Lucrare de laborator nr. 13
Programare grafică în Java (I)

1. Scopul lucrării

În această lucrare se realizează câteva aplicaţii grafice în care se folosesc componente grafice de tipul: *JButton, JLabel, JTextField, JRadioButton.* Se arată modul de aranjare al componentelor grafice în fereastra grafică, și se studiază următoarele gestionare de layout:

BorderLayout
FlowLayout
GridLayout

Se prezintă de asemenea modul de tratare al evenimentelor din cadrul aplicației grafice.

2. Breviar teroretic

În realizarea aplicaţiilor grafice sunt implicate, în principal, pachetele de clase: **java.awt** și **javax.swing**.

În general, o aplicaţie grafică constă din definirea unei ferestre, fereastră în care programatorul aranjează componentele grafice.

Componentele grafice pot avea o reprezentare vizibilă sau pot să nu aibă o reprezentare vizibilă.

Exemple de componente grafice cu reprezentare vizibilă:

a. Butoane: fac parte din clasa **JButton**
b. Etichete: din clasa **Jlabel**
c. Câmp de editare text de o linie : clasa **JTextField**
d. Casete de validare : din clasa **JCheckBox**
e. Butoane radio: **c**lasa **JRadioButton**
f. Meniuri: clasele **JMenuBar**, **JMenuItem**.
g. Câmpuri de editare: clasa **JTextField**
h. Arie de text : clasa **JTextArea**

Dintre componentele care nu au o reprezentare vizibilă ca exemplu tipic avem componentele de tip **container**. Astfel, clasa **JPanel** este un container intermediar, fără o reprezentare vizibilă.

Rolul lui este de a aduna in acelaşi grup mai multe componente grafice, care pot fi manevrate ca un ansamblu.

Programarea grafică este o programare orientată pe evenimente. La realizarea aplicaţiei grafice cooperează programatorul cu mediul de execuţie Java.

În programarea tradiţională operatorul este controlat de program în ce priveşte introducerea datelor (le introduce atunci când programul îi cere explicit aceasta). În programarea orientată pe evenimente, operatorul introduce datele atunci când doreşte (se generează un eveniment şi programul trebuie să trateze acest eveniment). În cazul programării orientate pe evenimente programul trebuie să conţină procedurile de tratare a evenimentelor. Aceste proceduri nu sunt apelate de către programul scris de programator, ci de către un program dispatcher.

În programarea grafică orientată pe evenimente, programatorul realizeaza urmatoarele lucruri:
- plasează componentele grafice în fereastră
- desemnează care sunt sursele de evenimente
- scrie codul pentru tratarea evenimentului respectiv (procedura de tratare a evenimentului).

Astfel, o sursă de eveniment poate să fie un buton, componentă de tipul **JButton**, şi evenimentul este declanşat atunci când utilizatorul face click cu mouse-ul pe acel buton.

O componentă grafică este desemnată de programator ca o sursă de evenimente, prin ataşarea de acea componentă a unui obiect de ascultare a evenimentului.

Ataşarea obiectului de ascultare a evenimentului se face printr-o metodă specifică. In cazul unui buton de tip **JButton**, ataşarea unui obiect de ascultare se face prin metoda **addActionListener( ).** Butonul respectiv devine astfel sursă de eveniment.
Exemplu:

```
JButton jb = new JButton (”Calcul”);
// desemnam pe jb ca sursa de evenimente.
// cream obiectul de ascultare:
AscultaButoane m = new AscultaButoane ( );
//Il atasam obiectului jb:
jb.addActionListener (m);
```

Observaţie: Acelaşi obiect de ascultare poate fi ataşat la mai multe componente grafice. În rutina de tratare a evenimentului, în acest caz, mai întâi se va stabili care a fost sursa evenimentului.

Nu orice componentă grafică poate fi sursă de evenimente. Un obiect de tipul **JLabel** nu poate să fie sursă de eveniment. De asemenea, nu toate componentele grafice care au potenţialul de a fi surse de evenimente, sunt desemnate de programator ca fiind surse de evenimente.

