TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 9

Applet–uri Java.
Componentele interfeței grafice.

I. SCOPUL LUCRĂRII

Lucrarea de față are rolul de a prezenta şi familiariza studentul cu modul de construire a unei interfețe grafice utilizator. Se vor prezenta câteva componente vizuale utile, împreună cu modul de creare şi dispunere a acestora pe suprafața unui applet.

La sfârşitul acestei lucrări, studentul va avea posibilitatea să scrie mini-aplicații Java în care să utilizeze noțiunile învățate.

II. NOȚIUNI TEORETICE

1. Construirea unei interfețe grafice

Interfețele grafice utilizator **GUI** (graphic user interface) sunt o parte importantă a oricărui program. Pachetul **java.awt** (Abstract Window Toolkit) furnizează funționalități extinse în acest sens.

1.1 Componentele interfeței grafice

Componentele Java sunt implementate printr-o serie de subclase ale superclasei **java.awt.Component** şi ale superclasei **java.awt.MenuComponent.**

O modalitate de a grupa componentele grafice este următoarea: componente vizuale, componente container şi componene meniu.

Există 11 componente vizuale. În cele ce urmează vom prezenta numai o parte dintre acestea: **Button, Checkbox, Choice, Label, List, TextArea, TextField**. Pentru a utiliza una dintre acestea într-o GUI, se va crea întâi o instanță prin apelarea constructorului corespunzător, apoi se va adăuga componenta unui container.

Există 4 componente container: **Applet, Frame, Panel, Dialog** (vor fi prezentate în următoarele lucrări de laborator). Container-ele sunt componente capabile să conțină alte componente în cadrul lor.

Câteva metode sunt implementate de toate componentele vizuale şi container, în virtutea moştenirii din clasa **java.awt.Component**.

- ***getSize( )*** – returnează dimensiunea unei componente. Tipul returnat este **Dimension**, care are datele membru publice **height** şi **width**.
- ***setForeground( ), setBackground( )*** – setează culoarea de scriere a textului şi respectiv culoarea de fond pentru o componentă. Fiecare primeşte ca argument o

instanță a clasei **java.awt.Color**. Dacă aceste culori nu se vor seta explicit, se vor utiliza culorile implicite ale containerului căruia îi aparține componenta.

- ***setFont( )*** – determină font-ul pe care o componentă îl va utiliza la scrierea textului. Dacă nu se setează explicit un font, componenta va utiliza font-ul containerului său.
- ***setSize( ), setBounds( )*** – stabilesc dimensiunile unei componente. Metoda **setSize()** primește 2 argumente: width (lățime) şi height (înălțime). Metoda **setBounds()** stabilește atât poziția cât şi dimensiunea. Poziția este specificată relativ la containerul componentei, sau în cazul unei ferestre relativ la ecran. O formă a metodei are 4 argumente: x, y , width şi height.
- ***setVisible( )*** – primește un argument de tip Boolean şi stabilește dacă componenta va fi vizibilă sau nu. Se utilizează în general pentru Frame-uri.

1.2 Layout Managers

Alte sisteme GUI încurajează programatorul să gândească în termenii unei specificări precise a dimensiunii şi locației componentelor interfeței grafice. Java, în schimb, furnizează o serie de 5 **layout managers**, fiecare având propria politică de dispunere a componentelor în containere.

Fiecare componentă Java are o **dimensiune preferată**. Dimensiunea preferată este în general cea mai mică dimensiune necesară pentru a reprezenta componenta într-un mod vizual ce are sens. De exemplu, dimensiunea preferată a unui buton este dimensiunea etichetei sale, plus o mică margine de spațiu liber din jurul textului, plus decorațiunile care marchează marginea butonului. Dimensiunea preferată este dependentă de platformă, deoarece decorațiunile marginii componentei variază de la un sistem la altul. Când un **layout manager** așează componentele unui container, trebuie să ia în considerare 2 criterii: politica sa şi dimensiunea preferată a fiecărei componente. (Prima prioritate este de a respecta politica de aşezare a componentelor, ignorându-se dimesiunea preferată a componentelor.)

În această lucrare de laborator vom prezenta numai două dintre cele 5 **layout mangers**, şi anume **Flow Layout Manager** şi **Border Layout Manager**.

Flow Layout Manager aranjează componentele în rânduri orizontale. Este managerul implicit pentru applet-uri şi pentru panel-uri. Întotdeauna onorează dimensiunea preferată a componentelor.

Border Layout Manager este managerul implicit pentru frame-uri. Îşi divide teritoriul în 5 regiuni: NORTH, SOUTH, EAST, WEST şi CENTER. Fiecare regiune poate fi vidă sau poate conține o singură componentă.

