Aplicatii JAVA

#1

JAVA
Introducere in programarea vizuala

Adrian Runceanu
www.runceanu.ro/adrian

Obiectivele disciplinei

Obiectivele generale ale disciplinei:

1. Cunoaşterea noţiunilor privind algoritmii şi proprietăţile lor
2. Reprezentarea algoritmilor prin scheme logice, pseudocod, programe Java
3. Utilizarea platformei integrate de dezvoltare ECLIPSE

Obiectivele specifice:

1. Pentru curs:

- Elemente avansate de programare cu ajutorul limbajului Java
- Elaborarea de programe complexe în Java
- Analiza şi proiectarea algoritmilor cu ajutorul limbajului de programare Java

2. Pentru aplicații:

- Implementarea unor algoritmi într-un mediu de dezvoltare – ECLIPSE
- Implementarea unor algoritmi într-un limbaj de programare utilizat pe scară largă – Java

Câteva precizări

Structura cursului

- ✓**2 ore curs** – titular curs: Lector dr. Adrian Runceanu
- ✓**1 oră laborator** – titular aplicaţii practice: Lector dr. Adrian Runceanu

Câteva precizări

Bibliografia necesară cursului:

1. Tudor Sorin, Vlad Hutanu - **Bazele programarii in Java**, Editura L&S Info-Mat, Bucuresti, 2005.

2. Doina Logofatu – **Algoritmi fundamentali in Java. Aplicatii** – Editura Polirom, Iasi, 2007.

3. Horia Georgescu – **Introducere in universul Java**, Editura Tehnica, Bucuresti, 2002.

Câteva precizări

1. Suport curs - varianta electronică disponibilă pe site-ul:

www.runceanu.ro/adrian

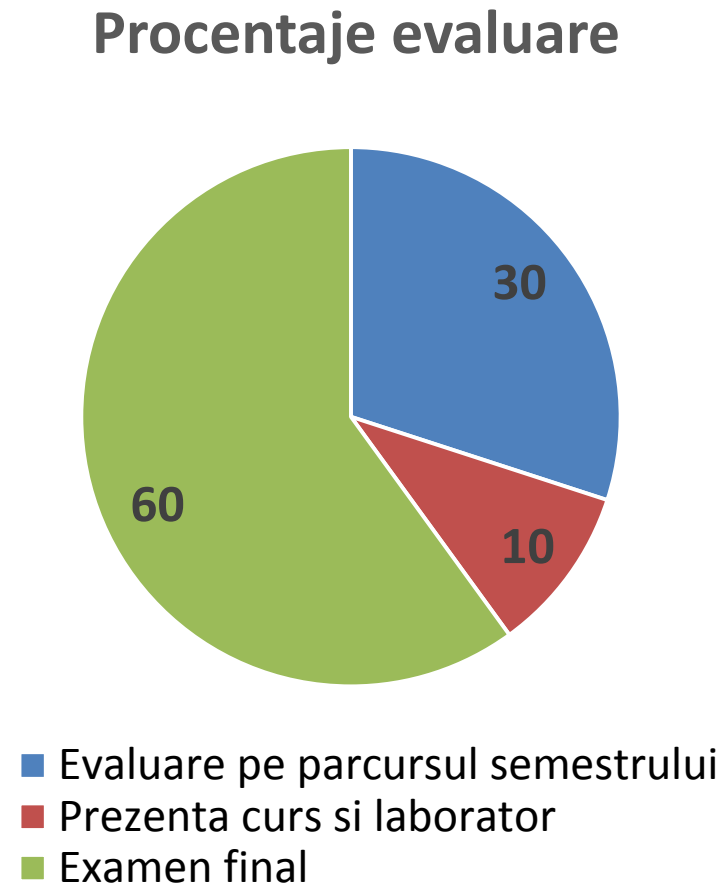
2. Îndrumar de laborator - varianta electronică disponibilă pe site pentru fiecare lucrare de laborator.

Notă: Actualizarea site-ului se face săptămânal.

Câteva precizări

Forme de examinare:

- Examen final = 60%
- Evaluare pe parcursul semestrului a activităţii de laborator = 30%
- Verificare finală lucrări de laborator = 10%