Programatorul trebuie să definească clasa din care se crează obiectul de ascultare (obiect ce se ataşează sursei de evenimente).

În această clasă de ascultare se defineşte metoda de tratare a evenimentului.

Mediul de execuţie Java este cel care detectează evenimentele şi apelează în mod automat, la detectarea evenimentului respectiv, metoda corespunzătoare de tratare a evenimentului (metodă scrisă de programator în clasa de ascultare). În plus, numele metodei de tratare a evenimentului este impus de către Java, dar programatorul este cel care îi dă codul.

Astfel, pentru tratarea evenimentului, click cu mouse-ul pe un buton, numele impus pentru metoda de tratare este **actionPerformed() .**

Exemplu:

Tratarea apasarii butonului "Exit", se face în clasa de ascultare a butonului, astfel:

```
class AscultaButonExit implements ActionListener
{
        public void actionPerformed (ActionEvent ev)
        {
                System.exit (0); // ceea ce scrie programatorul
        }
}
```

Clasa AscultaButonExit, deoarece implementează interfaţa ActionListener, trebuie să implementeze definiţiile tuturor metodelor descrise de aceasta interfaţă. Interfaţa ActionListener declară doar o singură metodă: metoda *actionPerformed(),* de aceea clasa AscultaButonExit o va implementa în mod obligatoriu. În acest exemplu nu s-a folosit parametrul metodei *actionPerformed(),* dar se foloseşte atunci când acelaşi obiect de ascultare este ataşat la mai multe componente.

Fereastra aplicaţiei extinde clasa de bibliotecă **JFrame**.

De la aceasta moşteneşte forma (bară de titlu, cele 3 butoane: minimizare, maximizare şi închidere) şi comportamentul de fereastră (mărirea, micşorarea dimensiunilor ferestrei, mutarea ferestrei).

Componentele grafice, în cazul utilizării ferestrei de tip **JFrame** (definită în pachetul **javax.swing**), nu se adaugă direct pe suprafaţa ferestrei, ci într-un container extras din fereastră cu ajutorul metodei **getContentPane( )**. Acesta se extrage astfel:

Container c=getContentPane();

Adăugarea componentelor în container se face cu metoda add() .

Din considerente de portabilitate, nu se recomandă poziţionarea componentelor grafice la coordonate absolute.

Poziţionarea componentelor se face cu ajutorul unor clase denumite **layout manager** (clase ce fac gestionarea modului de poziţionare).

Astfel, containerul extras din **JFrame** cu metoda getContentPane() are ca şi layout, mod de aranjare implicit al componentelor, layout-ul de tip **BorderLayout**.

Acest layout plasează componentele pe 5 direcţii (cele patru puncte cardinale si centrul): **North**, **South**, **East**, **West**, **Center**.

Mai sunt si alte layout-uri posibile pe lângă **BorderLayout.** De exemplu:

● **FlowLayout** – în care componentele grafice sunt aranjate în linie, de la stânga la dreapta. Obiectele de tip **JPanel**, au implicit acest layout.

● **GridLayout –** în care componentele grafice sunt aranjate într-o matrice, specificată prin numărul de linii şi numărul de coloane.

Layout-ul implicit al unui container poate fi modificat la un nou layout, folosind metoda **setLayout( )** care are ca parametru un obiect instanţiat din noul layout.

Exemplu:

Pentru a aranja într-o componentă JPanel, patru butoane JButton, pe verticală, procedăm astfel:

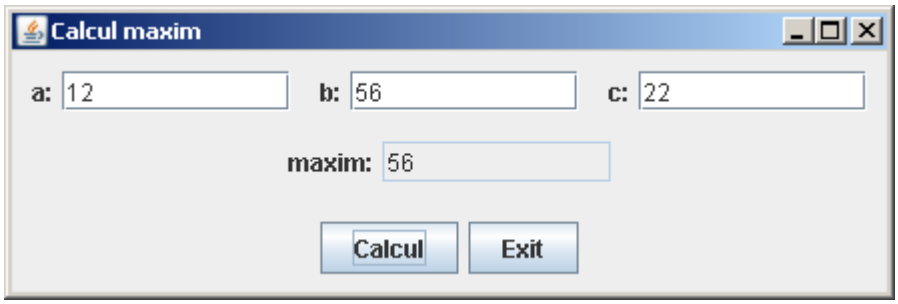
```
JButton jb1=new JButton(”A”);
JButton jb2=new JButton(”B”);
JButton jb3=new JButton(”C”);
JButton jb4=new JButton(”D”);
//Containerul intermediar:
JPanel jp=new JPanel;
//Ii schimbam layoutul implicit:
jp.setLayout(new GridLayout(4,1));
jp.add(jb1);
jp.add(jb2);
jp.add(jb3);
```

jp.add(jb4);

3. Probleme rezolvate

Problema 1

Se va realiza interfaţa grafică prezentată mai jos. Ea conţine ca şi componente grafice patru componente JTextField, patru componente JLabel şi două butoane JButton. În fiecare din primele trei câmpuri de editare se va tasta câte un număr întreg. La apăsarea butonului **Calcul,** în al patrulea JTextField se va afişa maximul dintre cele trei numere.




```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class CalculMaxim
{
 public static void main(String args[ ])
 {
  Fereastra f=new Fereastra( );
  f.pack();//dimensiuni imagine optima
  f.setLocationRelativeTo(null);//centrare. Dar dupa pack()
  f.setVisible(true);
 }
}
class Fereastra extends JFrame
{
 private JButton jbCalcul,jbExit;
 private JTextField jtf1, jtf2, jtf3,jtf4;
 public Fereastra( )
 {
```

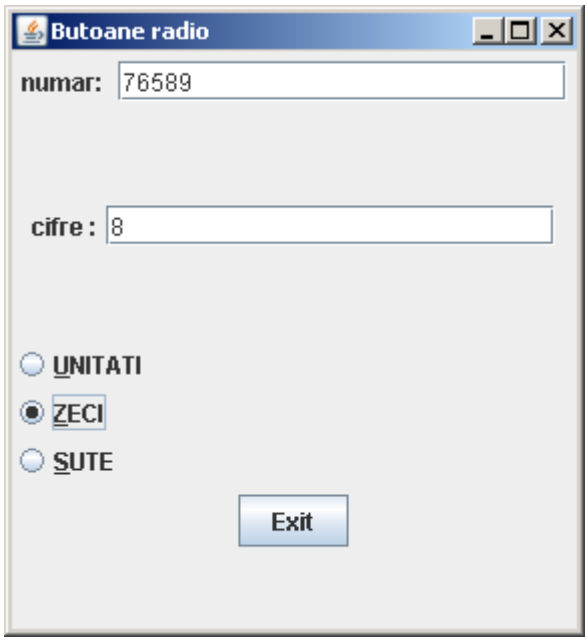
```
 this.setTitle("Calcul maxim");
 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 JLabel jl1=new JLabel("a:");
 jtf1=new JTextField(10);
 JPanel jp1=new JPanel( );
 jp1.add(jl1);
 jp1.add(jtf1);
 JLabel jl2=new JLabel("b:");
 jtf2=new JTextField(10);
 JPanel jp2=new JPanel( );
 jp2.add(jl2);
  jp2.add(jtf2);
 JLabel jl3=new JLabel("c:");
 jtf3=new JTextField(10);
 JPanel jp3=new JPanel( );
 jp3.add(jl3); jp3.add(jtf3);
 JPanel jpA=new JPanel();
 jpA.add(jp1);jpA.add(jp2);jpA.add(jp3);
 JLabel jl4=new JLabel("maxim:");
 jtf4=new JTextField(10);
 jtf4.setEditable(false);
 JPanel jpB=new JPanel();
 jpB.add(jl4);
 jpB.add(jtf4);
 AscultaButoane ab=new AscultaButoane( );
 jbCalcul=new JButton("Calcul");
 jbCalcul.addActionListener(ab);
 jbExit=new JButton("Exit");
 jbExit.addActionListener(ab);
 JPanel jpC=new JPanel( );
 jpC.add(jbCalcul);
 jpC.add(jbExit);
 JPanel jp=new JPanel( );
 jp.setLayout(new GridLayout(3,1));
 jp.add(jpA); jp.add(jpB);jp.add(jpC);
 Container cFinal=this.getContentPane( );
 cFinal.add(jp,"South");
}// end constructor
/*
```

