Curs 3

Principalele metode folosite în manipularea obiectelor JLabel:

Metoda	***Acţiune***
String getText()	Obţine Stringul afişat de către obiectul JLabel
void setText(String text)	Stabileşte printr-un String textul ce va fi afişat
int getHorizontalAlignment()	Obţine un întreg care indică modul de aliniere a conţinutului etichetei relativ la axa x
int getVerticalAlignment()	Obţine un întreg care indică modul de aliniere a conţinutului etichetei relativ la axa y
Icon getIcon()	Obţine obiectul Icon afişat de către obiectul JLabel
void setIcon(Icon icon)	Stabileşte obiectul Icon afişată de componenta JLabel
Component getLabelFor()	Obţine componenta care este etichetată folosind obiectul JLabel
void setLabelFor(Component c)	Stabileşte componenta care va fi etichetată cu obiectul JLabel
void setHorizontalAlignment(int alignament)	Stabileşte prin întregul specificat modul de aliniere relativ la axa x

void setVerticalAlignment(int alignament)	Stabileşte prin întregul specificat modul de aliniere relativ la axa y

Câmpuri pentru text JTextField

Biblioteca SWING introduce, pentru a gestiona câmpurile care pot primi text din partea utilizatorilor, componenta *JTextField* ce lucrează asemănător cu java.awt.*TextField* (dar care nu este derivată din aceasta).

java.lang.Object

| ___ java.awt.Component

| ___ java.awt.Container

| ___ javax.swing.JComponent

| ___ javax.swing.text.JTextComponent

| ___ javax.swing.JTextField

Constructorii principali ai componentei *JTextField* sunt:

Constructor	***Acţiune***
JTextField ()	Creează un nou câmp gol
JTextField (int columns)	Creează un nou obiect JTextField fără conţinut de lungimea indicată (în coloane)

JTextField(String text)	Creează un nou obiect JTextField conţinând textul specificat
JTextField(String text, int columns)	Creează un nou obiect JTextField conţinând textul specificat şi având (afişând) lungimea indicată (în coloane)

Elemente pentru editare text

Pentru a putea afisa, respectiv edita o linie de text se pot utiliza elemente de tip ***JTextField***. Acestea permit afisarea si editarea unei linii de text, avand metode precum

setText() – stabileste textul afisat

getText() – citeste textul introdus

Pentru a edita mai multe linii de text se poate folosi clasa **JTextArrea** care permite editarea de text neformatat.

Principalele metode ale componentei JTextField sunt:

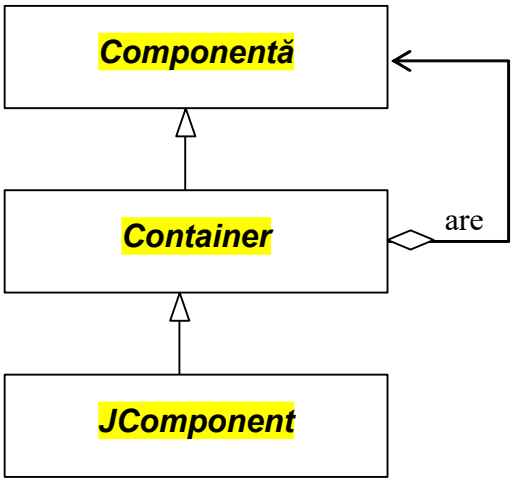
Metoda	***Acţiune***
String getText()	Obţine String-ul conţinut în câmp
void setText(String text)	Stabileşte printr-un String textul ce va fi conţinut de către obiectul JTextField
int getHorizontalAlignment()	Obţine un întreg care indică modul de aliniere relativ la axa x

void setHorizontalAlignment(int alignament)	Stabileşte prin întregul specificat modul de aliniere relativ la axa x
Dimension getPreferredSize()	Obţine dimensiunile (lăţime, lungime printr-un obiect de tip Dimension) preferenţiale
int getColumns()	Obţine numărul de coloane pe care este afişat textul
void setColumns(int columns)	Stabileşte numărul de coloane pe care este afişat textul

Containere

Un obiect care poate conţine componente se numeşte ***container***. Acestea sunt modelate de către clasa abstractă ***java.awt.Container***. Un aspect important, care simplifică lucrul cu intefeţele grafice, constă în faptul că această clasă este, la rândul ei, derivată din clasa *Component*. Prin urmare un *Container* poate conţine un alt *Container*. De asemenea clasa Swing *JComponent* este o subclasă a clasei *Container*. Cum orice *JComponent* poate conţine alte componente putem spune că:

- O **componentă** reprezintă un element distinct al unei interfeţe grafice utilizator, cum ar fi un buton, un câmp textual etc.
- Un **container** reprezintă o componentă a interfeţei grafice utilizator care poate conţine alte componente.