Conținutul cursului

1. Mediul de dezvoltare aplicații orientate-obiect **ECLIPSE**

2. Limbajul de programare **JAVA**

Curs 1

Introducere in programarea vizuala

Introducere in programarea vizuala

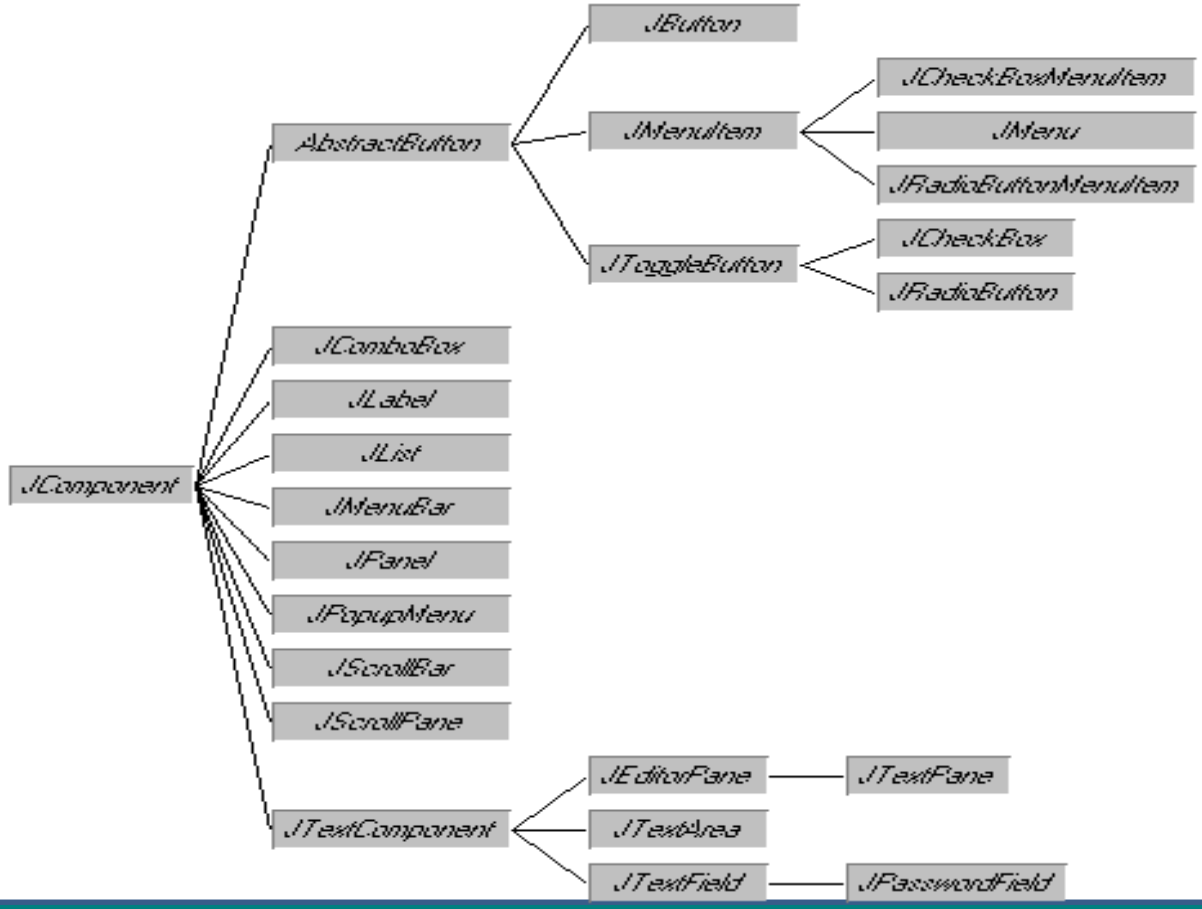
1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.1. Pachetele AWT si Swing

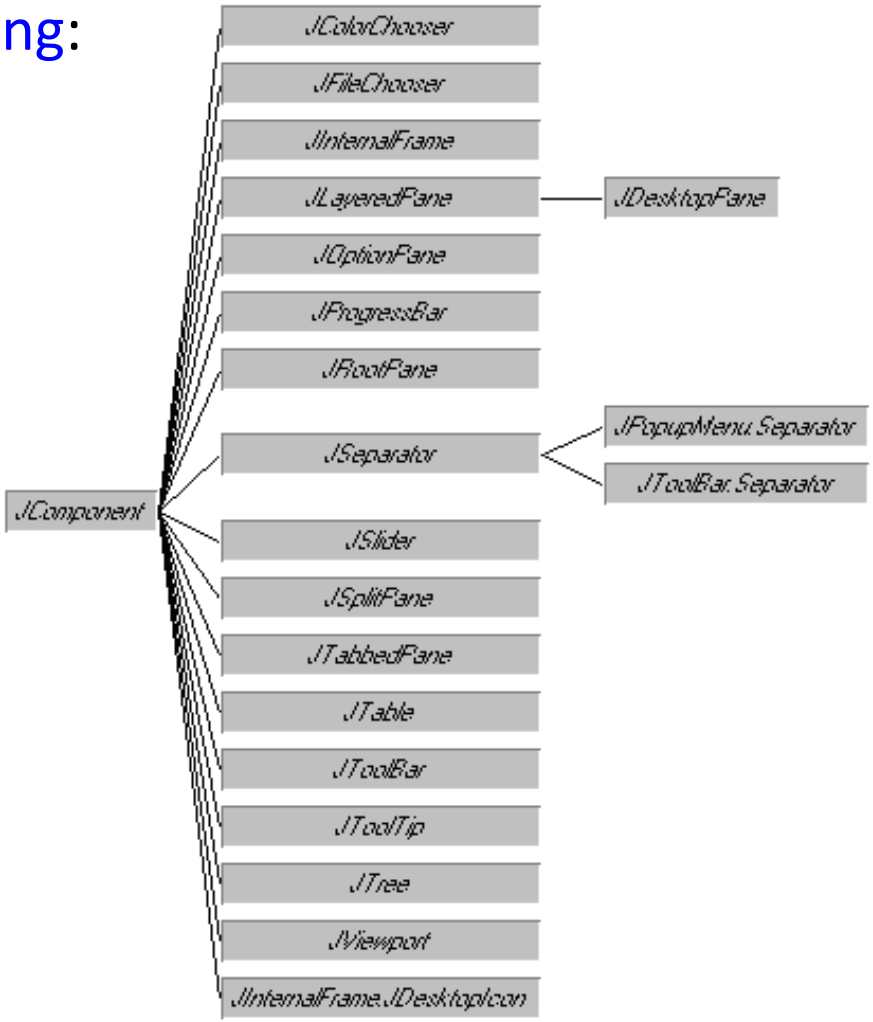
- Aplicatiile realizate pana acum au avut interfata in mod text.
- Astfel, ecranul era privit ca o matrice de caractere.
- Pentru simplificarea utilizarii aplicatiilor complexe se prefera crearea unei interfete in mod grafic.
- In acest caz ecranul este considerat o matrice de puncte (pixeli) de diverse culori (de obicei fiecare pixel are asociate trei componente de culoare R-red, G-green, B-blue).
- O astfel de interfata poarta denumirea de "GUI" – Graphical User Interface.

