare a acestei metode conform dorinței programatorului aceasta poate fi rescrisă. Prin rescriere se vor modifica numai acele componente ce prezintă interes în structura programului. Această modalitate de tratare a evenimentelor este mai compactă şi are o structură mult mai apropiată de programarea orientată pe obiecte.

La rescrierea metodei **handleEvent(),** se pot ignora evenimentele ce nu prezintă interes, fără a uita ca atunci când se doreşte ignorarea unui mesaj furnizat de evenimentul Java să se introducă răspunsul **false** la evenimentul rescris.

În exemplul următor se reia aplicația prezentată în cadrul tratării mişcării mouse-ului.

Aplicație

```
import java.awt.*;
import java.applet.*;

public  class  MouseApplet3  extends  Applet

{
Point startPoint;
Point points[];
int numPoints;
boolean drawing;

public  void  init()

{
startPoint =newPoint(0,0);
points = new Point[1000];
numPoints = 0;
```

```
drawing =false;
resize(400,300);
}

public  void  paint(Graphics  g)
{

int oldX = startPoint.x;
int oldY = startPoint.y;
for  (int  x=0;  x<numPoints;  ++x)
{
g.drawLine(oldX, oldY, points[x].x, points[x].y);
oldX = points[x].x;
oldY  =  points[x].y;

}
}

public  boolean  handleEvent(Event  evt)

{
switch(evt.id)
{
case Event.MOUSE_DOWN: drawing = true;
startPoint.x = evt.x;
startPoint.y = evt.y;
return true;

case  Event.MOUSE_MOVE:

if ((drawing) && (numPoints < 1000))
 {

points[numPoints] = new Point(evt.x, evt.y); ++numPoints;
repaint();
   }

return true;
default:return false;
  }
 }
}
```

Probleme propuse

1. Scrieți un applet care desenează un dreptunghi a cărui culoare se schimbă de ori de câte ori utilizatorul apasă o tastă.
2. Scrieți un applet care permite utilizatorului să introducă un şir de caractere de la tastatură şi să le afişeze pe ecran. Utilizați un obiect de tip String pentru a memora caracterele, adăugând la obiect fiecare nou caracter introdus de la tastatură. Folosiți apoi o metodă de tip paint() pentru a vizualiza obiectul.