***Relaţiile* Componentă → Container → JComponent**

Cel mai simplu container este ***JPanel***. Un *JPanel* este folosit în general ca o regiune simplă în care sunt aduse şi grupate o colecţie de alte componente. Componentele sunt adăugate unui container prin **metoda *add()*.** De exemplu setul următor de instrucţiuni creează un *JPanel* şi adaugă două butoane:

```
JPanel p = new JPanel;
p.add(new JLabel("Ok"));
p.add(new JLabel("Cancel"));
```

Observatie: distribuirea, localizarea si poziţionarea obiectelor într-o fereastră poate fi controlată folosind un anumit model de gestionare -*layout manager*- care este utilizat prin metoda *setLayout()*.

Dacă se optează pentru poziţionarea absolută (fără nici un factor de relativizare a obiectelor funcţie de mărimea ferestrei de exemplu) atunci metoda *setLayout()* este apelată în felul următor:

setLayout(null);

În acest fel obiectele (butoane, căsuţe de text, grupuri de opţiuni etc.) vor fi afişate întotdeauna la coordonatele stabilite de programator. Spre exemplu poziţionarea şi dimensionarea unui buton de comandă (din clasa *java.swing.JButton* din Swing) se realizează prin metoda:

var_JButton.setBounds(x, y, lăţime, lungime);

Adăugarea unei componente (add) nu este şi condiţia suficientă pentru afişarea acesteia. Modul cum va fi „expusă" şi unde va fi expusă respectiva componentă cade în sarcina unui alt obiect cu care este echipat fiecare container, şi anume *LayoutManager*-ul. Prin urmare un container delegă responsabilitatea poziţionării (dispunerii) şi dimensionării componentelor unui obiect de tip *Layout Manager*, care trebuie să implementeze interfaţa *java.awt.LayoutManager*. Această interfaţă specifică metodele tuturor tipurilor de *layout manager*. Pentru accesarea şi configurarea propriului *layout manager* un container are la dispoziţie metodele *getLayout* şi *setLayout*:

public LayoutManager *getLayout*();
public void *setLayout(LayoutManager manager)*;
Distribuţia Java furnizează mai multe clase care implementează interfaţa *LayoutManager* printre care *FlowLayout, BorderLayout, GridLayout, CardLayout, GridBagLayout, BoxLayout, OverlayLayout.*

Unele se găsesc în package-ul *javax.swing*, iar altele în *java.awt*.

Caracteristicile de bază ale layout manager-ilor standard:

Componenta	**Descriere**
FlowLayout	Dispune sau aşează componentele de la stânga spre dreapta, de sus în jos. Este layout manager-ul implicit(default) pentru *JPanel*
BorderLayout	Afişează până la cinci componente, poziţionate ca “north” – nord, “south” – sud, “east” – est, “west” – vest şi “center” –centru. Este layout manager-ul implicit pentru panoul de componente al *JFrame*.
GridLayout	Aşează componentele într-un grid bidimensional.
CardLayout	Componentele sunt afişate pe rând, dispuse fiind prin suprapunere (parţială).
GridBagLayout	Afişează componentele vertical şi orizontal funcţie de un set re restricţii specifice. Este cel mai complex şi mai flexibil layout manager.
BoxLayout	Afişează componentele fie o singură linie orizontală fie pe o singură coloană verticală. Este layout-managerul implicit pentru containerul *Box* din biblioteca Swing.
OverlayLayout	Afişează componentele aşa încât referinţele de aliniere ale lor indică acelaşi loc. Prin urmare sunt dispuse sub forma unei stive: unele deasupra celorlalte.

Plasarea unor componente GUI într-un cadru implică în Java două declaraţii sau specificaţii:

1. declararea în clasa derivată din *Frame* (sau *JFrame*) a membrilor care desemnează componentele grafice ce vor fi afişate la execuţie;
2. la instanţierea ferestrei respective (sau mai exact a clasei derivate din *Frame*) se va specifica explicit (în cadrul constructorului) instanţierea componentelor grafice şi apoi afişarea lor prin metoda *add() astfel*:

add(InstantaButon);

Componente grafice SWING----- ***Butoane***

Observatie: Una dintre clasele fundamentale din package-ul javax.swing este clasa abstractă *JComponent*, majoritatea componentelor grafice folosite în Java sunt instanţe ale acesteia.

