Poo Laboratoare[1]

Contents

[1] http://www.google.ro/

Laborator8

Programare Orientată pe Obiecte: Laborator 8

1 Introducere în Swing

1.1 Overview

De la apariția limbajului Java, bibliotecile de clase care oferă servicii grafice au suferit probabil cele mai mari schimbări în trecerea de la o versiune la alta. Acest lucru se datorează, pe de o parte dificultății legate de implementarea noțiunii de portabilitate, pe de altă parte nevoii de a integra mecanismele GUI cu tehnologii apărute și dezvoltate ulterior, cum ar fi **Java Beans**.

In momentul actual, există două modalități de a crea o aplicație cu interfață grafică și anume:

- **AWT (Abstract Windowing Toolkit)** - este API-ul inițial pus la dispoziție începând cu primele versiuni de Java
- **Swing** – este parte dintr-un proiect mai amplu numit **JFC (Java Foundation Classes)** creat în urma colaborării dintre **Sun**, **Netscape** și **IBM**, care **se bazează pe modelul AWT**, extinzând funcționalitatea acestuia și adăugând sau înlocuind unele componente pentru dezvoltarea aplicațiilor GUI.

Este preferabil ca aplicațiile Java să fie create folosind tehnologia Swing, deoarece aceasta pune la dispoziție o paletă mult mai largă de facilități, însă nu se va renunța complet la AWT deoarece aici există clase esențiale, reutilizate în Swing. Dezavantajul AWT-ului este că arhitecții lui au fost nevoiți să ia în considerare numai acele clase de obiecte grafice, care există într-o formă sau alta pe toate platformele. Aceasta a făcut ca numărul de clase de obiecte grafice din pachetul **java.awt** să fie destul de restrâns, renunțându-se la funcționalitățile specifice numai anumitor platforme.

1.2 Pachetul javax.swing

Componentele Swing, spre deosebire de predecesoarele din versiunile Java anterioare, sunt implementate în întregime în Java. Aceasta are ca rezultat o mai bună compatibilitate cu platforme diferite decât în cazul folosirii componentelor AWT. Unul din principalele deziderate ale tehnologiei Swing a fost să pună la dispoziție un set de componente GUI extensibile care să permită dezvoltarea rapidă de aplicații Java cu interfață grafică competitivă din punct de vedere comercial. Cel mai important pachet, care conține componentele de bază este **javax.swing**.

Orice interfață utilizator Java este compusă din următoarele elemente:

- **Componente** – orice poate fi plasat pe o interfață utilizator, cum ar fi butoane, liste de derulare, meniuri pop-up, casete de validare sau câmpuri de text.
- **Containere** – acestea reprezintă componente care pot conține alte componente (de exemplu panouri, casete de dialog sau ferestre independente)
- **Administratori de dispunere** – reprezintă obiecte care definesc modul în care sunt aranjate (dispuse) componentele într-un container. Administratorul de dispunere nu este vizibil într-o interfață, însă sunt vizibile rezultatele "muncii" sale. Dispunerea componentelor interfeței este de mai multe feluri: dispunere secvențială, dispunere tabelară, dispunere marginală sau dispunere tabelară neproporțională.

1.3 Componente și containere

Componentele Swing sunt derivate dintr-o singură clasă de bază, numită **JComponent**, care moștenește la rândul ei clasa **Container** din **AWT**. Componentele folosite pentru crearea interfețelor grafice Swing pot fi grupate astfel:

- Componente atomice: **JLabel**, **JButton**, **JCheckBox**, **JRadioButton**, **JToggleButton**, **JScrollBar**, **JSlider**, **JProgressBar**, **JSeparator**
- Componente complexe: **JTable**, **JTree**, **JComboBox**, **JSpinner**, **JList**, **JFileChooser**, **JColorChooser**, **JOptionPane**
- Componente pentru editare de text: **JTextField**, **JFormattedTextField**, **JPasswordField**, **JTextArea**, **JEditorPane**, **JTextPane**
- Meniuri: **JMenuBar**, **JMenu**, **JPopupMenu**, **JMenuItem**, **JCheckboxMenuItem**, **JRadioButtonMenuItem**
- Containere intermediare: **JPanel**, **JScrollPane**, **JSplitPane**, **JTabbedPane**, **JDesktopPane**, **JToolBar**, **JLayeredPane**, **JRootPane**
- Containere de nivel înalt: **JFrame**, **JDialog**, **JWindow**, **JInternalFrame**, **JApplet**

Containerele reprezintă suprafețe de afișare pe care pot fi plasate alte componente, eventual chiar alte containere. **Superclasa** componentelor de acest tip este **Container**, din modelul **AWT**.

