Programare orientată pe obiecte

Breviar - Laboratorul 10

Mihai Nan
mihai.nan.cti@gmail.com

Facultatea de Automatică şi Calculatoare
Universitatea Politehnica din Bucureşti
Anul universitar 2015 - 2016

1 Interfete grafice

1.1 Introducere

In cazul programelor pe care le-am facut pana acum, toate mesajele si raspunsurile apareau ca linii de text sugestive, ecranul fiind folosit in mod text. Un astfel de stil de comunicare nu este atractiv pentru utilizatori, motiv pentru care se prefera dialogul prin interfete grafice sau ***GUI*** (***Graphical User Interface***), ecranul fiind folosit in mod grafic.

In trecerea de la o versiune la alta, bibliotecile de clase care ofera servicii grafice au suferit, probabil, cele mai mari schimbari. Acest lucru se datoreaza, pe de o parte, dificultatii legate de implementarea notiunii de portabilitate, iar pe de alta parte nevoii de a integra mecanismele GUI cu tehnologii aparute si dezvoltate ulterior.

In momentul actual, exista doua modalitati de a crea o aplicatie cu interfata grafica, iar acestea sunt:

- ***AWT (Abstract Windowing Toolkit)*** - este API-ul initial pus la dispozitie incepand cu primele versiuni de Java;
- ***Swing*** - este parte dintr-un proiect mai amplu numit ***JFC (Java Foundation Classes)*** creat in urma colaborarii dintre *Sun*, *Netscape* si *IBM*, care se bazeaza pe modelul ***AWT***, extinzand functionalitatea acestuia si adaugand sau inlocuind unele componente pentru dezvoltarea aplicatiilor GUI.

⚠ IMPORTANT !

> ⚠ Este preferabil ca aplicatiile Java sa fie create folosind tehnologia ***Swing***, deoarece aceasta pune la dispozitie o paleta mult mai larga de facilitati, insa nu se va renunta complet la ***AWT***, deoarece aici exista clase esentiale, reutilizate in ***Swing***.

1.2 Pachetul Swing

Componentele ***Swing***, spre deosebire de predecesoarele din versiunile Java anterioare, sunt implementate in intregime in Java. Aceasta are ca rezultat o mai buna compatibilitate cu platforme diferite decat in cazul folosirii componentelor ***AWT***. Unul din principalele deziderate ale tehnologiei ***Swing*** a fost sa puna la dispozitie un set de componente GUI extensibile care sa permita dezvoltarea rapida de aplicatii Java cu interfata grafica competitiva, din punct de vedere comercial. Cel mai important pachet, care contine componentele de baza este ***javax.swing***.

Orice interfata utilizator Java este compusa din urmatoarele elemente:

- **Componente** – orice poate fi plasat pe o interfata utilizator, cum ar fi *butoane, liste de derulare, meniuri pop-up, casete de validare* sau *campuri de text*;
- **Containere** – acestea reprezinta componente care pot conţine alte componente (de exemplu *panouri, casete de dialog* sau *ferestre independente*);
- **Administratori de dispunere** – reprezinta obiecte care definesc modul in care sunt aranjate (dispuse) componentele intr-un container. Administratorul de dispunere nu este vizibil intr-o interfata, insa sunt vizibile rezultatele "muncii" sale. Dispunerea componentelor interfetei este de mai multe feluri: dispunere secventiala, dispunere tabelara, dispunere marginala sau dispunere tabelara neproportionala.

1.2.1 Crearea ferestrelor

Clasa ***JFrame*** este cea pe care o vom folosi pentru a crea ferestre. Ca orice alta clasa care reprezinta componente ***Swing*** ea se afla in pachetul ***javax.swing***. Pentru ca este un container, vom folosi, de cele mai multe ori, aceasta clasa prin mostenire nu prin instantiere. Altfel spus, vom crea clase care sa reprezinta ferestre si pentru ca acestea sa devina ferestre de tip ***JFrame*** ele vor mosteni aceasta clasa.

