Curs 12 POO
© Conf. univ dr. Crenguta M. Puchianu

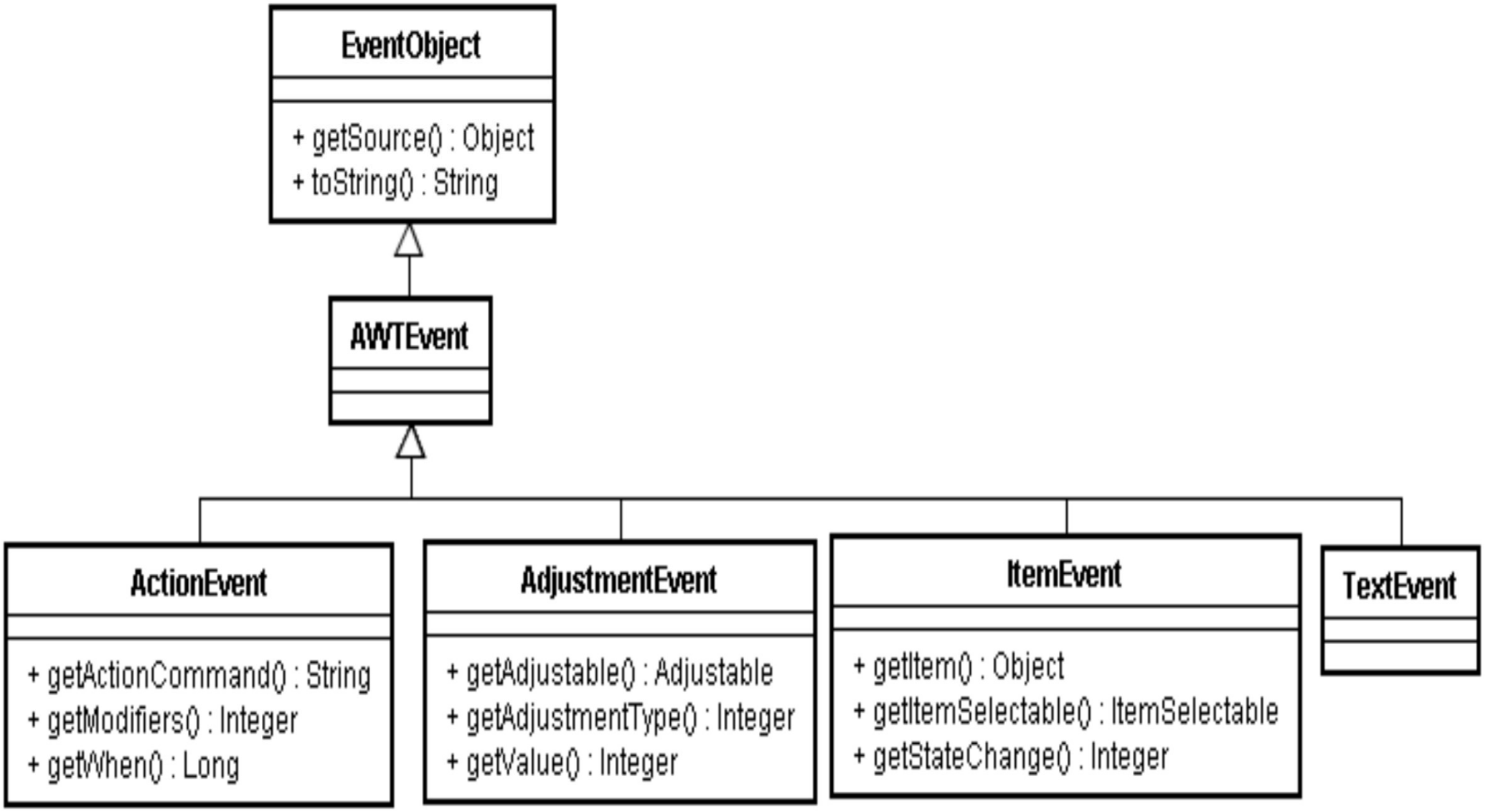
- Evenimente
- Evenimente semantice
- Evenimente de nivel coborat
- Clase anonime
- Functii lambda
- Interfete functionale