1.3.1 Containerele de nivel înalt

Pentru a fi afișate pe ecran componentele grafice ale unei aplicații trebuie plasate pe o **suprafață de afișare (container**). Fiecare componentă poate fi conținută doar într-un singur container, adăugarea ei pe o suprafață nouă de afișare determinând eliminarea ei de pe vechiul container pe care fusese plasată. Deoarece containerele pot fi încapsulate în alte containere, o componentă va face parte la un moment dat dintr-o ierarhie. Rădăcina acestei ierarhii trebuie să fie un așa numit container de nivel înalt, care este reprezentat de una din clasele **JFrame**, **JDialog** sau **JApplet**.

În general orice aplicație Java independentă bazată pe Swing conține cel puțin un container de nivel înalt reprezentat de fereastra principală a programului, instanță a clasei JFrame.

1.3.2 Containerele intermediare

Acestea reprezintă suprafețe de afișare cu ajutorul cărora pot fi organizate mai eficient componentele aplicației, putând fi imbricate.

JPanel are aceeași funcționalitate ca și clasa **Panel** din **AWT**, fiind folosit pentru gruparea mai multor componente Swing și plasarea lor împreună pe o altă suprafață de afișare. Gestionarul de poziționare implicit este **FlowLayout**, acesta putând fi schimbat însă, chiar în momentul construirii obiectului JPanel, sau ulterior cu metoda **setLayout()**. Adăugarea de componente se realizează ca pentru orice container, folosind metoda **add()**.

JScrollPane este o clasă foarte importantă în arhitectura modelului Swing, deoarece oferă suport pentru derularea pe orizontală și verticală a componentelor a căror reprezentare completă nu încape în suprafața asociată, nici o componentă Swing neoferind suport intrinsec pentru această operație.

Clasa **JComponent** este **superclasa** tuturor componentelor Swing, mai puțin a celor care descriu containere de nivel înalt JFrame, JDialog, JApplet. Deoarece **JComponent extinde clasa Container,** deci și Component, ea moștenește funcționalitatea generală a containerelor și componentelor AWT, furnizând bineînțeles și o serie întreagă de noi facilități.

1.3.3 Componente pentru editare de text

Componentele Swing pentru afișarea și editarea textelor sunt grupate într-o ierarhie ce are ca **rădăcină clasa JTextComponent** din pachetul **javax.swing.text**. Clasele pot împărțite în trei categorii, corespunzătoare tipului textului editat:

- Text simplu pe o singură linie
  - **JTextField** - permite editarea unui text simplu, pe o singură linie

  - **JPasswordField** - permite editarea de parole. Textul acestora va fi ascuns, în locul caracterelor introduse fiind afișat un caracter simbolic, cum ar fi '*'.

- Text simplu pe mai multe linii
  - **JTextArea** - permite editarea unui text simplu, pe mai multe linii. Orice atribut legat de stil, cum ar fi culoarea sau fontul, se aplică întregului text și nu poate fi specificat doar unei anumite porțiuni. Uzual, o componentă de acest tip va fi inclusă într-un container **JScrollPane**, pentru a permite navigarea pe verticală și orizontală dacă textul introdus nu încape în suprafața alocată obiectului. Acest lucru este valabil pentru toate componentele Swing pentru care are sens noțiunea de navigare pe orizontală sau verticală, nici una neoferind suport intrinsec pentru această operațiune.