În continuare în clasa Fereastra definim ca şi clasă interioară, clasa de ascultare a butoanelor: clasa AscultăButoane. O clasă interioară a unei alte clase, are acces în mod direct la toate variabilele de instanţă ale clasei exterioare(şi la cele private).
*/

```
// clasa interioară:
class AscultaButoane implements ActionListener
{
 public void actionPerformed(ActionEvent ev)
 {
  Object sursa=ev.getSource();
  if (sursa==jbExit) System.exit(0);
  else if (sursa==jbCalcul){
   String s1=jtf1.getText( );
   int a=Integer.parseInt(s1);
   int b=Integer.parseInt(jtf2.getText( ));
   int c=Integer.parseInt(jtf3.getText( ));
   int max=a;
   if (b>max)max=b;
   if(c>max)max=c;
   jtf4.setText(""+max); }
 }//actionPerformed
}
}
```

Problema 2

Se va realiza interfaţa grafică prezentată mai jos. Ea conţine ca şi componente grafice două componente JTextField, cinci componente JLabel, trei butone radio JRadioButton şi un buton JButton. În câmpul de editare numar se va tasta un număr întreg. Din butoanele radio se selectează cifra ce se va afişa în câmpul de editare cifre.




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

class ButoaneRadio
{
 public static void main(String args[])
 {
  Fereastra f=new Fereastra();
  f.pack();//dimensiuni imagine optima
  f.setLocationRelativeTo(null);
  f.setVisible(true);
 }//main
}

class Fereastra extends JFrame
{
 private JButton  jbExit;
 private JTextField jtf1, jtf2;
 private JRadioButton jrbCifraU,jrbCifraZ,jrbCifraS;
```

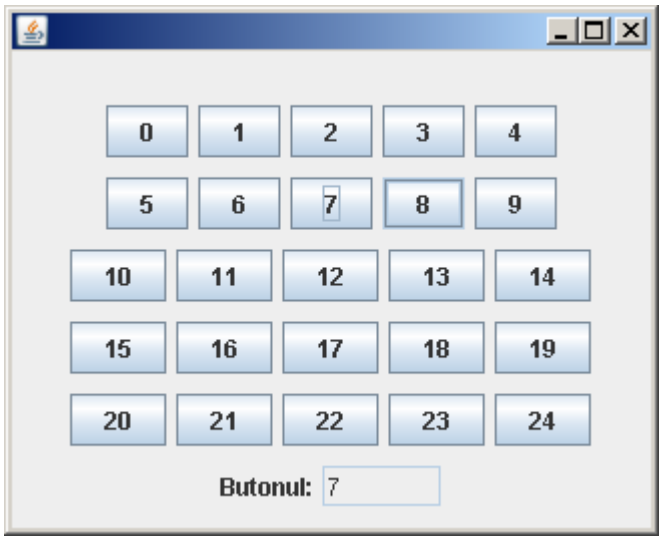
```
public Fereastra( )
{
 this.setTitle("Butoane radio");
 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 JLabel jl1=new JLabel("numar: ");
 Jtf1=new JTextField(20);
 JPanel jp1=new JPanel( );
 jp1.add(jl1);
 jp1.add(jtf1);
 JLabel jl2=new JLabel("cifre :");
 jtf2=new JTextField(20);
 JPanel jp2=new JPanel( );
 jp2.add(jl2);
 jp2.add(jtf2);
 jrbCifraU = new JRadioButton("UNITATI");
 jrbCifraU.setMnemonic(KeyEvent.VK_U);
 jrbCifraZ = new JRadioButton("ZECI");
 jrbCifraZ.setMnemonic(KeyEvent.VK_Z);
 jrbCifraS = new JRadioButton("SUTE");
 jrbCifraS.setMnemonic(KeyEvent.VK_S);
 //Grupam butoanele radio
 // (pt. a pastra starea butonului selectat din grup)
 ButtonGroup group = new ButtonGroup();
 group.add(jrbCifraU);
 group.add(jrbCifraZ);
 group.add(jrbCifraS);
 JPanel jpRadio=new JPanel();
 jpRadio.setLayout(new GridLayout(3,1));
 jpRadio.add(jrbCifraU);
 jpRadio.add(jrbCifraZ);
 jpRadio.add(jrbCifraS);
 AscultaButoaneRadio abr=new AscultaButoaneRadio( );
 jrbCifraU.addActionListener(abr);
 jrbCifraZ.addActionListener(abr);
 jrbCifraS.addActionListener(abr);
 AscultaButoane ab=new AscultaButoane( );
 jbExit=new JButton("Exit");
 jbExit.addActionListener(ab);
 JPanel jp3=new JPanel( );
 jp3.add(jbExit);
```