Modelul de gestiune a evenimentelor

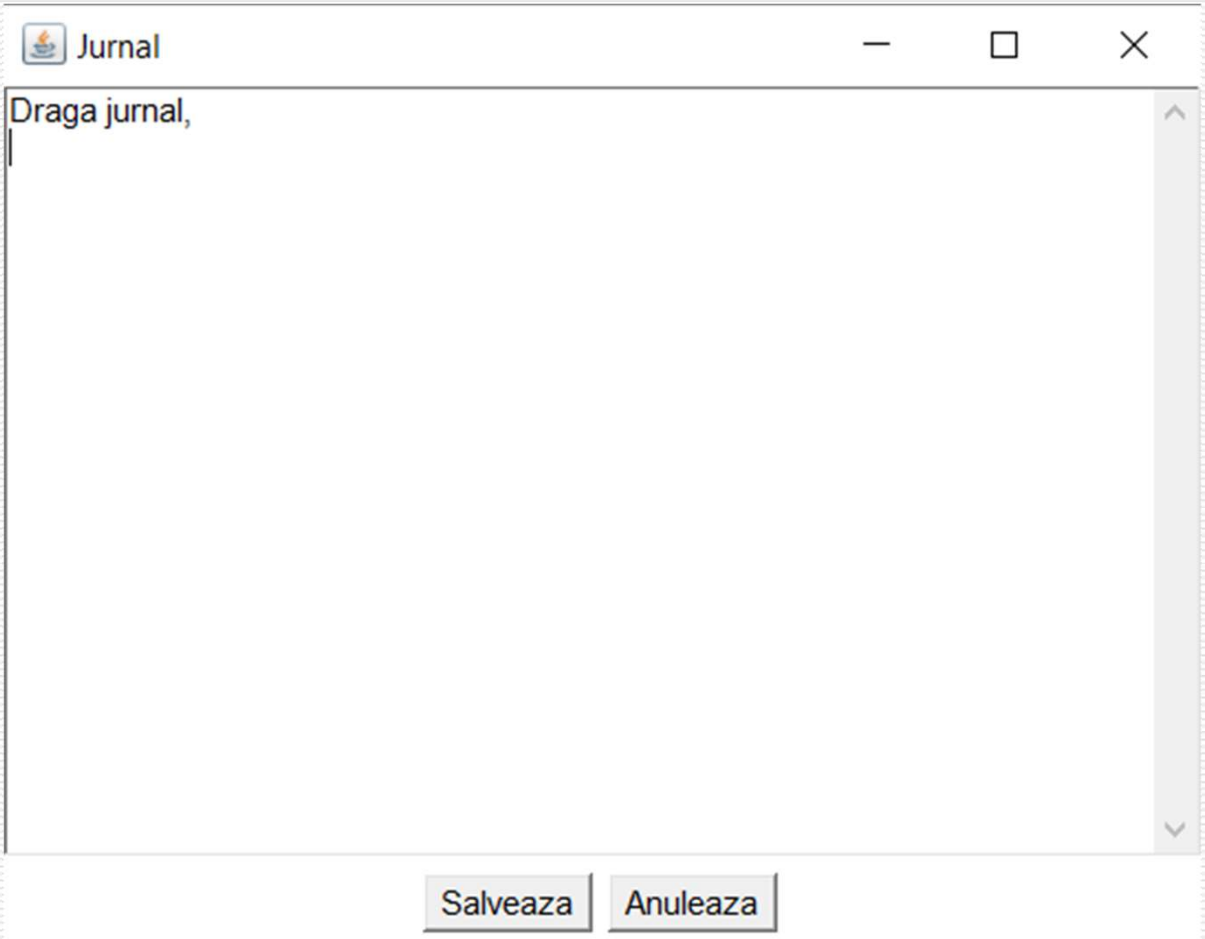
- **Sursă de evenimente**: orice componentă grafică.
- **Eveniment**: obiect lansat de o sursă de evenimente. Evenimentul este de un tip care specializează clasa AWTEvent.
- **Ascultător**: obiect care implementează un gestor de evenimente de un anumit tip. Gestorul respectiv va indica acţiunile ce vor fi executate când apare evenimentul pentru care s-a înregistrat. Clasa trebuie să implementeze o interfaţă de tip EventListener.