- O interfata GUI este formata din ferestre (portiuni dreptungiulare pe ecran).
- Ferestrele unei aplicatii contin "widgets" (elemente grafice de control) care permit interactiunea cu utilizatorul.
- Interactiunea cu utilizatorul se realizeaza prin mouse si tastatura.
- O interfata grafica se creaza de obicei cu sprijinul sistemului de operare (printr-o componenta numita server grafic).
- Limbajul Java pune la dispozitia programatorului doua biblioteci pentru realizarea unei interfete grafice: **java.awt** si **javax.swing**.

- Pachetele awt si swing contin clase pentru gestionarea completa a unei interfete.
- In figura sunt expuse clasele corespondente cu cele din awt (*Abstract Window Toolkit):*



- In plus fata de pachetul standard awt, pachetul swing adauga noi clase care permit imbunatatirea interfetei realizate.
- In figura urmatoare sunt prezentate clasele noi introduse de catre swing:



Introducere in programarea vizuala

1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.2. Ferestre

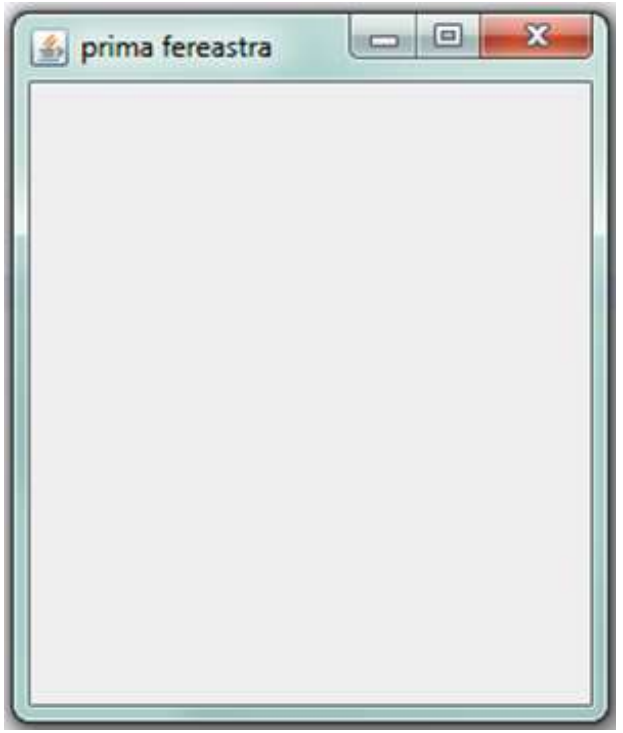
- Oricarei aplicatii grafice îi corespunde o fereastra principala (de tip FRAME) si una sau mai multe ferestre aditionale.

- In pachetul swing exista trei clase pentru gestionarea ferestrelor:
  1. **JFrame**
  2. **JWindow**
  3. **JDialog**

1.2. Ferestre

1. Clasa JFrame permite *crearea unei ferestre de aplicatie*.

- Fereastra are:
  - o bara de titlu
  - o margine
  - butoane de:
    - minimizare
    - maximizare
    - inchidere (butoane "system")



1.2. Ferestre

2. Clasa JWindow permite crearea unei ferestre *fara bara de titlu, meniu, butoane sistem,* etc.

1.2. Ferestre

3. Clasa JDialog permite *crearea de ferestre de dialog*.

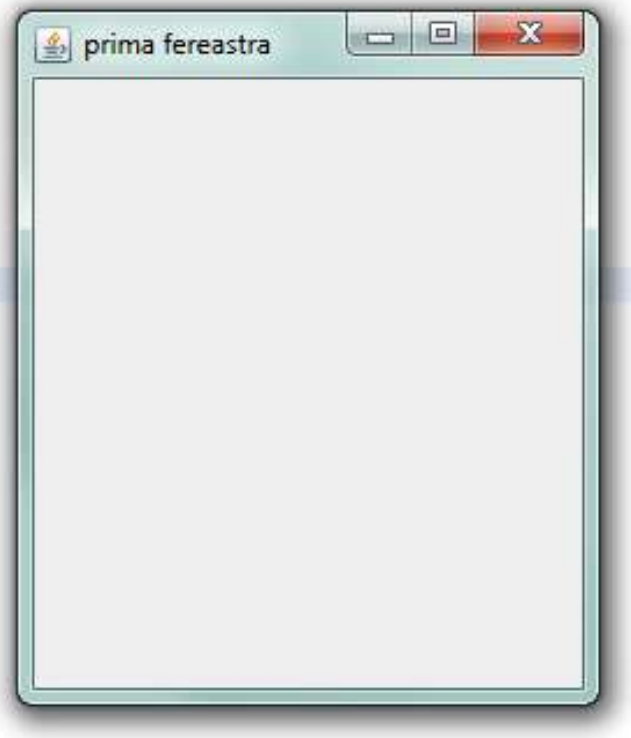
- Ferestrele de dialog sunt dependente de ferestrele parinte de tip Frame.

