Lucrarea nr. 12 - Grafice în Java

Breviar teoretic

Deoarece graficele sunt de departe elementele cele mai sugestive în cadrul unor aplicații de tip pagini web am considerat că se cuvine să acordăm un spațiu separat facilităților de desenare pe care le oferă mediul Java. Deoarece Windows este un sistem grafic, tot ce este afişat în cadrul unei ferestre sunt elemente grafice, fie că este vorba de un text, fie că este vorba de desene sau animații complicate.

Elemente grafice de tip bara de defilare

O bară de defilare poate fi implementată prin utilizarea unui obiect de tip **scrollbars**. Structura unui astfel de obiect este:

```
Scrollbar scrollbar = new Scrollbar(orientation, start, page, min, max);
```

Orientarea obiectului poate fi specificată prin argumentele: **Scrollbar.HORIZONTAL** şi **Scrollbar.VERTICAL**, **start** va indica printr-un întreg valoarea de la care fi localizat inițial cursorul, **page** va identifica pagina – uzual va avea valoarea 0, **min** va indica valoarea minimă, iar **max**, va indica valoarea maxima a obiectului.
Spre exemplu o bară de defilare a cărei valoare va putea fi modificată între 0 şi 100 cu valoarea implicită de 50 va fi specificată astfel:

```
Scrollbar scrollbar=new Scrollbar(Scrollbar.HORIZONTAL, 50, 0, 1, 100);
```

Evenimente asupra barei de defilare

Metoda care se ocupă de tratarea evenimentelor ce apar în cadrul unei aplicații Java este **handleEvent()** din cadrul clasei **Component**. Această metodă gestionează toate evvenimentele ce pot apărea în cadrul unei aplicații Java. În tabelul următor sunt prezentate primcipalele evenimente tratate de metoda handleEvent().

ACTION_EVENT	Un eveniment generat de metoda action().
GOT_FOCUS	Componenta este focalizată.
KEY_PRESS	A fost activată o tastă.
KEY_RELEASE	A fost eliberată o tastă.
LIST_DESELECT	Un element dintr-o listă a fost deselectat.
LIST_SELECT	Un element al listei a fost selectat.
LOST_FOCUS	Componenta nu mai este focalizată.
MOUSE_DOWN	Utilizatorul a apăsat butonul stâng al mouse-ului
MOUSE_DRAG	Mouse-ul a fost mişcat având butonul stâng apăsat
MOUSE_ENTER	Mouse-ul a fost mutat într-o zonă de interes.
MOUSE_EXIT	Mouse-ul a părăsit o zonă de interes.
MOUSE_MOVE	Mouse-ul a fost mişcat de utilizator.
MOUSE_UP	Utilizatorul a eliberat butonul stâng al mouse-ului.
SCROLL_ABSOLUTE	Utilizatorul a mutat cursorul barei de defilare.
SCROLL_LINE_DOWN	Utilizatorul a activat săgeata din zona inferioară a barei de defilare.
SCROLL_LINE_UP	Utilizatorul a activat săgeata din zona superioară a barei de defilare.
SCROLL_PAGE_DOWN	Utilizatorul a activat zona de sub cursorul barei de defilare.
SCROLL_PAGE_UP	Utilizatorul a activat zona de deasupra cursorului barei de defilare.
WINDOW_DEICONIFY	Fereastra a fost restaurată de la dimensiunea de icon.
WINDOW_DESTROY	Fereastra a fost distrusă.
WINDOW_EXPOSE	Fereastra a fost activată.

WINDOW_ICONIFY	Fereastra a fost redusă la un icon.
WINDOW_MOVED	Feresatra a fost mutată.

