C10

Grafica 2D în Java

- AWT şi Swing
- JFrame
- JPanel
- grafică 2D

Obiective

După parcurgerea acestui curs ar trebuie să puteţi:

- ☐ înţelege principiile de lucru cu geometria 2D din Java;
- ☐ crea un cod grafic sigur utilizand Swing;
- ☐ înţelege şi modifica cod JDeveopler pentru geometria 2D;
- ☐ înţelege si genera grafica pe baza primitivelor geometrice 2D: linie, dreptunghi, cerc, text, poligon si culoare.

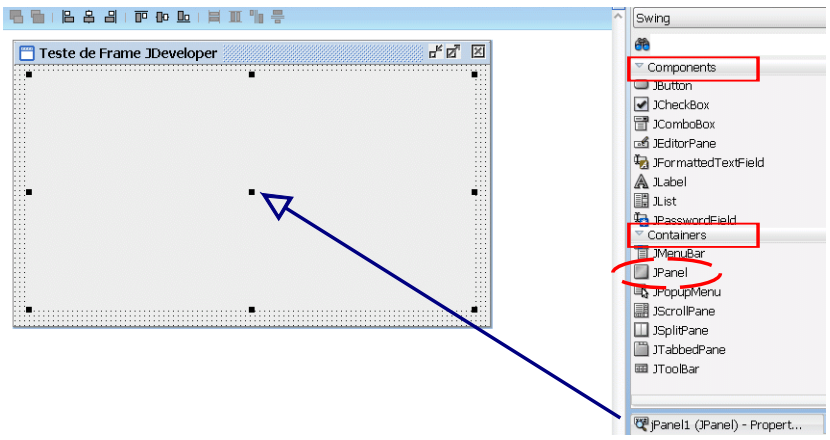
AWT şi Swing

Java 1.0 pornea cu o bibliotecă de clase, scrise de Sun şi numită Abstract Windows Toolkit (AWT) ce oferea funcţiile de bază pentru realizarea de interfeţe grafice. AWT delega gestionarea entităţilor grafice sistemului de operare nativ (Windows, Macintos, UNIX ...). Pe baza acestui principu al delegării aplicaţiile simple erau portabile, însă aplicaţiile complexe puneau probleme serioase ca urmare a dependenţei lor de elementele de interfaţă native. Netscape a început dezvoltarea unei noi biblioteci unde elementele de interfaţă se afişau direct pe o fereastră vidă, unica interacţiune cu sistemul de operare nativ apărând la asamblarea acestor ferestre într-una singură. Sun a lucrat pe acest principiu cu Netscape la dezvoltarea unei biblioteci noi, numite Swing, care avea:

- un număr mare de elemente de interfaţă grafice;
- mai puţine dependenţă de platforma pe care se rula;
- comportament uniform pe toate platormele.

Azi, Swing, face parte din Java Foundation Class (JFC) biblioteca oficială Java pentru crearea de interfeţe grafice portabile împreună cu AWT şi Java2D. Swing s-a implementat în Java şi respectă arhitectura software Model-View-Controller (MVC) în care este separată reprezentarea datelor de interacţiune cu utilizatorul. Modelul (model) conţine datele aplicaţiei şi codul de prelucrare specific respectivelor date. Vederea (view) reprezintă o modalitate de vizualizare a datelor specifică unei anumite cerinţe sau utilizator. Controlorul (controller) este o interfaţă software dintre intrare şi model ce converteşte transferul de date în comenzi specifice modelului. Idea de bază din spatele aceste arhitecturi este separarea şi reutilizarea codului.

Iniţial distribuite ca şi o bibliotecă separată, azi sunt inclusă în bibliotecile din Java Standard Edition.



