C11

Componente Swing si Evenimente

- Swing: JLabel, JTextField, JTextArea, JCheckBox, JRadioButton
- Evenimente

Obiective

După parcurgerea acestui curs ar trebuie sa puteţi:

- înţelege principiile de lucru cu elementele de interfata Swing;
- înţelege posibilitatile de pozitionare a componentelor in contianere;
- modalitatea de crea o interfata grafica sigura utilizand elementele Swing: JLabel , JTextField, JTextArea, JCheckBox, JRadioButton ;
- înţelege conceptul de eveniment şi crea sau modifica rutine de tratare a evenimentelor generate din JDeveopler;

Sablonul model-view-controller

Componentele de interfaţă grafică Swing beneficiază de o arhitectură modelată prin trei caracteristici:

- conţinutul: variabilele de stare a obiectului (buton apăsat sau nu, textul introdus în obiect);
- partea vizuală: culoarea, mărimea obiectului;
- comportamentul: reacţia la evenimente.

Componentele Swing sunt implementate respectând un şablon de proiectare numit model-view-controller. Conform acestui model obiectele Swing sunt implementate prin trei categorii de clase distincte:

- model: stochează conţinutul;
- view: afişează conţinutul;
- controller: tratează interacţiunea cu utilizatorul.

Şablonul defineşte precis modul în care aceste trei clase interacţionează. Modelul stochează conţinu, o stare în vaiabilele de instantă şi nu are cod de interfaţă cu utilizatorul. El trebuie să implementeze metode pentru modificarea conţinutului şi verificarea acestuia. Treaba părţii de vizualizare este aceea de afişare pe ecran a stării stocate în datele modelului. Într-o situaţie generală este posibil ca pe bază modelului să se construiască, simultan, mai multe view-uri, fiecare afişând doar un anumit aspect al conţinutului stocat. Controller-ul tratează evenimentele de interacţiune generate de utilizator cum sunt clic de mouse sau apăsare de tastă. Aici se decide dacă aceste evenimente realizează sau nu modificări în model sau în partea viazuală.

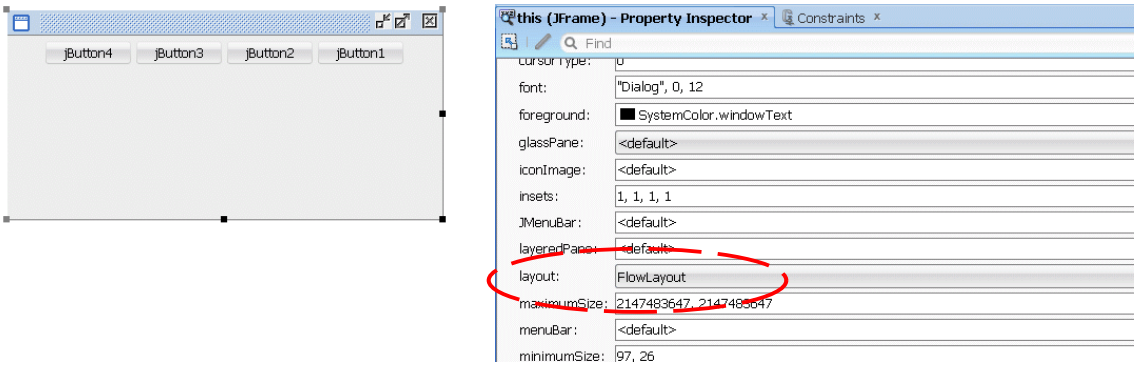
Ca si programator utilizator al controalelor Swing nu este necesară cunoaşterea arhitecturii model-view-controller. Fiecate control are o clasă ce ambalează (de ex. JButton sau JtextField) partea de model şi de view. Dacă dorim să interogăm conţinutul clasa de ambalare întreabă modelul şi ne întoarce răspunsul. Dacă dorim sa modifică aspectul vizual, clasa de ambalare este cea care transmite cerearea către partea vizuală.