Prin urmare unei bare de defilare i se pot asocia 5 metode şi implicit 5 evenimente distincte posibile SCROLL_ABSOLITE, SCROLL_LINE_DOWN, SCROLL_PAGE_DOWN, şi SCROLL_PAGE_UP. După determinarea unuia dintre aceste evenimente, pot fi determinati prin utilizarea unor metode suplimentare, o serie de parametri. În tabelul următor sunt prezentate metodele ce pot furniza informaţii asupra evenimentelor ce au avut loc în legătură cu bara de defilare-Scrollbar:

Metoda	***Descriere***
`Int getLineIncrement()`	Furnizează incrementul liniei
`Int getMaximum()`	Oferă valoarea maximă
`Int getMinimum()`	Oferă valoarea minimă
`Int getOrientation()`	Oferă orientarea barei
`Int getPageIncrement()`	Indică valoarea incrementului paginei
`Int getValue()`	Furnizează valoarea curentă a poziţiei cursorului
`Int getVisible()`	Indică valoarea paginei
`setLineIncrement(int inc)`	Setează increment-ul liniei
`setPageIncrement(int inc)`	Setează increment-ul paginei
`setValue(int value)`	Setează poziţia cursorului la valoarea argument
`setValues(int value, int visible,int minimum,int maximum)`	Setează bara de defilare la valoarile argument: pgsize, int min, int max)

Aplicatie: Creaţi şi implementaţi urmatoarele variante pentru applet-ul *BaraDefilare*:

`Varianta 1`

```
import java.awt.*;
import java.applet.*;
//sunt importate pachetele necesare


public class BaraDefilare extends Applet
{

Scrollbar BaraDefilare; String s;

public void init()
{//se initializează elementele appletului
BorderLayout layout = new BorderLayout();
setLayout(layout);

//se stabileste gestionarul de poziţie

BaraDefilare=new Scrollbar(Scrollbar.HORIZONTAL,50, 0, 1, 100);

add("North", scrollbar);
//se adaugă bara de defilare
s = "50";

Font font = new Font("TimesRoman", Font.BOLD, 72);
setFont(font);

resize(200, 200);
//s-a stabilit valoarea s, tipul de font şi dimensiunea
}
public void paint(Graphics g)
{//s-a lansat metoda de desenare
g.drawString(s, 60, 120);
```

```
//este  afişată  valoarea  şirului  s
}


public  boolean  handleEvent(Event  evt)
{//tratarea evenimentelor ce pot apărea
if (evt.target instanceof Scrollbar)
         {

scrollbar = (Scrollbar)evt.target;
int value = BaraDefilare.getValue();
s = String.valueOf(value);
repaint();

return  true;
   }

else
    {

boolean result = super.handleEvent(evt);
return result;
     }

  }
}
```

Varianta 2

```
import java.awt.*;
import java.applet.*;
//sunt  importate  pachetele  necesare

public class BaraDefilare extends Applet
{

Scrollbar BaraDefilare;
String s;

public  void  init()
{//se initializează elementele appletului

BorderLayout layout = new BorderLayout(); setLayout(layout);
//se stabileste  gestionarul  de  poziţie

BaraDefilare=new Scrollbar(Scrollbar.HORIZONTAL,50,0,1,100);

add("North", BaraDefilare);
//se adaug ă bara de defilare s = "50";

Font font = new Font("TimesRoman", Font.BOLD, 72);
setFont(font);

resize(200,  200);
//s-a stabilit valoarea  s,  tipul de font şi dimensiunea
}

public void paint(Graphics g)
{//s-a lansat metoda de desenare
g.drawString(s,  60,  120);
//este  afişată  valoarea  şirului  s
}

public void raspuns()

{
int value = BaraDefilare.getValue();
s = String.valueOf(value);
```

```
repaint();
}

public  boolean  handleEvent(Event  evt)

{//tratarea evenimentelor ce pot apărea switch (evt.id) {
case Event.SCROLL_ABSOLUTE: {raspuns();return true;};
case Event.SCROLL_LINE_DOWN: {raspuns();return true;};
case Event.SCROLL_LINE_UP: {raspuns(); return true;};
case Event.SCROLL_PAGE_DOWN:{ raspuns(); return true;};
case Event.SCROLL_PAGE_UP: {raspuns();return true;};
default: return false;
      }
   }
}
```