Cod sursa Java

```
1  public class Fereastra extends JFrame {
2      private JPanel panel;
3
4      public Fereastra(String text) {
5          super(text);
6          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7          this.setMinimumSize(new Dimension(300, 200));
8          this.getContentPane().setBackground(Color.blue);
9          this.setLayout(null);
10         panel = new JPanel();
11         panel.setBounds(40, 40, 100, 70);
12         this.add(panel);
13         this.setLocation(50, 70);
14         this.show();
15         this.pack();
16     }
17
18     public static void main(String args[]) {
19         Fereastra f = new Fereastra("Laborator POO");
20     }
21 }
```

Metoda ***add(Component c)*** este folosita pentru a adauga pe fereastra o componenta exact ca in cazul Appleturilor. Metoda ***add()*** este mostenita din clasa ***Container***.

1.2.2 Crearea butoanelor

Un buton poate fi creat folosind clasa ***JButton***. De obicei, butonul este contruit folosind unul dintre constructorii:

- ***public JButton(); // un buton fara text***
- ***public JButton(String text) // un buton cu text dat ca parametru***
- ***public JButton(String text, Icon ico) // buton cu text si imagine***

Textul de pe buton poate fi modificat folosind metoda ***setText(String text)*** sau poate fi preluat folosind metoda ***getText()***. Metodele ***setLabel()*** si ***getLabel()*** sunt considerate *obsolete* si nu se mai folosesc in prezent.

Cod sursa Java

```
class Button extends JFrame implements ActionListener {
    private JButton button;

    public Button(String text) {
        super(text);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setMinimumSize(new Dimension(300, 200));
        this.getContentPane().setBackground(Color.blue);
        this.setLayout(null);
        button = new JButton("Apasa");
        button.setBounds(100, 60, 70, 30);
        button.addActionListener(this);
        this.add(button);
        this.setLocation(50, 70);
        this.show();
        this.pack();
    }

    public static void main(String args[]) {
        Button b = new Button("Laborator POO");
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();
        if(button.getText().equals("Apasa")) {
            System.out.println("Butonul a fost apasat!");
        }
    }
}
```

Din moment ce butonul reprezinta o componenta, poate fi adaugat pe un container folosind metoda ***add(Component c)*** a containerului. Butonul ***JButton*** este sensibil la evenimente de tip ***ActionEvent***, asadar, modalitatea de atasare a ascultatorilor si de creare evenimente este similara butoanelor ***Button*** din pachetul ***java.awt***.

1.2.3 Crearea componentelor text

Cod sursa Java

```
class Text extends JFrame implements ActionListener {
    private JButton button;
    private JTextField user;
    private JPasswordField pass;

    public Text(String text) {
        super(text);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setMinimumSize(new Dimension(300, 200));
        this.getContentPane().setBackground(Color.blue);
        this.setLayout(null);
        button = new JButton("Apasa");
        button.setBounds(100, 100, 70, 30);
        button.addActionListener(this);
        this.add(button);
        user = new JTextField();
        user.setBounds(50, 20, 170, 30);
        this.add(user);
        pass = new JPasswordField();
        pass.setBounds(50, 60, 170, 30);
        this.add(pass);
        this.show();
        this.pack();
    }

    public static void main(String args[]) {
        Text b = new Text("Laborator POO");
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();
        if(button.getText().equals("Apasa")) {
            System.out.println(user.getText()+pass.getText());
        }
    }
}
```

Vom folosi trei tipuri de componete text: ***JTextField***, ***JTextArea*** si ***JPasswordField***. Componenta ***JTextFiled*** este un camp de text cu un singur rand pe care pot incapea mai multe caractere. In general, aceasta componenta este folosita pentru a introduce un text de dimensiuni mici. Componenta ***JTextArea*** este un camp de text care permite introducerea unui text pe mai multe randuri, deci, aceasta componenta este folosita pentru texte de dimensiuni mai mari. Componenta ***JPasswordField*** este similara componentei ***JTextField***, doar ca, aceasta ascunde caracterele introduse de utilizator ca in cazul unui camp pentru introducerea parolei obisnuit.