- O fereastra de dialog poate fi:
  - modala (blocheaza aplicatia pana la inchiderea dialogului)
  - sau nemodala (nu blocheaza)

Exemplu de fereastra:

Pv.java

```
1  package curs16;
2  import javax.swing.*;
3  public class Pv {
4
5      public static void main(String[] args) {
6
7          JFrame fer=new JFrame("prima fereastra");
8          fer.setSize(200,300);
9          fer.setLocation(300,400);
10         fer.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11         fer.setVisible(true);
12         }
13
14 }
15
```



Clasa JFrame

Clasa JFrame contine cateva metode:

- JFrame() – constructor (daca o fereastra este creata printr-un astfel de constructor ea apare fara titlu).
- JFrame(String titlu) – constructor (constuieste o fereastra care afiseaza un anumit titlu)
- void setSize(int width, int height) – stabileste latimea si inaltimea ferrestrei

Clasa JFrame

Clasa JFrame contine cateva metode:

- void setLocation(int x, int y) – stabileste pozitia unde va fi afisat coltul din stanga sus al ferestrei (si implicit pozitia ferestrei), in raport cu coltul din stanga sus al ecranului.
  - Parametrul x precizeaza distanta pe orizontala a coltului ferestrei, iar y distanta pe verticala a acestuia (ambele sunt date in pixeli).
- void setResizabile(boolean ac) – daca parametrul este false nu se pot modifica dimensiunile ferestrei

Clasa JFrame

Clasa JFrame contine cateva metode:

- void setDefaultCloseOperation(int x) – stabileste ce se intampla atunci cand se inchide fereastra (s-a plasat acel buton x).
  - Parametrii sunt constante de tip intreg ale clasei. Cea mai importanta constanta este **EXIT_ON_CLOSE** si prin ea, se cere inchiderea ferestrei si inchiderea executiei programului.

- setVisible(boolean x) – stabileste daca fereastra este vizibila (apare pe ecran) sau nu (desi exista, nu este afisata).

Se poate construi o clasa, numita Fereastra, al carei constructor sa returneze obiectul cu toate datele din exemplul anterior:

```
import javax.swing.*;
    class Fereastra extends JFrame{
    Fereastra(String Nume, int lat, int inalt, int dreapta,
    int stanga){
            super(Nume);
            setSize(lat,inalt);
            setLocation(dreapta, stanga);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setVisible(true);
    }
}
```

```
public class Pv{
    public static void main(String args[ ]){
        Fereastra f=new Fereastra("fereastra", 100, 200,
    25, 100);
    }
}
```

Pv.java

```
1  package curs16;
2  import javax.swing.*;
3
4  class Fereastra extends JFrame{
5
6      Fereastra(String Nume, int lat, int inalt, int dreapta, int stanga)
7      {
8          super(Nume);
9          setSize(lat,inalt);
10         setLocation(dreapta, stanga);
11         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12         setVisible(true);
13     }
14 }
15
16 public class Pv{
17     public static void main(String args[ ]){
18         Fereastra f=new Fereastra("fereastra", 100, 200, 25, 100);
19     }
20 }
21
```

Introducere in programarea vizuala

1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container

- Pentru ca unei ferestre (obiect de **JFrame**) sa i se poata atasa alte componente:
  - Butoane
  - Liste, etc. (obiecte de alte tipuri),
- este necesar ca aceasta sa contina o referinta catre o structura speciala, care la randul ei va retina referinte catre obiectele (componente) care sunt atasate ferestrei.
- Structura care retine referintele catre obiectele care se afla pe fereastra este un obiect al clasei **Container**.

1.3. Mecanismul prin care se ataseaza componentele ferestrei. Clasa Container

Ierarhia claselor din care a rezultat clasa **JFrame**:

Object
- **Component**
  - **Container**
    - **Window**
      - **Java.awt.Frame**
        - **Javax.swing.JFrame**

1.3. Mecanismul prin care se ataseaza componentele ferestrei. Clasa Container

- Accesul la container-ul unei ferestre se face utilizand o metoda a clasei JFrame,numita **getContentPane( ).**

- **Container getContentPane( )** - returneaza o referinta catre container-ul ferestrei.

1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container

Pornind de la aceasta referinta, putem realiza doua lucruri:

1. Putem adauga ferestrei componentele dorite.

2. Putem spune cum sa fie aranjate in fereastra componentele adaugate.

1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container

1. Putem adauga ferestrei componentele dorite.

Pentru aceasta se foloseste metoda add() a clasei Container:

Component add(**Component comp**) - adauga o componenta ferestrei.

De retinut:

Toate componentele sunt derivate din clasa Component.

Prin urmare, parametrul **comp** poate retine referinte catre componente de orice tip.

1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container

2. Putem spune cum sa fie aranjate in fereastra componentele adaugate.

De retinut:

- Mecanismul din Java care rezolva aceasta problema presupune existenta unor asa numiti *gestionari de pozitionare*.
- *Gestionarii de pozitionare* sunt obiecte ale unor clase specifice.
- Ei *aranjeaza automat componentele unui container*.

1.3. Mecanismul prin care se ataseaza componentele ferestrei. Clasa Container

Pentru a atasa unui container un gestionar de pozitionare, se utilizeaza metoda clasei **Container** numita **setlayout ()**:

~ **void setLayout(LayoutManager gest)** – ataseaza unui container un gestionar de pozitionare.

- LayoutManager – este o interfata.
- Toti gestionarii de pozitionare pe care ii vom studia într-un paragraf separat au rezultat ca urmare a implementarii acestei interfete.
- Aceasta inseamna ca metoda poate fi utilizata pentru atasarea oricarui gestionar de pozitionare.

1.3. Mecanismul prin care se ataseaza componentele ferestrei. Clasa Container

- Pentru inceput, vom folosi gestionarul de pozitionare **FlowLayout**, care are un constructor fara parametri.

- Pe scurt acesta *aseaza componentele in fereastra, pe linie, una dupa alta*.

- In cazul in care o linie s-a umplut, se trece la linia urmatoare .

1.3. Mecanismul prin care se ataseaza componentele ferestrei. Clasa Container

- Clasa Container se gaseste in pachetul java.awt.
- Exemplu de program care afiseaza doua butoane (fisierul *Pv1.java*).
- Apasarea butoanelor nu are nici un efect.