Butoanele sunt elemente de control care, prin apasare, pot genera o actiune. Butoanele deriva din clasa JButton. In costructor primesc textul afisat pe buton. Prin metoda setMnemonic se pot asocia taste de apelare rapida (shortcut).

Pentru a adauga un cuvant de comanda asociat butonului (cuvant ce va fi testat pentru efectuarea actiunii asociate) se foloseste metoda addActionCommand().

Exemplu de definire a unui buton

```
JButton buton = new JButton("BUTON SWING!");
buton.setMnemonic('i');
buton.addActionCommand("butonulSwing");

// adaugarea unui buton pe o suprafata

JPanel panouButon = new JPanel();
panouButon.add(buton);
```

https://docs.oracle.com/javase/tutorFselial/uiswing/layout/layoutlist.html

JCheckBox

Butoanele de marcaj sunt elemente de control care retin o anumita stare. In plus fata de butoanele simple, pentru ele se pot apela metodele:

- **setSelected** (boolen marcat) pentru a stabili marcajul prin program,
- **getSelected**() pentru a testa starea marcajului.

```
JCheckBox cb1 = new JCheckBox("Optiune1");

cb1.setSelected(true);

JPanel checkPanel = new JPanel();

checkPanel.setLayout(new GridLayout(0, 1));

//GridLayout documentare parametrii

Aşează componentele într-un grid bidimensional.

checkPanel.add(cb1);
```

JRadioButton

Butoanele radio sunt elemente de control care retin o anumita stare, la fel cu cele de marcaj. Deosebirea principala consta in faptul ca toate butoanele radio incluse in acelasi grup logic sunt mutual exclusive. Pentru gestionarea grupurilor de butoane radio se va folosi clasa **ButtonGroup**, respectiv metoda **add**() care adauga un buton la grup.

Si pentru butoanele radio se pot apela metodele **setSelected**(boolen marcat) pentru a stabili marcajul prin program, respectiv **getSelected**() pentru a afla starea marcajului.

```
// creare buton radio
JRadioButton primulbuton = new JRadioButton(“BUTON 1”);
primulbuton setActionCommand(“BUTON 1”);

//setActionCommand – pentru a atasa actiuni butonului
primulbuton.setSelected(true);

JRadioButton alt_doilea_buton = new JRadioButton(“BUTON 2”);
alt_doilea_buton.setActionCommand(“BOTON 2”);
…
// definirea unui grup de butoane

ButtonGroup grupbutoane = new ButtonGroup();
gropbutoane.add(primulbuton);
gropbutoane.add(alt_doilea_buton);
…
// adaugarea butoanelor radio pe o suprafata Jpanel ...

JPanel radioPanel = new JPanel();
```

```
radioPanel.setLayout(new GridLayout(0, 1));
radioPanel.add(primulbuton);
radioPanel.add(alt_doilea_buton);
```

JTable

Clasa JTable face parte din Swing si se extinde din Component, utilizeaza cate un model din mai multe interfete de tip „ascultător” , cum ar fi TableModelListener, TableColumnModelListener, ListSelectionListener ...

JTable este de obicei plasat într-un JScrollPane.

Fiecare JTable are trei modele

TableModel, TableColumnModel, si ListSelectionModel.

TableModel este folosit pentru a specifica modul în care datele tabelului se stochează. Datele JTable este gestionat intr-o matrice cu două dimensiuni sau un vector de vectori. TableModel, de asemenea, este utilizat pentru a specifica modul în care datele pot fi editate în tabel.

TableColumnModel este utilizat pentru a gestiona toate coloanele tabelului.

ListSelectionModel permite ca pentru tabel sa se definesca un alt mod de selectie, ca de exemplu intervalul.

Constructorii pentru obiectul JTable sunt:

Tabel 1

JTable Constructors	Description
JTable()	Create an empty table
JTable(int rows, int columns)	Create a table with rows and columns empty cell.