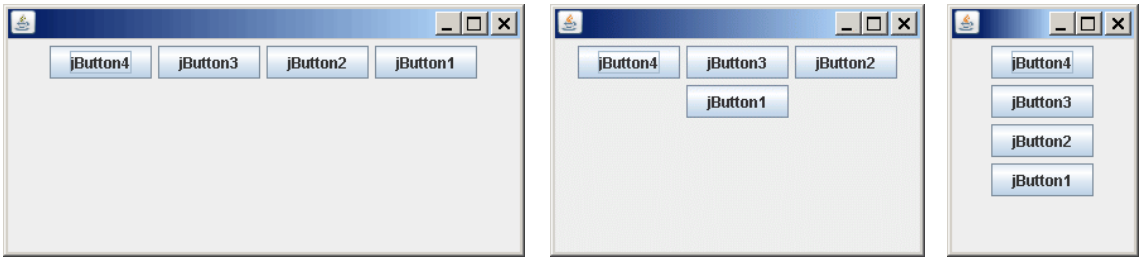
La implementarea Swing-ului partea de model se poate separa foarte clar pentru fiecare componentă de interfaţă, însă responsabilitaţile parţilor de view si controller nu se pot separa foarte clar fiind distribuite în mai multe clase distincte. Clasa de ambalare vine să uniformizeze accesul la aceste regiuni dacă programatorul doreşte să le acceseze cât mai unitar din cod.

Pozitionarea componentelor

Înainte de a prezenta câteva componente Swing utilizate mai des în crearea de interfeţe grafice este momentul să descriem modul de aranjare al acestora în interiorul unui Frame. Spre deosebire de Visual Basic, JDK-ul nu are un generator de interfeţe, programatorul trebuie să poziţioneze prin cod locul în care se afişează componenta. JDeveloper pune la dispoziţia programatorului un astfel de instrument, însă până şi în acest caz exită situaţii ce necesită cod. De exemplu dacă pe suprafaţa unui Frame punem 4 butoane şi redimensionă fereastră principală apar situaţiile de mai jos.

Dacă folosim însă un manager de aşezare al componentelor, definit prin proprietatea (din Visual) layout pe care o setăm la valoarea FlowLayout, atunci lucrurile se modifică. Adică redimensionarea ferestrei ce conţine componetele produce rearanjarea automată a controalelor aşa încât acestea să fie vizibile pe ecran. În cazul unui `JFrame`, implicit, nu avem un astfel de manager asociat Frame-ului iar la redimensionarea ecranului de fereastră zonele afişate nu reacţionează la acest eveniment. Codul scris de JDeveloper la modificarea proprietăţii `layout` este data în continuare.






```
public class Frame1 extends JFrame {
    private JButton jButton1 = new JButton();
    . . .
    private FlowLayout flowLayout1 = new FlowLayout();

    public Frame1() {
        try {
            jbInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        this.getContentPane().setLayout(flowLayout1);
        this.setSize(new Dimension(414, 200));
        jButton1.setText("jButton1");
        . . .
        jButton4.setText("jButton4");
        this.getContentPane().add(jButton4, null);
        . . .
        this.getContentPane().add(jButton1, null);
    }
  . . .
}
```

Observăm că în acest caz aranjarea componentelor se face dinamic. Poziţionarea rezultă ca urmare a managerului de amplasare (layout manager). Există predefinite mai multe astfel de manager-e, **`FlowLayout`** aliniază componentele pe orizontală cât este spaţiu, după care începe un nou rând. O alta metodă de amplasare a componentelor este prin utilizarea de panouri (`JPanel`). Aceasta acţionează ca şi un container al elementelor de interfaţă ce se pot aranja independent de alte regiuni de ecran.