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame{
    public Fer(String titlu)
    {
        super(titlu);
        setSize(200,100);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container x = getContentPane();
        x.setLayout(new FlowLayout());
        JButton A = new JButton("Buton 1"); x.add(A);
        JButton B = new JButton("Buton 2"); x.add(B);
        setVisible(true);
    }
}
```

```
public class Pv1{
    public static void main(String
    args[ ]){
        Fer fp = new Fer("fereastra
    cu doua butoane");
    }
}
```

Observatie:

Butoanele atasate sunt componente de tip **JButton**, care au un constructor de tip **JButton (String s).**

Sirul s va aparea pe suprafata butonului.

Pv1.java

```
1  package curs16;
2  import java.awt.*;
3  import javax.swing.*;
4
5  class Fer extends JFrame{
6          public Fer(String titlu)
7              {
8                  super(titlu);
9                  setSize(200,100);
10                 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11                 Container x = getContentPane();
12                 x.setLayout(new FlowLayout());
13                 JButton A = new JButton("Buton 1"); x.add(A);
14                 JButton B = new JButton("Buton 2"); x.add(B);
15                 setVisible(true);
16             }
17         }
18
19         public class Pv1{
20             public static void main(String args[ ]){
21                 Fer fp = new Fer("fereastra cu doua butoane");
22         }
23 }
24
```

Introducere in programarea vizuala

1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”

- Pana in prezent stim sa construim o fereastra, stim sa-i atasam unul sau mai multe butoane, dar nu stim sa facem astfel incat la ”apasarea” butonului (click pe suprafata lui) sa aiba loc o anumita actiune.

1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”

- In Java exista o interfata numita **ActionListener**, ”ascultatorul” de evenimente de tip **ActionEvent**.
- Un exemplu de eveniment de tip **ActionEvent** este “apasarea”unui buton.
- Interfata contine antetul unei singure metode:

actionPerformed (ActionEvent e)

1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”

- Pentru ca o componenta sa poata raspunde la un eveniment de tipul **ActionEvent** trebuie sa implementeze clasa **ActionListener.**
- Aceasta inseamna ca:

1. Clasa care include componenta (fereastra) sa contina clauza

Implements ActionListener;

2. Sa fie implementata metoda **actionPerformed( ).**

- Aceasta metoda se va executa automat atunci cand este apasat butonul.
- Prin urmare, implementarea ei va scrie codul necesar actiunii dorite.

1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”

- **ActionEvent** este o clasa care contine metoda:
  - **String getActioncommand ( ) –** returneaza sirul de caractere asociat componentei care a transmis evenimentul.
  - Metoda poate fi utilizata pentru a depista componenta care a transmis evenimentul.

Exemplu:

Am extins programul anterior. Cand se apasa un buton, in fereastra **CMD** (Console din Eclipse) va aparea sirul retinut de butonul apasat.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Fer extends JFrame implements ActionListener{
    public Fer(String titlu)
    {
        super(titlu);
        setSize(200,100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container x = getContentPane();
        x.setLayout(new FlowLayout());
        JButton A = new JButton("Buton 1"); x.add(A);
        JButton B = new JButton("Buton 2"); x.add(B);
        A.addActionListener(this);
        B.addActionListener(this);
        setVisible(true);
    }
```

```
public void actionPerformed(ActionEvent e)
{
    if(e.getActionCommand().compareTo("Buton 1") == 0)
        System.out.println("Ai apasat Buton 1");
    else
        System.out.println("Ai apasat Buton 2");
    }
}
public class Pv2{
    public static void main(String args[ ]){
    Fer fp = new Fer("Doua butoane");
    }
}
```

Pv2.java

```
1  package curs16;
2  import java.awt.*;
3  import javax.swing.*;
4  import java.awt.event.*;
5  class Fer extends JFrame implements ActionListener{
6      public Fer(String titlu)
7      {
8          super(titlu);
9          setSize(200,100);
10         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11         Container x = getContentPane();
12         x.setLayout(new FlowLayout());
13         JButton A = new JButton("Buton 1"); x.add(A);
14         JButton B = new JButton("Buton 2"); x.add(B);
15         A.addActionListener(this);
16         B.addActionListener(this);
17         setVisible(true);
18     }
19
20     public void actionPerformed(ActionEvent e)
21     {
22         if(e.getActionCommand().compareTo("Buton 1") == 0)
23             System.out.println("Ai apasat Buton 1");
24         else
25             System.out.println("Ai apasat Buton 2");
26         }
27     }
28
29 public class Pv2 {
30     public static void main(String[] args) {
31         Fer fp = new Fer("Doua butoane");
32     }
33 }
```

```
32     }
33 }
34
```

Problems @ Javadoc Declaration Console

```
Pv2 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (14 dec. 2014, 10:51:02)
Ai apasat Buton 1
Ai apasat Buton 2
Ai apasat Buton 1
Ai apasat Buton 2
```

Introducere in programarea vizuala

1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.5. Clasa JComponent

- *Prin componentă vom înţelege un obiect care are o reprezentare grafică.*
- Exemple de componente:
  - butoane
  - liste
  - edit-uri
  - etichete
- Dacă o componentă este obiect, înseamnă că o componentă rezultă în urma instanţierii unei clase.

1.5. Clasa JComponent

- Fiecare tip de componentă pe care o vom studia rezultă în urma instanţierii unei clase specifice ei.

- De exemplu:
  - un buton rezultă în urma instanţierii clasei **JButon**
  - o etichetă rezultă în urma instanţierii clasei **JLabel**
  - un edit rezultă în urma instanţierii clasei **JtextField**, ş.a.m.d.

1.5. Clasa JComponent

Cateva metode ale clasei **JComponent**:

- **void setBackground(Color c) –** metoda stabileste culoarea de fond a componentei; parametrul este de tip **Color.**
- **void setForeground (Color c) –** seteaza culoarea caracterelor (in cazul in care componenta contine un text).