JTable Constructors	Description
JTable(Object[][] data, Object[] heading)	Create a table with data specify in two-dimensional array *data* and column heading *heading*
JTable(TableModel dm)	Create a table with a given TableModel
JTable(TableModel dm, TableColumnModel cm)	Create a table with a given TableModel and TableColumnModel.
JTable(TableModel dm,TableColumnModel cm, ListSelectionModel sm)	Create a table with a given TableModel, TableColumnModel, and ListSelectionModel.
JTable(Vector data, Vector heading)	Create a table with data in vector of Vectors *data* and column headings *headin.*

Aplicatii

```
import javax.swing.*;
class ap1
{
   public static void main(String args[])
   {
      JFrame win=new JFrame("Seminar 3 - 2021");
      win.setSize(300,200);
      win.setLocation(20,20);
      win.setResizable(false);
      win.getContentPane().add(new JLabel("Utilizare swing!"));
      win.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      win.setVisible(true);
   }
}
```

```
import javax.swing.*;
import javax.swing.border.TitledBorder;

public class Ap2 extends JFrame
{
  JComponent comp =new JLabel("Test lable");
public Ap2()
{
   comp.setBorder(new TitledBorder("Titlu border"));
   getContentPane().add(comp);
   setSize(250,250);
   setVisible(true);
}
public static void main(String args[])
   {
   new Ap2();
   }
}
```

Utilizarea diferitelor tipuri de layout

```
import java.awt.*;
import javax.swing.*;
public class Ex1
{
   public static void main(String[] args)
   {
      JFrame f=new JFrame("Dispunere cu FlowLayout");
      JButton b1=new JButton("1");
      JButton b2=new JButton("2");
      JButton b3=new JButton("3");
      JButton b4=new JButton("4");
      JButton b5=new JButton("5");
      JButton b6=new JButton("6");
      Container cp=f.getContentPane();
      cp.setLayout(new FlowLayout());
      cp.add(b1);
      cp.add(b2);
      cp.add(b3);
      cp.add(b4);
      cp.add(b5);
      f.setSize(200,100);
      f.setVisible(true);
   }

}
```

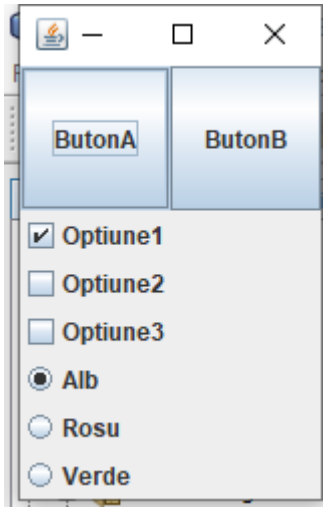
```
import java.awt.*;
import javax.swing.*;

public class Ex2
{
   public static void main(String[] args)
   {
   JFrame f=new JFrame("Dispunere cu BorderLayout");
   Container cp=f.getContentPane();
   cp.setLayout(new BorderLayout());
   cp.add(new JButton("Nord"),BorderLayout.NORTH);
   cp.add(new JButton("Sud"),BorderLayout.SOUTH);
   cp.add(new JButton("Est"),BorderLayout.EAST);
   cp.add(new JButton("Vest"),BorderLayout.WEST);
   cp.add(new JButton("Centru"),BorderLayout.CENTER);
   f.setSize(300,200);
   f.setVisible(true);
   }
}
```

```
import java.awt.*;
import javax.swing.*;

public class ex3
{
```

```
    public static void main(String[] args)
    {
    JFrame f=new JFrame("Dispunere cu GridLayout");
    Container cp=f.getContentPane();
    cp.setLayout(new GridLayout(3,4));
    cp.add(new JButton("1"));
    cp.add(new JButton("2"));
    cp.add(new JButton("3"));
    cp.add(new JButton("4"));
    cp.add(new JButton("5"));
    cp.add(new JButton("6"));
cp.add(new JButton("7"));
    f.setSize(300,200);
    f.setVisible(true);
    }
}
```

```
mport java.awt.*;
import javax.swing.*;

public class ex4
{
    public static void main(String[] args)
    {
    JFrame f=new JFrame("Dispunere cu GridLayout");
    Container cp=f.getContentPane();
    cp.setLayout(new GridLayout(3,2));
    cp.add(new JButton("1"));
    cp.add(new JButton("2"));
    cp.add(new JButton("3"));
    cp.add(new JButton("4"));
    cp.add(new JButton("5"));
    cp.add(new JButton("6"));
    f.setSize(300,200);
    f.setVisible(true);
    cp.add(new JButton("7"));
    cp.add(new JButton("8"));
    if(!cp.isValid())
        cp.validate();
    }
}
```