Componentă AWT/Swing	Tipuri de evenimente generate									
	acti on	adjus - tment	compo- nent	contain er	focu s	ite m	key	mouse	text	windo w
Component			X		X		X	X		
Container			X	X	X		X	X		
JButton	X		X		X		X	X		
JCheckBox			X		X	X	X	X		
JCheckboxMe nuItem						X				
JComboBox			X		X	X	X	X		
JDialog			X	X	X		X	X		X
JFrame			X	X	X		X	X		X
JLabel			X		X		X	X		
JList	X		X		X	X	X	X		
JMenuItem	X									
JPanel			X	X	X		X	X		
JScrollbar		X	X		X		X	X		
JScrollPane			X	X	X		X	X		
JTextArea			X		X		X	X	X	
JTextField	X		X		X		X	X	X	
JWindow			X	X	X		X	X		X

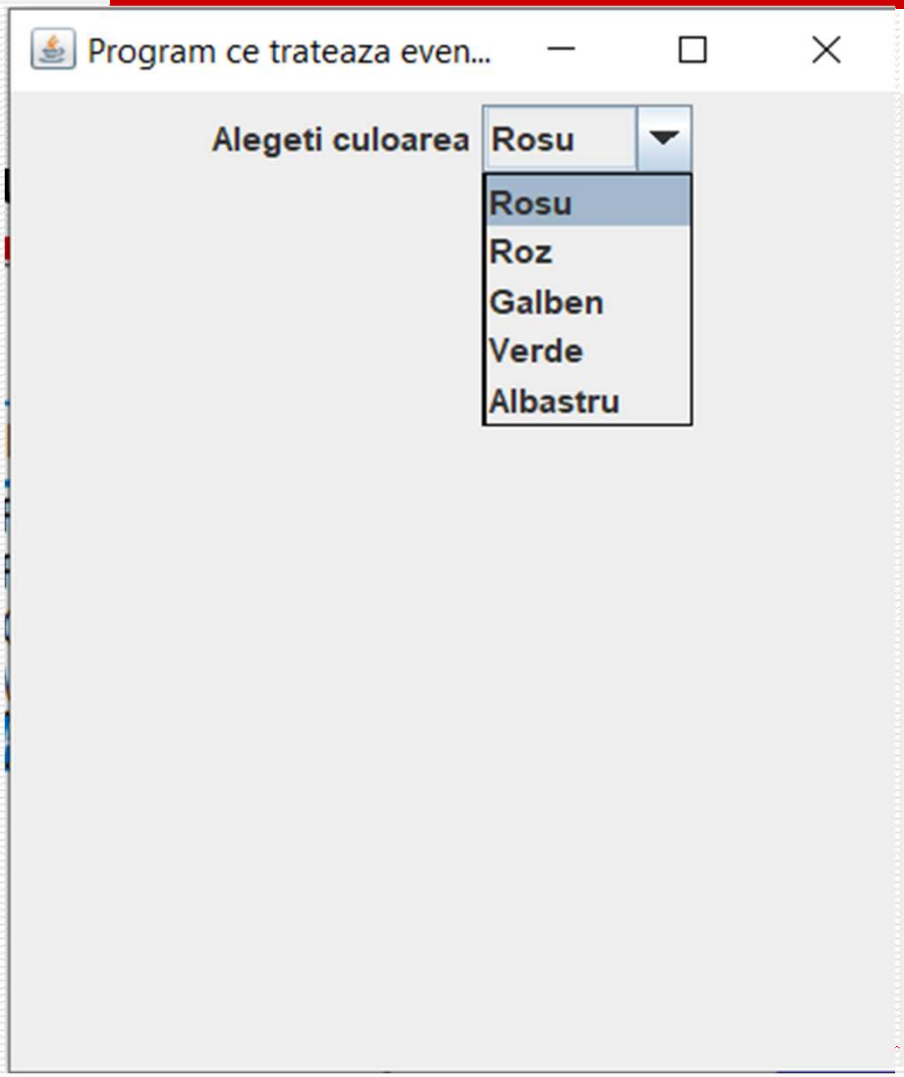
Evenimente semantice

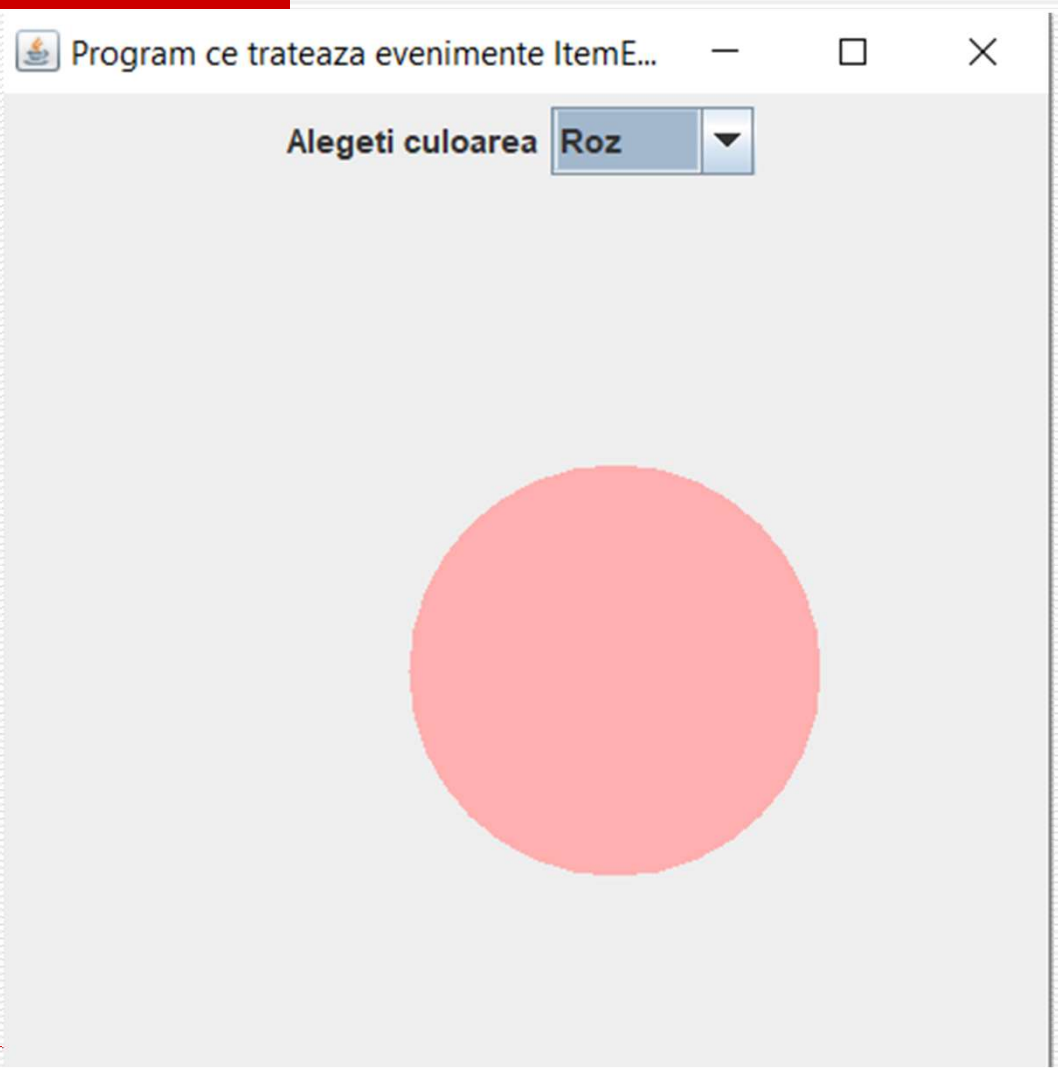