1.5. Clasa JComponent

- **Clasa Color** contine anumite constante care indica culoarea si mai multe metode prin care se poate stabili o culoare.

- Exemple de constante de culoare: **black, red, white, yellow,** etc.
- De asemenea,clasa contine constructorul **Color(float r, float g, float b)** prin care se poate forma o culoare si sistemul **RGB (Red, Green, Blue)**, sistem studiat la cursul de Grafica asistata de calculator (anul I).

1.5. Clasa JComponent

- **setFont(Font f)** seteaza font-ul cu care se scrie, stilul său şi mărimea.

Parametrul este un obiect al clasei **Font.**

- Clasa **Font** are, printre altele, constructorul:

Font(String nume, int stil, int marime)

unde:

- **nume –** este numele fontului;
- **still –** stilul. Valorile uzuale sunt:
  - **Font.ITALIC (italic)**
  - **Font.BOLD (bold)**
  - **Font.PLAIN(classic)**
  - Se pot folosi si combinatii,cum ar fi pentru italic si bold: **Font.ITALIC + Font.BOLD**.
- **marime –** marimea fontului

1.5. Clasa JComponent

Exemplu:

- buton cu fond rosu,”**Apasa”** contine text scris cu verde, se utilizeaza font-ul **Arial,** marimea **20** si este scris italic+bold.
- Dimensiunea butonului este stabilita automat, in functie de marimea textului pe care il contine.

```
JButton  A=new JButton(“Apasa”);
A.setBackground (Color.RED);
A.setFont(new Font(“Arial”, Font.ITALIC+Font.BOLD, 20));
A.setForeground(Color.GREEN);
```

1.5. Clasa JComponent

- **void setToolTipText(String text); -** metoda setează un şir de caractere care va fi afişat atunci când cursorul mouse-ului staţionează asupra componentei.
- Şirul are rolul unui mesaj lămuritor despre funcţia respectivei componente.

Exemplu:

```
JButton  B = new JButton(“Button 2”);
B.setToolTipText(“Eu sunt butonul 2”);
```

1.5. Clasa JComponent

- **void setEnabled(Boolean v)** – Face ca o componentă să fie activată (v reţine **true**) sau nu (v reţine **false**).
- O componentă dezactivată nu mai răspunde comenzilor şi are un aspect specific prin care utilizatorul este anunţat de faptul că aceasta este dezactivată.

- **void setVisible(Boolean v)** – dacă parametrul reţine **true**, componenta este vizibilă, altfel ea este invizibilă.

Introducere in programarea vizuala

1. Introducere in programarea vizuala
  1.1. Pachetele AWT si Swing
  1.2. Ferestre
  1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
  1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
  1.5. Clasa JComponent
    1.5.1. Poziţionarea componentelor
    1.5.2. Gestionarul de poziţionare FlowLayout
    1.5.3. Gestionarul de pozitionare GridLayout
    1.5.4. Gestionarul de pozitionare BorderLayout

1.5.1. Poziţionarea componentelor

- Pentru a aşeza componentele în poziţia dorită, putem să utilizăm:
  - *poziţionarea absolută* (dar este nerecomandată pentru că programul trebuie să ruleze corect pe o diversitate de platforme)
  - sau unul dintre gestionarii de poziţionare existenţi.

- De retinut: programele Java trebuie să funcţioneze pe orice platfomă.

1.5.1. Poziţionarea componentelor

1. Poziţionarea absolută

- Pentru a utiliza poziţionarea absolută a componentelor trebuie să lucrăm în absenţa unui gestionar de poziţionare, iar coordonatele sunt date în pixeli.
- Pentru aceasta trebuie să cunoaştem următoarele:

a) Secvenţa de mai jos se utilizează pentru a lucra în absenţa unui gestionar de poziţionare.

```
Container x = getContentPane();
x.setLayout(null);
```

1.5.1. Poziţionarea componentelor

b) În poziţionarea absolută componentele trebuie dimensionate şi poziţionate, altfel nu sunt vizibile.

Pentru dimensionarea componentelor se utilizează următoarele metode ale clasei **JComponent**:

setBounds(int x, int y, int lat, int inal);

- metoda poziţionează şi dimensionează componenta.
- Parametrii **x** şi **y**, dau poziţia componentei în raportată la colţul din stânga sus al componentei care o găzduieşte, iar **lat** şi **lung** o dimensionează.

1.5.1. Poziţionarea componentelor

- Metoda anterioara poate fi înlocuită cu următoarele două metode:

➢ **setLocation(int x, int y)** – metodă care are rolul de poziţionare;

➢ **setSize (int x, int y)** – metodă care are rolul de dimensionare.

Exemplu: utilizăm poziţionarea absolută şi aşezăm un buton în fereastră:

```
Container x=get.ContentPane();
x.setLayout(null);
JButton A=new JButton (“Exemplu”);
A.setBounds(10, 10, 100, 40);
```

Introducere in programarea vizuala

1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.5.2. Gestionarul de poziţionare FlowLayout