Metoda folosita pentru preluarea textului dintr-o componenta text este: ***getText()*** care returneaza o instanta de ***String***. Pentru a modifica textul dintr-un camp de text, se foloseste metoda ***setText(String text)***. Componenta ***JTextArea*** are in comportament si metoda ***append(String text)*** care permite adaugarea unui text la componenta.

1.2.4 Butoane de selectie

Clasele ***JCheckBox*** si ***JRadioButton*** definesc componente de tip butoane de selectie. Standard, butoanele ***JCheckBox*** sunt folosite pentru a crea liste de optiuni de tip multiple-choice (din care poti selecta mai multe variante), iar ***JRadioButton*** pentru crearea de liste de tip single-choice (din care se poate selecta o singura optiune). Astfel, pentru butoanele ***JRadioButton*** adaugam o noua clasa ***ButtonGroup*** cu ajutorul careia precizam care sunt butoanele din care se selecteaza o singura optiune. ***ButtonGroup*** defineste un grup de butoane. Din butoanele ce apartin aceluiasi grup, nu putem selecta decat o singura optiune.

Ambele tipuri de componente se creaza similar butoanelor ***JButton***. In general, acestea nu sunt folosite cu evenimente si prin verificari ale starilor acestora (se verifica daca este sau nu select la un anumit eveniment). Verificarea starii unei astfel de componente se face folosind medoda ***isSelected()*** care returneaza **true** daca butonul este bifat si **false** daca nu este bifat.

Observatie

Pentru o intelegere mai buna, se recomanda analizarea exemplelor propuse in arhiva laboratorului.

1.2.5 JScrollPane

JScrollPane este o componenta folosita pentru adaugarea de bare de defilare pentru o alta componenta care ar putea depasi dimensiunile containerului in care este adaugata. Instanta ***JScrollPane*** nu este folosita in actiunile directe pe care le are componenta pe care o sustine. La construire instantei, i se da o

componenta pentru care este creata si pe care adauga bare de defilare (scrollbars). Apoi, instanta ***JScrollPane*** este adaugata pe container in locul componentei din instanta ***JScrollPane***.

Cod sursa Java

```
class Text extends JFrame {
    private JTextArea textArea;
    private JScrollPane scroll;

    public Text(String text) {
        super(text);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setMinimumSize(new Dimension(300, 200));
        this.getContentPane().setBackground(Color.blue);
        this.setLayout(null);
        textArea = new JTextArea(200, 100);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setFont(new Font("Tahoma", 2, 12));
        scroll = new JScrollPane(textArea);
        scroll.setBounds(10, 10, 250, 130);
        this.add(scroll);
        this.show();
        this.pack();
    }

    public static void main(String args[]) {
        Text b = new Text("Laborator POO");
    }
}
```

1.2.6 Etichete

O eticheta reprezinta cea mai simpla componenta. Aceasta este folosita pentru afisarea unui text static in cele mai multe cazuri. Textul de pe o eticheta poate fi modificat sau preluat folosind metodele ***getText()*** si ***setText(String text)***.

⚠ IMPORTANT !

⚠ Un ***JLabel*** poate fi uzitat si pentru afisarea unei imagini. In acest caz, nu i se aplica niciun text, ci doar un ***imageIcon***, aceasta modalitate fiind considerata cea mai simpla de a afisa o imagine intr-un container.

1.3 Tratarea evenimentelor

Un eveniment este produs de o actiune a utilizatorului asupra unei componente grafice si reprezinta mecanismul prin care utilizatorul comunica efectiv cu programul. Exemple de evenimente sunt: apasarea unui buton, modificarea textului intr-un control de editare, inchiderea sau redimensionarea unei ferestre, etc. Componentele care genereaza anumite evenimente se mai numesc si surse de evenimente.

Interceptarea evenimentelor generate de componentele unui program se realizeaza prin intermediul unor clase de tip listener (ascultator, consumator de evenimente). In Java, orice obiect poate "consuma" evenimentele generate de o anumita componenta grafica.

Avand in vedere gama larga de evenimente existente in limbajul Java, am ales sa vi le prezint pe cele considerate mai important, intr-o serie de exemple incluse in arhiva laboratorului.