Probleme propuse

1. Inhibați pe rând metodele ce se ocupă cu bara de defilare. Ce observați?

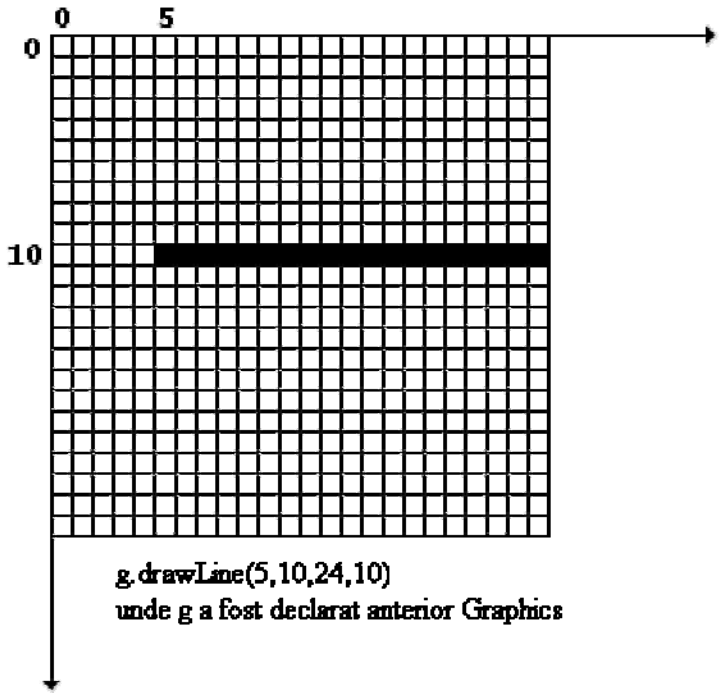
2. Utilizați metodele specifice mouse-ului şi apoi pe cele specifice ferestrelor, prin rescrierea metodei handleEvent.

3. Utilizați 2 bare de defilare şi afişați atât valoarea numerică a fiecărei bare, cât şi suma, produsul şi câtul celor 2 valori(utilizați un gestionar de poziție corespunzător, iar pe cât posibil faceți în aşa fel încât cele 2 bare de defilare sa fie una verticală şi cealaltă orizontală).

Mediul de desenare al unui applet - Canvas

Fiecare applet are o zonă numită canvas, în care afişăm ceea ce dorim. Parametrii de desenare ai unui applet sunt specificați prin intermediul fişierului html ca argumente pentru variabilele weidth şi height ce apar în cadrul unui tag de tip **<applet>.** Orice este desenat în afara acestor limite nu va fi vizibil pe monitor.

Coordonatele utilizate în desenare sunt exprimate în pixeli, puntul de coordonate 0,0 fiind colțul ecranului din stânga sus. Incrementarea valorii lui z duce la deplasarea pe verticală în jos, iar incrementarea valorii lui x duce la deplasarea pe orizontală spre dreapta.



Desenarea formelor

Pentru utilizarea metodelor grafice disponibile în Java va trebui în primul rând importate pachetele ce permit lucrul cu grafice . Acest lucru se realizează astfel:

```
import  java.awt.Graphics
```

sau mai general: `import  java.awt.*`

Cele mai utilizate metode de desenare sunt prezentate în tabelul următor:

Metoda	***Descriere***
clearRect()	Şterge o zonă dreptunghiulară dintr-un desen – canvas.
copyArea()	Copiază o zonă dreptunghiulară dintr-un desen într-o altă zonă.
drawArc()	Desenează un arc de cerc.
drawLine()	Desenează o linie dreaptă.
drawOval()	Desenează o elipsă.
drawPolygon()	Desenează un poligon.
drawRect()	Desenează un dreptunghi.
drawRoundRect()	Desenează un dreptunghi cu colturi rotunjite.
drawString()	Afişează un şir de tip text.
fillArc()	Deseanează un arc cu interiorul colorat sau haşurat.
fillPolygon()	Deseanează un poligon cu interiorul colorat sau haşurat.
fillRect()	Deseanează un dreptunghi cu interiorul colorat sau haşurat.
fillRoundRect()	Deseanează un dreptunghi cu interiorul colorat sau haşurat şi margini rotunjite.
getColor()	Extrage culoarea de desenare curentă.
getFont()	Extrage fontul curent utilizat.
getFontMetrics()	Extrage informaţiile despre fontul curent.

Prin urmare pentru a desena un anumit element grafic va fi apelată metoda respectivă, furnizându-se totodată şi argumentele corespunzătoare. Astfel pentru a desena o linie între coordonatele 5,10 şi 20,30 apelul metodei **drawLine** va fi următorul:

```
g.drawLine(5, 10, 20, 30);
```

Elementul g va fi declarat anterior ca fiind un obiect de tip **Graphics**, iar desenarea prin respectiva metodă va avea loc în cadrul unei metode de tip **paint()** , argumentele metodei de desenare fiind coordonatele X, Y ale începutului lşinie, respectiv sfârşitul acesteia.

Desenarea unui dreptunghi

În cadrul acestui subpunct este exemplificată desenarea unui dreptunghi. Metoda este asemănătoare celei de desenare a unei linii. În plus dacă se doreşte pe lângă desenarea dreptunghiului, inserarea în interiorul său a unui alt dreptunghi cu marginile rotunjite, vor fi apelate două metode: prima pentru dreptunghi -- **g.drawRect(startX, startY, lungime, înălțime)**, iar cea de-a doua metodă **-**
`g.fillRoundRect(startX, startY, lungime, înălțime, lungRound, înaltRound);`

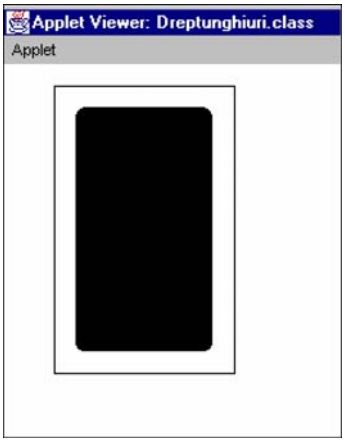
unde **lungRound, înaltRound** sunt lungimea – **lung**, respectiv înălțimea - **înalt** dreptunghiului de-a lungul cărei diagonale se va trasa arcul de cerc ce va fi amplasat la colțurile dreptunghiului declarat anterior

Aplicatie: Exemplu de desenare a unui dreptunghi in cadrul unui applet Java ***Dreptunghiuri:***

```
import java.awt.*;
import java.applet.*;


public  class  Dreptunghiuri  extends  Applet
{

public void paint(Graphics g)
  {
      //se lansează clasa de desenare

      g.drawRect(35, 15, 125, 200);
      //este desenat dreptunghiul gol

      g.fillRoundRect(50, 30, 95, 170, 15, 15);
      //este  desenat  dreptunghiul  cu  interior  colorat  şi  margini
      rotunjite

  }
}
```

Se reaminteşte că toate appletele nu pot fi rulate decât având ca suport un fişier html. Un exemplu de fişier HTML suport este prezentat în continuare:

```
<title>Exemplu  de  desenare  a  unor  dreptunghiuri</title>

<h1>Pagina  suport  Applet  de  desenare  a  dreptunghiurilor</h1>

<applet  code="Dreptunghiuri.class"  width=200  height=250>  </applet>
```



Desenarea unor elipse

Acest lucru se realizează relativ simplu, pruin apelul unui obiect al clasei **Graphics**. Forma acestei metode **drawOval(),** este următoarea:

```
instanţă_obiect_Graphics.drawOval(start X, start Y, lungime, înălţime);
```