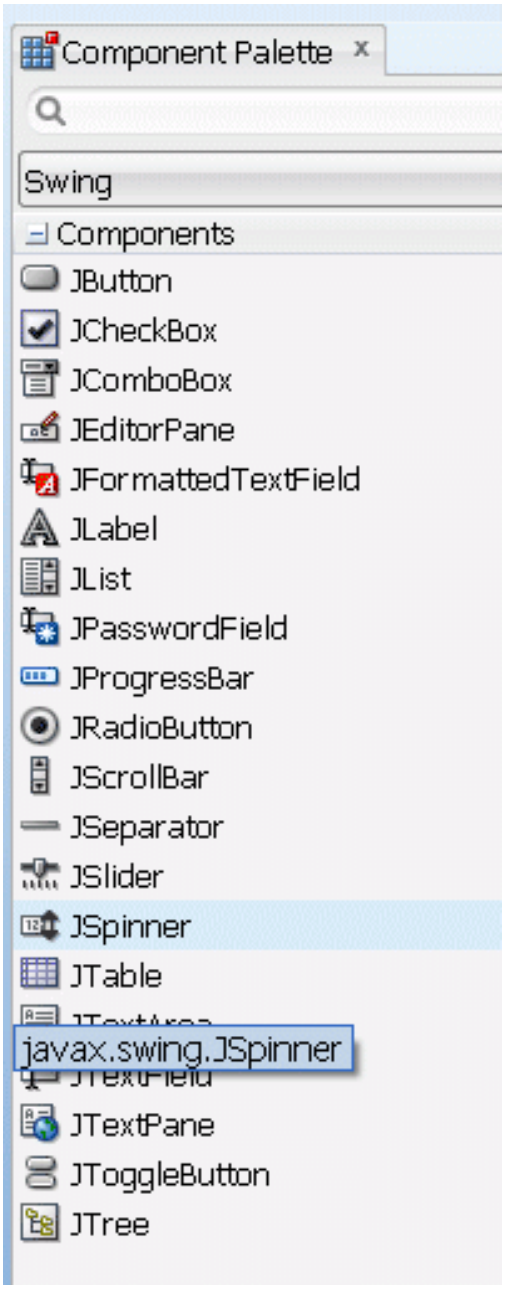
Componente Swing

Componentele utlizate pentru introducerea de date sau editare de text sunt: `JTextField` şi `JTextArea`. Field-ul poate accepta o singură linie de text în timp ce Area-ul poate accepta mai multe linii. O situaţie specială este Filed-ul de parole `JPasswordField`, unde caracterele introduse nu se afişează

Afişarea de date sau etichetarea de componente se face cu `JLabel`.

Există componente specializate şi pentru selectarea de variante dacă numărul acestora este finit. `JCheckBox` permite selectarea de "da" sau "nu" şi vine asociat împreună cu o etichetă. `JRadioButton` dacă este asociat unui grup permite selectarea unei singure opţiuni din grup. `JList` şi `JComboBox` sunt alternative bune dacă numărul de variante fixate este prea mare şi se ocupă o regiune prea mare de ecran cu ele. Variantele se prezinta sub forma unor liste care în funţie de anumite proprietăţi pot fi editate sau actualizate din cod.

Listele asigură selecţia de variante dintr-o mulţime de elemente discrete de text. Dacă însă mulţimea este numerică se poate folosi `JSlider`.



Evenimente

Aplicaţiile ce implementează interfeţe grafice sunt obligate sa lucreze cu evenimente. Pentru a putea crea o interfaţă programatorul trebuie să cunoască modul în care Java prelucrează evenimentele. Monitorizarea interfeţei grafice se face prin controlul sistemului de operare care raportează evenimente, de tipul apăsare de tasta sau clic de mouse, programelor aflate în execuţie. Fiecare program trebuie să decida pe mai departe dacă reacţionează cumva la acele evenimente.

În Visual Basic corespondenţa dintre un eveniment specific şi codul este directă. Programatorul scrie câte o secvenţă de cod, numită rutină de tratare a evenimentului, pentru fiecare eveniment de care este interesat. Atunci când apare evenimemtul, secvenţa respectivă este lansată automat în execuţie.

În limbajul C, programarea orientată pe evenimente implică interacţiune continuă cu sistemul de operare. Mai precis, coada de evenimente, trebuie verificată sistematic pentru a prinde evenimentele, iar programarea respectivului cod este mai dificilă deoarece presupune cunoasterea modului de interacţiune cu sistemul de operare. Avantajele constau în faptul că evenimentele la care se poate răspunde nu sunt limitate la nivel de limbaj, în Visual Basic ascunderea cozii de evenimente limitează evenimentele ce se pot percepe şi prelucra din limbaj.

În Java, mediul de programare, implementează oarecum o varianta imtermediară între Visual Basic şi C. Adică, în limita evenimentelor recunoscute de AWT, controlul programatorului este complet asupra evenimentelor plecând de la sursa de evenimente (butoane, mouse) şi până la listener-ul de eveniment (rutina de tratare). Orice obiect poate să fie listener, iar ca urmare a acestui proces de delegare al tratării evenimentului flexibilitatea codării este mai mare decât cea din Visual Basic, unde listener-urile sunt predefinite, dar este nevoie de mai mult cod pentru implementarea acestei libertăţi. Sursele de evenimente au metode ce permit înregistrarea listener-urilor. Câna apare un eveniment la sursă, aceasta trimite o notificare de apariţie a evenimentului către toate listener-ele de obiecte ce s-au înregistrat pentru acel eveniment. Java, fiind un limbaj orientat pe obiect, informaţiile legate de eveniment le încapsulează într-un obiect. Clasa de bază pentru orice eveniment este `java.util.EventObject`. Pentru fiecare eveniment specific există câte o subclasă cum sunt `ActionEvent` sau `WindowEvent`. Înregistrarea unui listener pentru o anumită sursă de evenimente se face conform secvenţei de code ce urmează:

```
obiectSursadeEveniment.addEventListener(listenerEvenimentObiect);
```