2. Exemple

```
// App1.java
import java.applet.Applet;
import java.awt.*;

public class App1 extends Applet
{
   public void init()
   {
     setLayout(new FlowLayout(FlowLayout.RIGHT));
```

```
      Label label=new Label("Introduceti text: ");
      add(label);

      TextField text1=new TextField("");
      add(text1);
      TextField text2=new TextField("textul 2");
      add(text2);
      Button but=new Button("OK");
      add(but);

    }
    public void paint(Graphics g)
    {
       g.drawString("Acesta este un exemplu",30,100);
    }
}


// App1.html
<html>
<body>
<applet code=App1.class width=250 height=400>
</applet>
</body>
</html>
```

Exercițiu:

- Comentați prima linie de cod din metoda **init()** a applet-ului.
- Modificați în **App1.html** lățimea applet-ului; setați **width=350** şi reîncărcați pagina html.

În ambele situații observați şi explicați rezultatele.

```
//App2.java
import java.applet.Applet;
import java.awt.*;

public class App2 extends Applet
{
  public void init()
  {
    setLayout(new BorderLayout());
    Panel toolbar=new Panel();
    toolbar.setLayout(new FlowLayout(FlowLayout.LEFT));
    toolbar.setBackground(Color.orange);
    toolbar.add(new Button("Buton 1"));
    toolbar.add(new Button("Buton 2"));
    add(toolbar,BorderLayout.NORTH);
    TextArea txA1=new TextArea();
    StringBuffer s=new StringBuffer("Acesta este un text \n mai
                                    lung. Ca sa observati ");
    s.append("\n plasarea \n sagetilor \n de defilare");
    txA1.setText(s.toString());
    txA1.setFont(new Font("Arial",Font.ITALIC,24));
    txA1.setBackground(Color.blue);
```

```
   txA1.setForeground(Color.white);
   add(txA1,BorderLayout.CENTER);

   TextArea txA2=new TextArea("Al doilea text.",5,20);
   txA2.setFont(new Font("Monospaced",Font.ITALIC|Font.BOLD,20));
   txA2.setBackground(Color.white);
   txA2.setForeground(Color.blue);
   txA2.setEditable(false);
   add(txA2,BorderLayout.SOUTH);
   Checkbox chk=new Checkbox("Bifati aici!");
   add(chk,BorderLayout.WEST);
   Panel option=new Panel();
   CheckboxGroup chgroup=new CheckboxGroup();
   option.add(new Checkbox("Prima",false,chgroup));
   option.add(new Checkbox("A doua",true,chgroup));
   add(option,BorderLayout.EAST);
  }
}

//App2.html
<html>
<body>
<applet code=App2.class width=400 height=400>
</applet>
</body>
</html>


//App3.java
import java.applet.Applet;
import java.awt.*;
public class App3 extends Applet
{
   public void init()
   {
     setBackground(Color.cyan);
     setFont(new Font("Arial",Font.BOLD,16));
     Label label1=new Label("Checkbox:");
     add(label1);
     Choice ch=new Choice();
     ch.addItem("Prima");
     ch.addItem("A doua");
     ch.addItem("A treia");
     ch.setForeground(Color.red);
     add(ch);
     Label label2=new Label("List:");
     add(label2);
     List list=new List(3,true);
     for(int i=1;i<10;i++)
       list.add("Floare "+i);
     list.setForeground(Color.blue);
     list.setFont(new Font("Courier new",Font.ITALIC,20));
     add(list);
   }
}
```

```
//App3.html
<html>
<body>
<applet code=App3.class width=500 height=200>
</applet>
</body>
</html>
```

III. MODUL DE LUCRU

1. Se editează codul sursă al programului Java folosind un editor de text disponibil(de ex., se poate utiliza Notepad).
2. Se salvează fişierul cu extensia **.java**. Fişierul trebuie salvat la următoarea locație: **c:\JBulider7\jdk1.3.1\bin**
3. Compilarea mini-aplicației Java se va face din linia de comandă. Se poate proceda astfel. Se deschide o fereastră MS-Dos: Start ->Run, se tipăreşte **command** în căsuța de text şi se apasă butonul OK. Printr-o schimbare de directoare şi subdirectoare se trece la locația: **c:\JBulider7\jdk1.3.1\bin.** Sau, se lansează WindowsCommander. Se trece la locația **c:\JBulider7\jdk1.3.1\bin.** Se deschide o fereastră MS-Dos: Commander ->Run Dos.
4. Pentru compilare, se tipăreşte la prompter **javac nume_fişier_sursă.java** şi se apasă Enter. De ex., dacă fişierul se numeşte Test.java, se va scrie **javac Test.java**. În cazul în care programul conține erori acestea vor fi semnalate şi afişate în fereastra MS-Dos, după care va apare iar prompter-ul. Dacă programul nu conține erori şi compilarea se face cu succes, atunci va apare numai prompter-ul.
5. Se editează fişierul .**html**. Se salvează în acelaşi director cu fişierul **.class** rezultat în urma compilării cu succes a fişierului sursă java.
6. Se încarcă fişierul **.html** într-un browser Web (ex., Internet Explorer). Alternativ se poate utiliza comanda **appletviewer nume.html**.

IV. TEMĂ

1. Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic şi explicând rezultatele obținute.
2. Scrieți un applet Java care să conțină următoarele: o zonă de editare TextArea cu 5 linii si 10 coloane, un buton cu eticheta OK şi un grup de componente CheckBox. Setați textul zonei de editare la un anume şir de caractere. Utilizați diverse culori şi font-uri.