- Text cu stil îmbogățit pe mai multe linii
  - **JEditorPane** - permite afișarea și editarea de texte scrise cu stiluri multiple și care pot include imagini sau chiar diverse alte componente
  - **JTextPane** - această clasă **extinde JEditorPane**, oferind facilități suplimentare pentru lucrul cu stiluri și paragrafe

1.3.4 Componente atomice

În categoria componentelor atomice sunt incluse componentele Swing cu funcționalitate simplă, a căror utilizare este facilă și în general asemănătoare cu a echivalentelor din AWT:

- Etichete: **JLabel**
- Butoane simple sau cu două stări: **JButton**, **JCheckBox**, **JRadioButton**. Mai multe butoane radio pot fi grupate folosind clasa **ButtonGroup**, pentru a permite selectarea doar a unuia dintre ele.
- Componente pentru progres și derulare: **JSlider**, **JProgressBar**, **JScrollBar**
- Separatori: **JSeparator**

1.3.5 Componente complexe

Clasa **JList** descrie o listă de elemente dispuse pe una sau mai multe coloane, din care utilizatorul poate selecta unul sau mai multe. Uzual un obiect de acest tip va fi inclus într-un container de tip **JScrollPane**. Clasa oferă metode pentru selectarea unor elemente din cadrul programului **setSelectedIndex()**, **setSelectedIndices()**, etc. și pentru obținerea celor selectate la un moment dat **getSelectedIndex()**, **getSelectedIndices()**, etc..

Clasa **JComboBox** este similară cu JList, cu deosebirea că permite doar selectarea unui singur articol, acesta fiind și singurul permanent vizibil. Lista celorlalte elemente este afișată doar la apăsarea unui buton marcat cu o săgeată, ce face parte integrantă din componentă. JComboBox funcționează după aceleași principii ca și clasa JList.

Clasa **JTable** permite crearea de componente care să afișeze o serie de elemente într-un format tabelar, articolele fiind dispuse pe linii și coloane. Un tabel poate fi folosit doar pentru afișarea formatată a unor date, dar este posibilă și editarea informației din celulele sale. De asemenea, liniile tabelului pot fi marcate ca selectate, tipul selecției fiind simplu sau compus, tabelele extinzând astfel funcționalitatea listelor. O serie de clase și interfețe necesare lucrului cu tabele se găsesc în pachetul **javax.swing.table**, acesta fiind așadar cel ce trebuie importat.

1.4 Concluzie

Până acum s-a încercat crearea unei imagini de ansamblu asupra facilităților oferite de Swing. Descrierea API-ului pe care îl pune la dispoziție fiecare clasa prezentată mai sus presupune abuzarea spațiului de afișare și nu constituie subiectul acetui laborator. Pentru viziualizarea în amănunt a metodelor și

membrilor de clasă, consultați **documentația oficială Sun**. De asemenea, pentru mici exemple de utilizare consulați **materialul de curs**.

2 Probleme de laborator

Observație! Se vor utiliza clasele Swing corespunzătoare celor AWT: JFrame pentru Frame, JButton pentru Button, JTextField pentru TextField etc.

2.1 Problema 1

(1 punct) Program pentru afișarea unei ferestre cu titlu folosind clasa JFrame. Funcția **main()** creează un obiect JFrame, stabilește dimensiunile și cere afișarea acestuia. Veți folosi metodele **setSize()**, respectiv **setVisible()**. Să se observe efectul unui clic pe butonul de închidere (X). Pentru terminarea aplicației tastați CTRL-C. Adăugați apoi următoarea instrucțiune, unde *jf* este obiectul JFrame tocmai creat:

```
jf.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
```

2.2 Problema 2

(1 punct) Pentru afișarea unei ferestre centrate pe mijlocul ecranului, se va folosi funcția următoare (ca metodă statică, separată de clasa JFrame):

```
public static void center (Window w) {
       Dimension ws = w.getSize();
       Dimension ss = Toolkit.getDefaultToolkit().getScreenSize();
       int newX = ( ss.width - ws.width ) / 2;
       int newY = ( ss.height- ws.height ) / 2;
       w.setLocation( newX, newY );
}
```

Să se adauge metoda de mai sus ca metodă a clasei anterioare.