Iată un exemplu pentru cazul unui obiect `JButton`.

```
ActionListener list = ...;
JButton bu1 = new Jbutton("Calculeaza");
bu1.addActionListener(list);
```

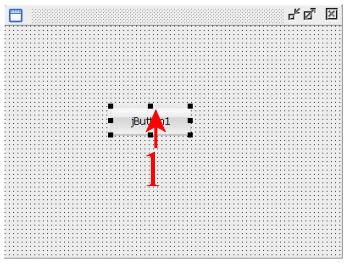
Prin acest cod listener-ul `list` este notificat când apare un eveniment în butonul `bu1`. Codul de mai sus necesită tipic o clasă de care listere-ul să aparţină implementânnd interfaţa corespunzătoare, în acest caz interfaţa `ActionListener`. Pentru implementarea acestei interfeţe listener-ul trebuie să aibă o metodă numită `actionPerformed` ce primeşte obiectul `ActionEvent` ca şi parametru:

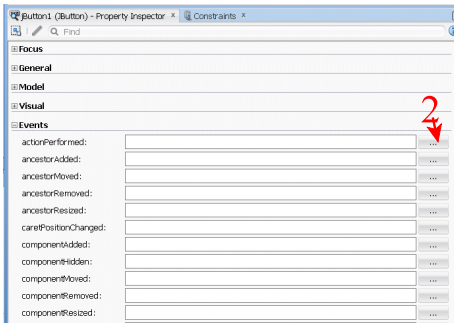
```
class buListener implements ActionListener {
      . . .
      public void actionPerformed(ActionEvent e) {
            //codul de tratare al evenimentului vine aici
            . . .
      }
}
```

De fiecare dată când utilizatorul apsă butonul `bu1` el crează un obiect `ActionEvent` şi apelează `list.actionPerformed(e)`, transferând obiectul eveniment. Este posibil ca mai multe obiecte să fie adăugate ca listener-uri pentru o sursă de evenimente. În aceste caz sunt apelate toate metodele `actionPerformed` ale listener-elor.

Considerăm codul de mai jos, el este generat automat de JDeveloper la crearea unui buton pe un Frame.

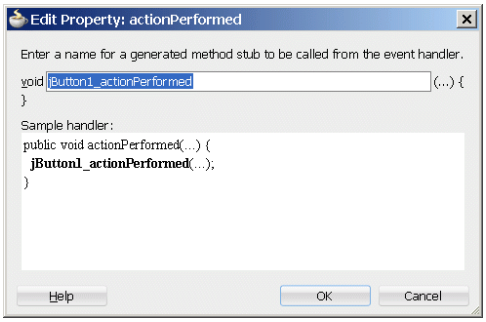
```
public class Frame1 extends JFrame {
    private JButton jButton1 = new JButton();

    public Frame1() {
        try {
            jbInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
         this.getContentPane().setLayout( null
);
         this.setSize( new Dimension(400, 300)
);
        jButton1.setText("jButton1");
         jButton1.setBounds(new Rectangle(125,
95, 95, 35));
            this.getContentPane().add(jButton1,
null);
    }
}
```





După selectarea butonului (etapa 1) şi după ce realizăm clic pe evenimentul `actionPerformed` (etapa 2), pe ecran se afişază o fereastă ce ne permite să alegem numele rutinei de tratare a evenimetului. La apăsarea butonului OK generatorul din Jdeveloper scrie automat codul de asociere

între obiectul sursă, eveniment şi rutina de tratare aevenimentului, după cum se observă în codul următor.




```
public class Frame1 extends JFrame {
    private JButton jButton1 = new JButton();

    public Frame1() {
        try {
            jbInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        this.getContentPane().setLayout( null );
        this.setSize( new Dimension(400, 300) );
        jButton1.setText("jButton1");
        jButton1.setBounds(new Rectangle(125, 95, 95, 35));
        jButton1.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    jButton1_actionPerformed(e);
                }
            });
        this.getContentPane().add(jButton1, null);
    }

    private void jButton1_actionPerformed(ActionEvent e) {
          //aici se scrie codul de tratare a evenimetului
          ...
    }
}
```

C11- Întrebari

1. Definiti sablonul model-view-controller.
2. Ce este un layout manager?
3. Ce componente Swing cunoasteti pentru introducerea de date?
4. Ce este programarea orientata pe evenimente?
5. Enumerati cateva surse de evenimente si de componente ce le pot trata.
6. Ce facilitati are JDeveloper pentru tratarea evenimentelor?