Clasele Swing şi componentele software sunt stocate în pachetul Java `javax.swing`. Includerea acestor componente într-o aplicaţie necesită importul componentelor în cod, iată câteva exemple:

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

Conceptul de componentă, în contextul interfeţelor grafice, defineşte un grup de clase ce colaborează pentru implementarea unei interfeţe. Una dintre clase se ocupă exclusiv de partea vizuală dar prin

delegare transferă o parte din sarcini altor clase din grup. Partea de vizualizare şi cea de prelucrare sunt suficient de decuplate încât să poată fi substituite fără a afecta restul de cod. Mediul JDeveloper afişează aceste componente de interfaţă Swing sub denumirea de Components, câteva exeple sunt: JButton, JCheckBox, JLabel, JTextField, JTextArea.

O altă secţiune specifică interfeţelor grafice Swing sunt containerele (Containers). Acestea sunt clase ce conţine componente Swing. Rolul lor este de afişare şi poziţionare a componentelor în baza unui plan de aranjare.

Frame-uri

Frame-ul este fereastră principală a unei aplicaţii având titlu şi margine. Clasa pentru crearea unui Frame din Swing se nuimeşte JFrame. Majoritatea componetelor Swing încept cu litera "J". Dacă se omite litera "J" este posibil ca aplicaţia să lucreze, dar utilizând o componentă din AWT. Combinarea AWT-ului cu Swing-ul duce la inconsistenţe vizuale la rularea aplicaţiei.

Implicit frame-ul are mărimea, în pixeli, de 0 x 0. Metoda `setSize(lăţime, înălţime)` se foloseşte pentru a modifica această dimensiune implicită inutilă. O aplicaţie poate avea una sau mai multe frame-uri. Utilizatorul are obligaţia să definească ce se petrece atunci când se închide o fereastră din clasa JFrame prin:

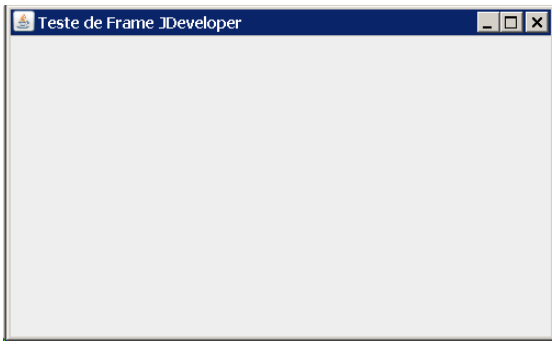
```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Prin codul de mai sus închiderea frerestrei duce şi la ieşirea din program. O altă situaţie este aceea când aplicaţi are mai mult ferestre şi noi închidem una singură, cu setarea de mai sus, aplicaţie va fi închisă, deşi poate nu era cazul.

Implicit frame-urile sunt invizibile, pentru afişarea lor trebuie apelată metoda `setVisible()` cu parametrul `true`.

Codul creat de JDeveloper pentru o aplicaţie Swing/AWT cu Frame este:

```
import java.awt.Dimension;
import javax.swing.JFrame;

public class TestFrame extends JFrame {
  public TestFrame() {
      try {
          jbInit();
      } catch (Exception e) {
          e.printStackTrace();
      }
  }

private void jbInit() throws Exception {
  this.getContentPane().setLayout( null );
  this.setSize( new Dimension(500, 300) );
  this.setTitle( "Teste de Frame
JDeveloper" );
}

public static void main(String[] args) {
  TestFrame frame = new TestFrame();
  //setare comportament inchidere frame

frame.setDefaultCloseOperation(JFrame.EXIT
_ON_CLOSE);
  //afisare  frame.
  frame.setVisible(true);
  //afisare pozitie colt stanga-sus

System.out.println(frame.getLocation());
  //afisare lungime, inaltime [width,
height]

System.out.println(frame.getSize());
    }
}
```




```
java.awt.Point[x=0,y=0]
java.awt.Dimension[width=500,height=300]
```

Afişarea pe Frame

Afişarea informaţiilor se poate face direct pe Frame dar procedura nu se consideră a fi o deprindere de programator sănătoasă. Frame-urile sunt proiectate pentru a putea stoca alte componente de interfaţă cum sunt ferestrele cu text, listele, liste combinate cu text, etc. În condiţii normale desenarea se face pe o componentă numită panou (`JPanel`) care se adugă Frame-ului cu metoda `add()`. Codul generat în JDeveloper ca urmare a adăugării unui JPanel la un frame este îngrosat:

```
import java.awt.Dimension;
import java.awt.Rectangle;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class TestFrame extends JFrame {
 private JPanel jPanel1 = new JPanel();

 public TestFrame() {
   try {
       jbInit();
   } catch (Exception e) {
       e.printStackTrace();
   }
 }

 private void jbInit() throws Exception {
   this.getContentPane().setLayout( null );
   this.setSize( new Dimension(500, 300) );
   this.setTitle( "Teste de Frame JDeveloper" );
   jPanel1.setBounds(new Rectangle(15, 10, 465, 245));
   this.getContentPane().add(jPanel1, null);
 }

 public static void main(String[] args) {
   TestFrame frame = new TestFrame();
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }
}
```

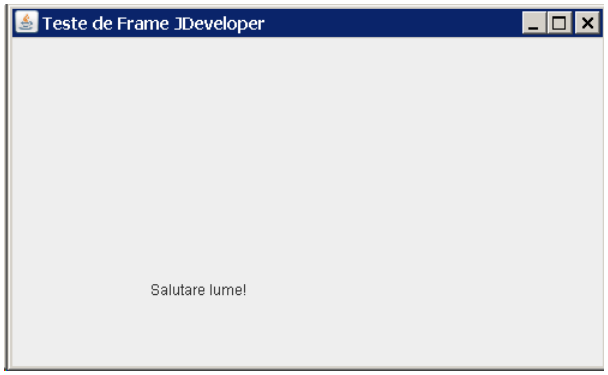
Structura unui JFrame este complexă, fiind organiză pe 4 panouri: root, layerd,content şi glass. Din punctul de vedere al creatorului de interfeţe esenţial este panoul de conţinut (content pane) deoarece aici se adaugă componentele de afişat. În codul de mai sus metoda `add()` se foloseste pentru adăugarea `JPanel`-ului cu numele `JPanel1` (nume dat implicit de JDeveloper) la `JFrame`. JPanel-ul asigură o suprafaţă de desenare ce poate fi container la rândul ei daca este cazul.

Afişarea pe JFrame cu JPanel

JPanel-ul poate conţine componente sau grafica Java 2D. Particularizarea conţinutului afişat de JPanel se face prin extinderea clasei JPanel unde obligatoriu se redefineşte metoda paintComponent() eventual se adaugă noi metode. paintComponent() este moştenită din superclasa JComponent şi are ca parametru un obiect Graphics. Tot procesul de desenare din Java trebuie să treacă prin obiectul Graphics. Acesta conţine metode pentru desenarea de forme (linie, cerc, etc.), imagini şi text. De fiecare dată când conţinutul lui JFrame trebuie redesenat, metodele paintComponent() ale tuturor componentelor sunt apelate automat. Acţiunile ce pot declanşa acest proces sunt modificarea dimensiunilor ferestrei, inchiderea unei ferestre ce acoperea fereastra curenta, total sau parţial, la prima vizualizare a ferestrei pentru afişarea conţinutului acesteia. Metoda paintComponent() nu se apelează direct, dacă se doreşte reafişarea conţinutului unei ferestre se apelează metoda repaint(). Ecranul grafic se măsoara in pixeli, cooronatele de (0,0) sunt în colţul din stânga-sus. Codul care urmează realizează afişarea în regim grafic a unul text pe JPanel.