Componente Swing




```
import javax.swing.*;
import java.awt.*;

public class app1_1 extends JFrame
{
   public static void main(String args[])
   {
   //creare obiecte
   app1_1 app = new app1_1();
   Butoane panouButoane = new Butoane();
   CheckBoxuri panouCheckBoxuri = new CheckBoxuri();
   ButoaneRadio panouButoaneRadio = new ButoaneRadio();
   JPanel panou = new JPanel();
   panou.setLayout(new GridLayout(0,1));
   //aduagare obiecte la panou
   panou.add(panouButoane);
   panou.add(panouCheckBoxuri);
   panou.add(panouButoaneRadio);
   //functii
   app.getContentPane().add(panou);
   app.pack();
  // app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   app.show();
   }
}

class Butoane extends JPanel
{
   public Butoane()
   {
   JButton b1 = new JButton("ButonA");
   JButton b2 = new JButton("ButonB");
   this.setLayout(new GridLayout(1,0));
   add(b1);
   add(b2);
   }
```

```
}

class CheckBoxuri extends JPanel
{
   public CheckBoxuri()
   {
//cream butoane de tip checkBox
   JCheckBox cb1 = new JCheckBox("Optiune1");
   cb1.setSelected(true);//alegem ca primul buton sa fie bifat la pornire
   JCheckBox cb2 = new JCheckBox("Optiune2");
   JCheckBox cb3 = new JCheckBox("Optiune3");

   this.setLayout(new GridLayout(0,1));//dispunere de tip Grid
   add(cb1);//adaugam butonul b1
   add(cb2);//adaugam butonul b2
   add(cb3);//adaugam butonul b3
   }
}

class ButoaneRadio extends JPanel
{
   public ButoaneRadio()
   {
// Creare butoane radio
   JRadioButton butonAlb = new JRadioButton("Alb");
   butonAlb.setActionCommand("Alb");
   butonAlb.setSelected(true);

   JRadioButton butonRosu = new JRadioButton("Rosu");
   butonRosu.setActionCommand("Rosu");

   JRadioButton butonVerde = new JRadioButton("Verde");
   butonVerde.setActionCommand("Verde");
// Adaugarea butoanelor la grup
   ButtonGroup group = new ButtonGroup();
   group.add(butonAlb);
   group.add(butonRosu);
   group.add(butonVerde);
// Adaugarea butoanelor la Layout
   this.setLayout(new GridLayout(0,1));
   add(butonAlb);
   add(butonRosu);
   add(butonVerde);
   }
}
```

```
import javax.swing.*;
import java.awt.*;

class Butoane2 extends JPanel {
public Butoane2() {
```

```
   JButton b1=new JButton("Seria A");
   JButton b2=new JButton("Seria B");
   this.setLayout(new GridLayout(1,0));
   add(b1);
   add(b2);
}
}
class ButoaneRadio2 extends JPanel
{
   public ButoaneRadio2(){
      //creare radio butoane
      JRadioButton butonAlb = new JRadioButton("var1 10-18");
      //butonAlb.setSelected(true);
      JRadioButton butonRosu=new JRadioButton("var2 10-18");
   //gruparea butoanelor
   ButtonGroup group = new ButtonGroup();
   group.add(butonAlb);
   group.add(butonRosu);
   //adaugarea butoanelor
   add(butonAlb);
   add(butonRosu);
}
}

class ListBox1 extends JScrollPane
{
   private JList list;
   public JList getList()
   {
      return list;}
   public ListBox1()
   {
      DefaultListModel listModel=new DefaultListModel();
   listModel.addElement("Randul 1");
   listModel.addElement("Randul 2");
   listModel.addElement("Randul 3");
   list=new JList(listModel);
   list.setVisibleRowCount(3);
   list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
   list.setSelectedIndex(0);
   setViewportView(list);
   }
}


public class app1_2 extends JFrame
{
public static void main(String args[])
{
app1_2 app = new app1_2();
```

```
Butoane panouButoane=new Butoane();
ButoaneRadio panouButoaneRadio=new ButoaneRadio();
ListBox1 panouliste = new ListBox1();
JPanel panou=new JPanel();
panou.add(panouButoane);
panou.add(panouButoaneRadio);
panou.add(panouliste);

app.getContentPane().add(panou);
app.pack();
//app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
app.setVisible(true);

}
}
```