Gestiunea evenimentelor de tip ActionEvent

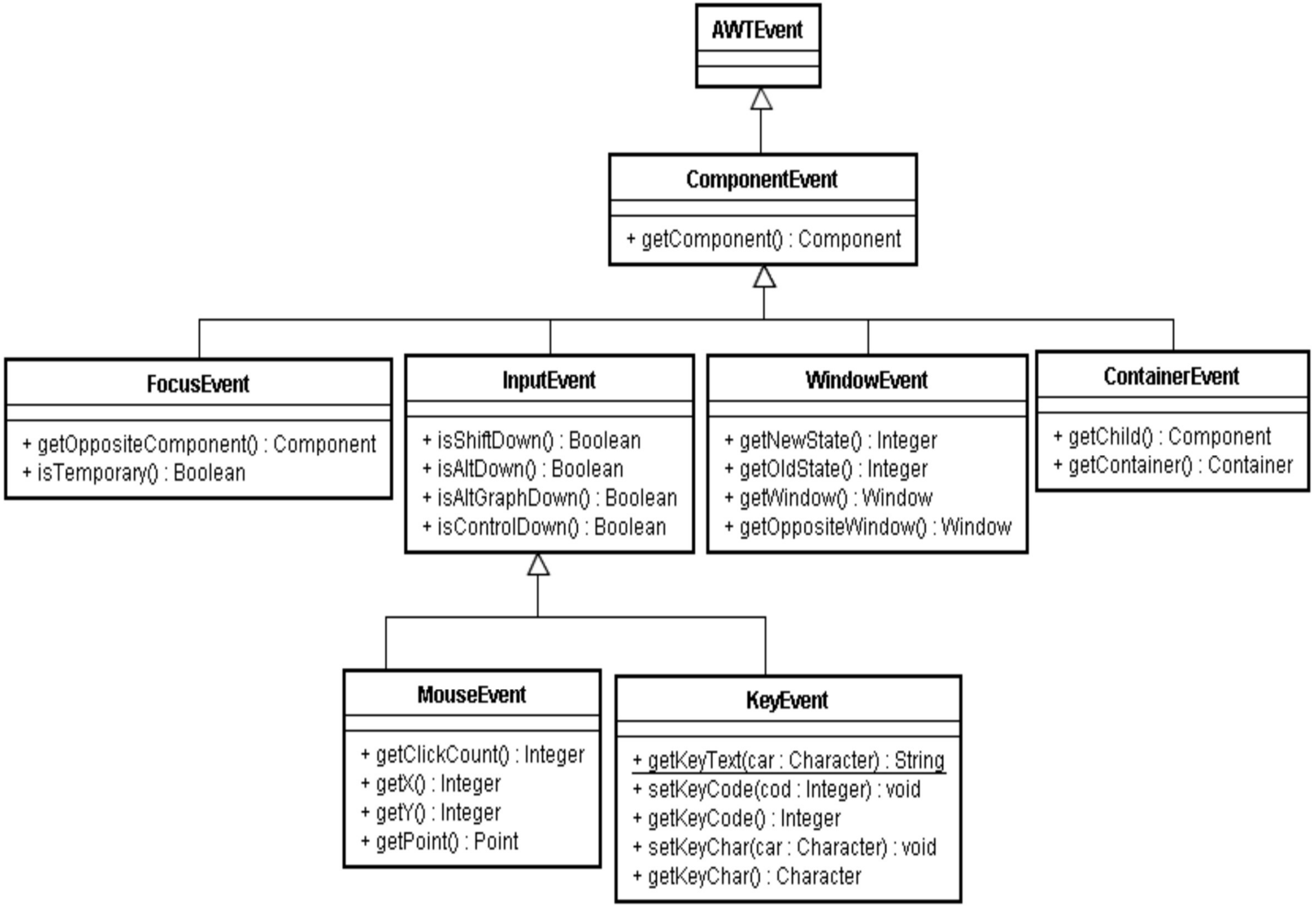


Gestiunea evenimentelor de tip ItemEvent





Evenimente de nivel coborat



Gestiunea evenimentelor de tip KeyEvent

Un program care vizualizează o fereastră vidă pe ecran şi permite utilizatorului să apese tastele săgeţi pentru a muta fereastra pe ecranul monitorului.