```
import java.awt.Graphics;
import javax.swing.JPanel;
class TestJPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("Salutare lume!",100,200);
    }
}

import java.awt.Dimension;
import java.awt.Rectangle;
import javax.swing.JFrame;

public class TestFrame extends JFrame {
    private TestJPanel jPanel1 = new TestJPanel();

    public TestFrame() {
        try {
            jbInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        this.getContentPane().setLayout( null );
        this.setSize( new Dimension(500, 300) );
```

```
        this.setTitle( "Teste de Frame JDeveloper" );
        jPanel1.setBounds(new Rectangle(15, 10, 465, 245));
        this.getContentPane().add(jPanel1, null);
    }

    public static void main(String[] args) {
        TestFrame frame = new TestFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



Clasa `TestJPanel` extinde clasa `JPanel`, metoda `paintComponent()` care are propriile ei reguli de afişare, din acest motiv trebuie ca superclasa să-şi facă treaba pentru aceasta se apelează înainte de orice altceva metoda `super.paintComponent()`.

Clasa `Graphics` are metode pentru trasarea de linii, dreptunghiuri, elipse, etc. însă aceste operaţii sunt destul de limitate. De exemplu, nu se poate varia grosimea liniilor sau realiza de rotaţii. O nouă biblioteca numită Java 2D a fost creată în acest scop. Pentru a desena un obiect 2D pe baza acestei biblioteci trebuie obţinut obiectul `Graphics2D` prin forţare de tip (cast) din obiectul `Graphics`. Codul tipic în acest scop este:

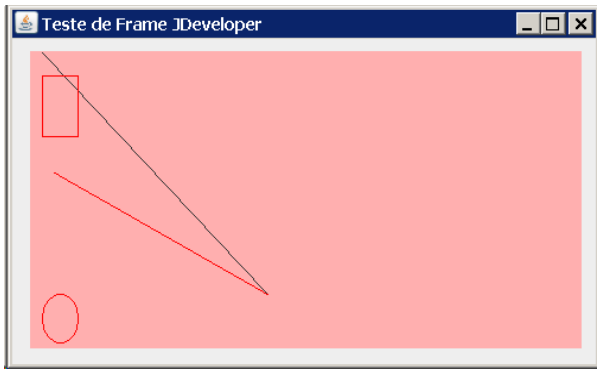
```
public void paintComponent(Graphics g) {
     Graphics2D g2 = (Graphics2G) g;
     ...
}
```

Metodele din Java 2D foloseau iniţial coordonate de tip `float` în locul pixelilor. Manipularea valorilor de tipul primitiv `float` era însă greoaie deoarece programatorul trebuia să facă conversia valorilor din `double` la `float`. Din acest motiv Java 2D a implementat metode atât cu parametrii `float`, pentru cei dornici de cast-uri, cât şi cu parametri `double` pentru cei ce doresc sa scrie mai puţin. Astfel se o linie se poate trasa în două modalităţi:

```
Line2D l1 = new Line2D.Float(10.F,1.0F,200.F,200.F);
Line2D l2 = new Line2D.Double(20.,100.0,200.,200.);
```

Primitive grafice 2D

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Shape;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;
import javax.swing.JPanel;

class TestJPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;

        //linie de la (x1,y1 la x2,y2)
        Line2D l1 = new Line2D.Float(10.F, 1.0F, 200.F, 200.F);
        Line2D l2 = new Line2D.Double(20., 100.0, 200., 200.);
        //dreptunghi din (x1,y1 cu latime, inaltime)
        Rectangle2D d = new Rectangle2D.Double(10, 20, 30, 50);
        //elipsa intr-un dreptunghi
        //din (x1,y1 cu latime, inaltime)
        Ellipse2D e = new Ellipse2D.Double(10,200,30,40);

        g2.draw(l1);
        //desenare cu rosu
        g2.setPaint(Color.RED);
        g2.draw(l2);
        g2.draw(d);
        //fond roz
        super.setBackground(Color.PINK);;
        g2.draw(e);
        //pentru umplere g2.fill(d);


    }
}
```



C10- Întrebari

1. Comparati Frame-ul si Panel-ul.
2. Ce modificari trebuie aduse primei aplicaţii pentru afisarea unui Frame cu titlul “Salutare …”.
3. Descrieti procedura prin care se utilizeaza JPanelul pentru afisarea de obiecte geometrice?
4. Cum se deseneaza un patrat?
5. Cum se deseneaza un cerc?