2.3 Problema 3

(2 puncte) Program pentru afișarea unei etichete JLabel urmată de un câmp text JTextField nemodificabil (se va folosi metoda **setEditable()** cu parametrul **false**), pentru afișarea numelui și valorii unei proprietăți (de exemplu numele *Color* în etichetă și valoarea *White* în campul text). Se va folosi așezarea FlowLayout. Se vor adăuga apoi:

- bordură la câmpul text (se va folosi **setBorder(new EtchedBorder())**)
- culoarea albă la câmpul text (se va folosi **setBackground(Color.white)**)
- dimensiunea dorită la câmpul text (de exemplu **setPreferredSize(new Dimension(180, 20))**)

2.4 Problema 4

(2 puncte) Program care adaugă unei ferestre JFrame (având dimensinile 240, respectiv 200) cinci butoane cu numele extrase din vectorul următor:

```
String pos[]= { ``East'', ``West'', ``North'', ``South'', ``Center'' };
```

- se va extrage din fereastră container-ul acesteia folosind apelul:

```
Container cp = frame.getContentPane();
```

- se va adăuga la acest panou (într-un ciclu) fiecare buton folosind metoda **add()** cu doi parametri:
  - numele variabilei buton JButton
  - un șir care arată poziția butonului în panou (același cu inscripția butonului din vectorul *pos*)

Observație! Modul de așezare implicit este BorderLayout.

- se va elimina al doilea argument al metodei **add()** și se va observa efectul
- se va modifica modul de așezare în fereastra principala la FlowLayout

```
frame.setLayout ( new FlowLayout() );
```

- se va modifica modul de așezare în fereastra principală la GridLayout, folosind succesiv constructor fără argumente sau cu două argumente întregi:
  - primul argument 0
  - al doilea argument numărul de coloane prin care se dispun componentele în panou: 0, 1, 2

2.5 Problema 5

(2 puncte) Program care folosește un vector de opt butoane JButton și un vector de panouri JPanel. Se va adăuga fiecare buton unui panou separat și apoi se va adauga fiecare panou ferestrei principale. Dispunerea în container-ul principal este GridLayout pe două coloane. Se va compara adăugarea directă a butoanelor la fereastră cu adăugarea prin panouri separate. Fereastra principală poate avea dimensiunile (200, 200). Pentru crearea de intervale între panouri se poate folosi metoda **setBorder()**:

```
pp[i].setBorder( new EmptyBorder( 1, 1, 1, 1 ) );
```

sau

```
pp[i].setBorder( BorderFactory.createEmptyBorder( 1, 1, 1, 1 ) );
```

2.6 Problema 6

(2 puncte) Program care adaugă ferestrei principale două panouri, unul sub altul:

- un panou cu un câmp text JTextField și o eticheta JLabel ce conține șirul *Open*. Eticheta poate fi în stânga sau deasupra câmpului text. Câmpul text are lungime de 30 de caractere (se va folosi **new JTextField(" ", 30)**)
- un panou cu trei butoane în linie, notate *OK*, *Cancel* și *Browse*.

Fereastra principală are dimensiunile (400, 160). Faceți modificările necesare pentru ca fereastra afișată să semene cât mai bine cu fereastra produsă de comanda *Run* din Windows (cu titlul *Run*). Modificați în programul anterior stilul de desenare (Look and Fell) pentru sistemul gazdă folosind metoda **setLookAndFeel()** astfel:

```
try {
      UIManager.setLookAndFeel( UIManager.getSystemLookAndFeelClassName() );
} catch( Exception e ) {}
```

2.7 Problema 7 (bonus)

(2 puncte) Program pentru afișarea unui buton cu inscripția *Colors* și modificarea culorii acestuia ca urmare a unui clic pe buton. Metoda **setForeground()** cu parametru de tip **Color** are efectul de modificare a culorii textului, iar metoda **setBackground()** modifică culoarea butonului. La fiecare clic se va itera pe un vector de culori initializat cu constante (Color.RED, Color.BLUE, etc.). Se va folosi o fereastră JFrame cu dimensiuni mici (100, 100) și organizarea FlowLayout. Să se prevadă și posibilitatea de acționare a butonului prin tasta ALT-C:

```
buton.setMnemonic( KeyEvent.VK_C );
```