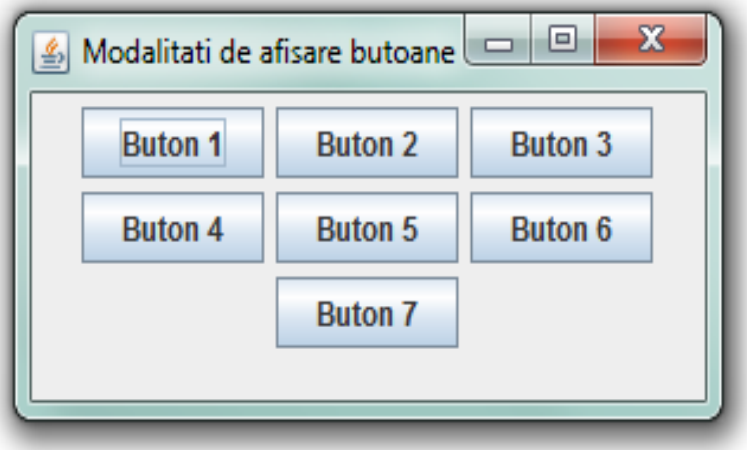
- Componentele sunt afişate pe linii, în ordinea în care au fost declarate.
- Pe fiecare linie ele sunt afişate de la stânga la dreapta (atâtea câte încap).
- Urmatoarea fereastra are 7 butoane:

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame{
    public Fer(String titlu)  {
        super(titlu);
        setSize(300, 150);
        Container x = getContentPane();
        x.setLayout(new FlowLayout());
        JButton A = new JButton("Buton 1"); x.add(A);
        JButton B = new JButton("Buton 2"); x.add(B);
        JButton C = new JButton("Buton 3"); x.add(C);
        JButton D = new JButton("Buton 4"); x.add(D);
        JButton E = new JButton("Buton 5"); x.add(E);
        JButton F = new JButton("Buton 6"); x.add(F);
        JButton G = new JButton("Buton 7"); x.add(G);
        setVisible(true);
    }
}
```

```
public class Pv4
{
  public static void main(String args[ ])
  {
     Fer fp=new Fer("Modalitati de afisare
    butoane");
  }
}
```

Pv3.java

```
1  package curs16;
2  import java.awt.*;
3  import javax.swing.*;
4  class Fere extends JFrame
5  {
6       public Fere(String titlu)
7       {
8            super(titlu);
9            setSize(300, 150);
10           Container x = getContentPane();
11           x.setLayout(new FlowLayout());
12           JButton A = new JButton("Buton 1"); x.add(A);
13           JButton B = new JButton("Buton 2"); x.add(B);
14           JButton C = new JButton("Buton 3"); x.add(C);
15           JButton D = new JButton("Buton 4"); x.add(D);
16           JButton E = new JButton("Buton 5"); x.add(E);
17           JButton F = new JButton("Buton 6"); x.add(F);
18           JButton G = new JButton("Buton 7"); x.add(G);
19           setVisible(true);
20      }
21 }
22
23 public class Pv3 {
24
25     public static void main(String[] args) {
26          Fere fp=new Fere("Modalitati de afisare butoane");
27     }
28 }
```



1.5.2. Gestionarul de poziţionare FlowLayout

- Clasa **FlowLayout** conţine constante de aliniere pe linie, din care mai importante sunt:
  - **CENTER,** aliniere în centru, opţiune implicită,
  - **LEFT**, la stânga
  - şi **RIGHT**, la dreapta.

- Clasa **FlowLayout** este înzestrată cu **3** constructori:

a) FlowLayout( ) – distanţa între rânduri este de 5 unităţi, distanţa pe orizontală între componente este de **5** unităţi şi componentele sunt aliniate pe linie la centru (**CENTER**)

1.5.2. Gestionarul de poziţionare FlowLayout

b) FlowLayout(int aliniere) – se cere explicit ca alinierea să fie într-un anumit fel (**CENTER, RIGHT, LEFT,** acestea sunt constante ale clasei FlowLayout).

c) FlowLayout(int aliniere, int dist_oriz, int dist_vert) – se specifica si distanta pe orizontala intre componente si distanta pe verticala dintre ele.

Exemple de utilizare a constructorilor:

1.5.2. Gestionarul de poziţionare FlowLayout

- Aspecte care trebuie retinute:

1. *Dimensionarea componentelor si pozitionarea lor este facuta automat de catre gestionar.*

2. *Metodele utilizate in pozitionare absoluta pentru pozitionarea si dimensionarea componentelor (setBounds(), setLocation(), setSize()) desi sunt acceptate la compilare, nu au efect.*

1.5.2. Gestionarul de poziţionare FlowLayout

3. Exista, totusi o metoda care dimensioneaza componentele si este acceptata de gestionarul FlowLayout.

Ea apartine clasei JComponent:

setPreferredSize(Dimension dim);

- clasa Dimension are constructorul Dimension(int lat, int inalt) prin care se specifica latimea, respectiv inaltimea componentei.

1.5.2. Gestionarul de poziţionare FlowLayout

Exemplu:

- Prin utilizarea gestionarului FlowLayout se adauga unei ferestre, doua butoane:
  - primul de dimensiune stabilita,
  - iar al doilea de dimensiune implicita:

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame
{
     public Fer(String titlu)   {
            super(titlu);
            setSize(300, 150);
            Container x = getContentPane();
            x.setLayout(new FlowLayout());
            JButton A = new JButton("Buton 1");
            A.setPreferredSize(new Dimension(100,100));
            x.add(A);
            JButton B = new JButton("Buton 2"); x.add(B);
            setVisible(true);
    }
}
public class Pv6 {
  public static void main(String args[ ])  {
              Fer fp=new Fer("Butoane de dimensiuni diferite");
  }
}
```

Pv5.java

```
1  package curs16;
2  import java.awt.*;
3  import javax.swing.*;
4  class Ferest extends JFrame
5  {
6       public Ferest(String titlu)   {
7            super(titlu);
8            setSize(300, 150);
9            Container x = getContentPane();
10           x.setLayout(new FlowLayout());
11           JButton A = new JButton("Buton 1");
12           A.setPreferredSize(new Dimension(100,100));
13           x.add(A);
14           JButton B = new JButton("Buton 2"); x.add(B);
15           setVisible(true);
16      }
17 }
18
19 public class Pv5 {
20
21     public static void main(String[] args) {
22         Ferest fp=new Ferest("Butoane de dimensiuni diferite");
23     }
24 }
25
```

Introducere in programarea vizuala

1. Introducere in programarea vizuala
  1.1. Pachetele AWT si Swing
  1.2. Ferestre
  1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
  1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
  1.5. Clasa JComponent
    1.5.1. Poziţionarea componentelor
    1.5.2. Gestionarul de poziţionare FlowLayout
    1.5.3. Gestionarul de pozitionare GridLayout
    1.5.4. Gestionarul de pozitionare BorderLayout

1.5.3. Gestionarul de pozitionare GridLayout

- Clasa GridLayout – aranjeaza componentele intr-o alta logica.