În plus, utilizatorul poate folosi tastele C, S şi D pentru a plasa fereastra în centrul, stânga sau dreapta ecranului.

Clase adapter ale interfeţelor de ascultare

Interfaţă Listener	Clasă adapter	Identificatori metode	Evenimente generate de:
ActionListener	nu există	actionPerformed	JButton, JList, JMenuItem, JTextField
AdjustmentListener	nu există	adjustmentValueChanged	JScrollBar
ComponentListener	ComponentAdapter	componentHidden componentMoved componenResized componentShown	Component
ContainerListener	ContainerAdapter	componentAdded componentRemoved	Container
FocusListener	FocusAdapter	focusGained focusLost	Component
ItemListener	nu există	itemStateChanged	JCheckBox, JComboBox, JCheckboxMenuItem, JList
KeyListener	KeyAdapter	keyPressed keyReleased keyTyped	Component
MouseListener	MouseAdapter	mouseClicked mouseEntered mouseExited mousePressed mouseReleased	Component
MouseMotionListener	MouseMotionAdapter	mouseDragged mouseMoved	Component
TextListener	nu există	textValueChanged	JTextComponent

Clase anonime si adaptori

- Orice interfaţă xxxxListener cu mai multe metode are o clasă xxxxAdapter care introduce o implementare vidă pentru metodele interfeţei. Obiectul ascultător poate fi al unei subclase a adaptorului care redefineşte numai metodele necesare.
- În practică, obiectul ascultător face parte dintr-o clasă internă anonimă definită ca subclasă a ascultătorului.

```
import java.awt.*;
import java.awt.event.*;
public class FereastraCareSeInchide extends Frame{
 public FereastraCareSeInchide(){
  addWindowListener(new WindowAdapter(){//clasa interna anonima
   public void windowClosing(WindowEvent e){
    System.exit(0);
   } });
 }
 public static void main(String[] args){
  Frame f = new FereastraCareSeInchide();
 f.setSize(300,300);   f.setVisible(true);
 }}
```

Functii lambda

Sintaxa:

ParametriLambda -> CorpLambda

Exemple:

```
x->x*x
(int x) -> x+2
() -> System.out.println("Buna")
(int x, int y) -> (x+y)/(x-y)
() -> { return 3.1415 };
(x,y) -> x.length()-y.length()
```

Observatii:

- Daca o functie lambda nu are parametri, trebuie sa apara ()
- Daca avem mai multi parametri, acestia trebuie separati prin virgulă
- Tipul unui parametru nu este obligatoriu sa fie trecut. El va fi dedus la compilare
- -> este un operator numit operator lambda
- Corpul lambda poate fi o expresie sau un bloc. Daca este un bloc, el trebuie cuprins intre {}.
- O functie lambda poate contine orice instructiuni, exceptand break şi continue.

Interfete functionale

O interfaţa funcţionala este o interfaţă ce conţine o singura metoda abstractă. De obicei, o interfaţă funcţională este adnotată cu @FunctionalInterface

Exemplu:

```
@FunctionalInterface
interface Interfata {
 void apeleaza();
}
class Expresie{
 public static void main(String []args) {
  Interfata in = () -> System.out.println("Buna ziua");
  in.apeleaza();
 }
}
```

Chiar daca o interfaţă nu extinde clasa Object, poate contine metode ale clasei Object. Dar, daca conţine numai metode ale clasei Object, nu este o interfaţă funcţională.

O interfaţă funcţională poate contine una sau mai multe metode statice sau default.