```
 JPanel jp=new JPanel( );
 jp.setLayout(new GridLayout(4,1));
 jp.add(jp1);  jp.add(jp2);  jp. add(jpRadio);  jp.add(jp3);
 Container cFinal=this.getContentPane( );
 cFinal.add(jp,"South");
}// end constructor

// clasa interioara:
class AscultaButoaneRadio implements ActionListener
{
 public void actionPerformed(ActionEvent ev)
 {
  String s1=jtf1.getText();
  if(!s1.equals("")){//trebuie ca in jtf1 sa fie scris ceva!
     int nr=Integer.parseInt(s1);
     if(jrbCifraU.isSelected()){
     int u=nr%10;
     jtf2.setText(u+"");}
  else if(jrbCifraZ.isSelected()){
     int z=(nr/10)%10;
     jtf2.setText(z+"");}
  else if(jrbCifraS.isSelected()){
     int s=(nr/100)%10;
     jtf2.setText(s+"");}
  }
 }//actionPerformed
}

class AscultaButoane implements ActionListener
{
 public void actionPerformed(ActionEvent ev)
 {
  System.exit(0);
 }//actionPerformed
}
}
```

Problema 3

Se va realiza interfaţa grafică prezentată mai jos. Ea conţine ca şi componente grafice un număr nB=25 de butoane JButton, dispuse

tabelar (5 linii și 5 coloane). Butoanele au ca nume scris pe fiecare , numărul de ordine din tabel (0, 1, …24). Interfaţa conţine şi un câmp de editare JTextField şi o etichetă JLabel. La apăsarea unui buton oarecare din tabel, se va afişa numărul acestuia în câmpul de editare.




```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class JocButoane
{
 public static void main(String args[])
 {
  int nB=25;
  Fereastra f=new Fereastra(nB);
  f.pack();//dimensiuni imagine optima
  f.setLocationRelativeTo(null);//centrare. Dar dupa pack()
  f.setVisible(true);
 }
}
class Fereastra extends JFrame
{
 private int nB;//nr. total butoane
 private JButton jb[];
 private JTextField jtf;
 public Fereastra(int nB)
```

```
{
 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 this.nB=nB;
 jb=new JButton[nB];
 int i,j;
 int N=(int)Math.sqrt(nB);//dimensiune matrice
 //Fiecare linie de butoane o introducem intr-un JPanel, deci
 // vom avea un vector de N JPaneluri:
 //Astfel se pastreaza dimensiunea la resize fereastra !
 AscultaButoane ab=new AscultaButoane();
 JPanel jp[]=new JPanel[N];
 for(i=0;i<N;i++){
 //linia i ce contine N butoane
 jp[i]=new JPanel();//containerul intermediar al liniei i:
 for(j=0;j<N;j++){
  int nrButonCrt=N*i+j;
  String numeButonCrt=nrButonCrt+"";
  jb[nrButonCrt]=new JButton(numeButonCrt);
  jb[nrButonCrt].addActionListener(ab);
  jp[i].add(jb[nrButonCrt]);
 }
 }
 JPanel jpFinal=new JPanel();
 jpFinal.setLayout(new GridLayout(N+1,1));
 //N linii cu butone si pe ultima linie un JTextField:
 for(i=0;i<N;i++)jpFinal.add(jp[i]);
 jtf=new JTextField(5);
 jtf.setEditable(false);
 JLabel jl=new JLabel("Butonul:");
 JPanel jtfPanel=new JPanel();
 jtfPanel.add(jl); jtfPanel.add(jtf);
 jpFinal.add(jtfPanel);
 Container c=this.getContentPane();
 c.add(jpFinal,"South");
}//end constructor