- Ideea de bază este aceea ca se împarte suprafata ferestrei în mai multe dreptunghiuri de suprafaţă egală şi în fiecare dreptunghi astfel obţinut se aşează o componentă care este de cele mai multe ori extinsa ca suprafaţă, astfel încât să ocupe întreaga suprafaţă a dreptunghiului care îi revine.

1.5.3. Gestionarul de pozitionare GridLayout

- Constructorii aceste clase:

a) GridLayout() - daca avem n componente care trebuie asezate, suprafata ferestrei este impartita intr-o singura linie si n coloane.

Fiecare componenta este asezata intr-un dreptunghi, iar componentele sunt fara spatiu intre ele.

1.5.3. Gestionarul de pozitionare GridLayout

b) GridLayout(int nr_linii, int nr_coloane) - suprafata ferestrei este impartita in nr_linii * nr_coloane dreptunghiuri. Fiecare dreptunghi retine o componenta.

GridLayout(3,3)

1.5.3. Gestionarul de pozitionare GridLayout

c) GridLayout(int nr_linii, inr nr_coloane , int sp_oriz, sp_vert) - suprafata ferestrei este impartita in nr_linii * nr_coloane dreptunghiuri numai ca se trec si spatiile pe orizontala si pe verticala intre dreptunghiuri.

GridLayout(3,3,5,8)

Introducere in programarea vizuala

1. Introducere in programarea vizuala
   1.1. Pachetele AWT si Swing
   1.2. Ferestre
   1.3. Mecanismul prin care se ataseaza componentele ferestrei.Clasa Container
   1.4. Un mecanism prin care butoanele raspund evenimentului de “apasare”
   1.5. Clasa JComponent
      1.5.1. Poziţionarea componentelor
      1.5.2. Gestionarul de poziţionare FlowLayout
      1.5.3. Gestionarul de pozitionare GridLayout
      1.5.4. Gestionarul de pozitionare BorderLayout

1.5.4. Gestionarul de pozitionare BorderLayout

Clasa BorderLayout – imparte suprafata ferestrei in 5 parti:

- nord(NORTH),
- sud(SOUTH),
- est(EAST),
- vest(WEST)
- si centru (CENTER).

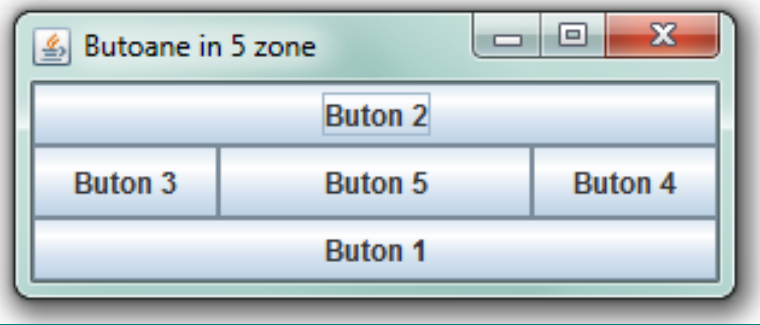
In fiecare parte se poate aseza o componenta.

Exemplu:

```
import java.awt.*;
import javax.swing.*;
class Fer extends JFrame
{
      public Fer(String titlu)  {
          super(titlu);
          setSize(300, 100);
          Container x = getContentPane();
          x.setLayout(new BorderLayout());
          JButton A = new JButton("Buton 1"); x.add(A, BorderLayout.SOUTH);
          JButton B = new JButton("Buton 2"); x.add(B, BorderLayout.NORTH);
          JButton C = new JButton("Buton 3"); x.add(C, BorderLayout.WEST);
          JButton D = new JButton("Buton 4"); x.add(D, BorderLayout.EAST);
          JButton E = new JButton("Buton 5"); x.add(E, BorderLayout.CENTER);
          setVisible(true);
    }
}
public class Pv6 {
  public static void main(String args[ ])  {
                Fer fp=new Fer("Butoane in 5 zone");
  }
}
```

Pv6.java

```
1  package curs16;
2  import java.awt.*;
3  import javax.swing.*;
4  class Fereastr extends JFrame
5  {
6       public Fereastr(String titlu)  {
7            super(titlu);
8            setSize(300, 100);
9            Container x = getContentPane();
10           x.setLayout(new BorderLayout());
11           JButton A = new JButton("Buton 1"); x.add(A, BorderLayout.SOUTH);
12           JButton B = new JButton("Buton 2"); x.add(B, BorderLayout.NORTH);
13           JButton C = new JButton("Buton 3"); x.add(C, BorderLayout.WEST);
14           JButton D = new JButton("Buton 4"); x.add(D, BorderLayout.EAST);
15           JButton E = new JButton("Buton 5"); x.add(E, BorderLayout.CENTER);
16           setVisible(true);
17      }
18 }
19
20 public class Pv6 {
21
22     public static void main(String[] args) {
23         Fereastr fp=new Fereastr("Butoane in 5 zone");
24     }
25 }
26
```



Întrebări?