class AscultaButoane implements ActionListener
{
 public void actionPerformed(ActionEvent ev)
 {
```

```
   Object sursa=ev.getSource();
   //polling printre butoane ca sa vedem care este:
   int i;
   for(i=0;i<nB;i++)
    if((JButton)sursa==jb[i]){
     jtf.setText(i+"");
     break;
    }
  }
 }
}
```

3. Probleme propuse

Problema 1

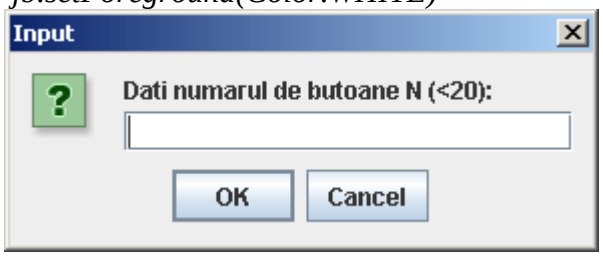
Să se scrie o aplicaţie în care se afişează o fereastră ce conţine patru componente grafice: JTextField pentru introducerea elementelor unui vector de numere intregi (elementele sunt separate prin spaţii), JTextField pentru afişarea maximului din vector şi două componente JButton. Atunci când este apăsat primul buton, se va afişa maximul din vectorul introdus. Când se apasă cel de-al doilea, se iese din program.

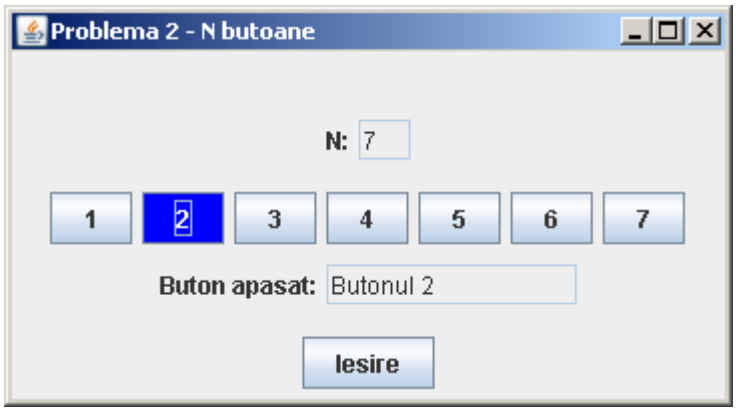
Problema 2

Se citeşte de la tastatură un număr natural N. Apoi se afişează în linie N butoane denumite 1, 2,… La apăsarea unui buton, se va scrie în câmpul de editare numele butonului apăsat. În interfaţă se va afişa de asemenea şi numărul N introdus de la tastatură.

Pentru a schimba culoarea butonului apăsat se vor folosi următoarele instrucţiuni:

```
jb.setBackground(Color.BLUE);
jb.setForeground(Color.WHITE)
```





Problema 3

Se va realiza interfaţa grafică prezentată mai jos. Ea conţine ca şi componente grafice două componente JTextField, două componente JLabel, două butoane radio JRadioButton, şi două butoane JButton. În primul JTextField se tastează un număr întreg. Dacă este selectat butonul radio DIVIZORI, la apăsarea butonului CALCULE , în al doilea câmp de editare se vor afişa toţi divizorii numărului. Dacă este selectat butonul radio CIFRE, la apăsarea butonului CALCULE , în al doilea câmp de editare se vor afişa toate cifrele numărului, separate printr-un spaţiu.