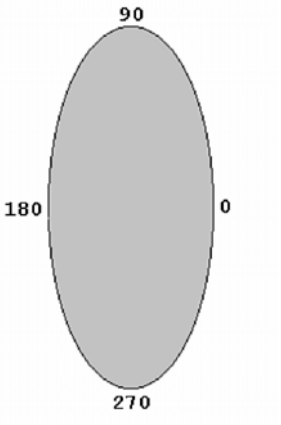
Desenarea unor arce de cerc

Forma generală a metodei de desenare a unui arc de cerc are următoarea structură:

```
g.drawArc(startX,startY, lungime, înălţime, unghi_start, număr_grade);
```

unde **starX**, **startY** reprezintă punctul de start al arcului, lungimea şi înălţimea reprezintă dimensiunea dreptunghiului care va înscrie cercul sau elipsa descrisă de respectivul arc**, unghi_start** reprezintă valoarea de start a capului inferior a arcului, **număr_grade** reprezintă unghiul cu vârful la centru ce va fi descris de raza generatoare a cercului/elipsei .

Modul de măsurare a elementelor de tip unghi este indicat în figura următoare:

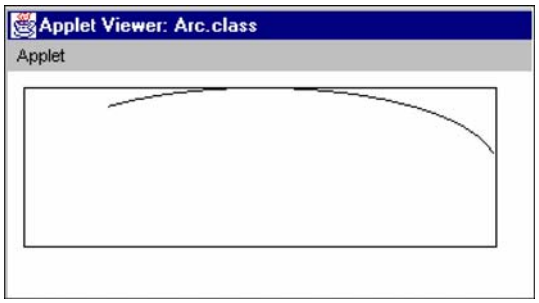


Un exemplu simplu este prezentat în continuare. Se presupune că se doreşte desenarea unui arc de cerc, ce porneşte din punctul de coordonate 10,10 , încadrat de un dreptunghi cu dimensiunile de 300,100 , unghiul de start de 10 grade, iar numarul de grade 120.

Pentru a înţelege rolul dreptunghiului ce înscrie arcul a fost adăugat în fişierul sursă şi desenarea unui dreptunghi având aceleaşi coordonate ca şi elementul suport.

Aplicatia de desenare a unui arc de cerc in cadrul unui applet Java - varianta 1

```
import java.awt.*;
import java.applet.*;
public  class  Arc  extends  Applet

{
public  void  paint(Graphics  g)

      {//se lansează clasa de desenare g.drawArc(10,10,300,100,10,120);

      //este desenat arcul cu argumentele specificate in cadrul
      explicatiilor
      g.drawRect(10,10,300,100);
      //este  desenat  dreptunghiul  suport
      }
}
```

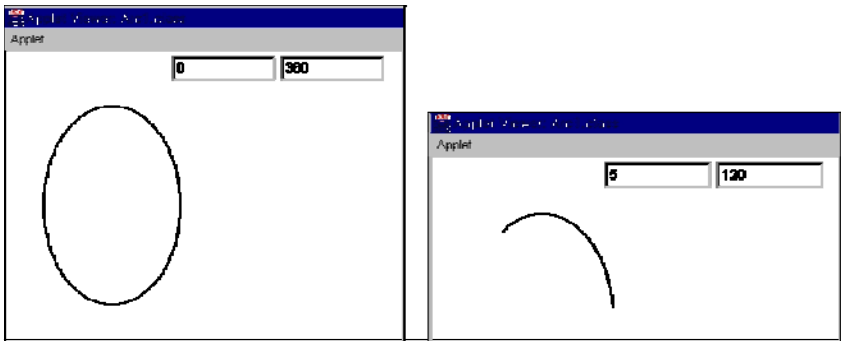


Aplicatia de desenare a unui arc de cerc in cadrul unui applet Java - varianta 2

Următorul exemplu utilizează metoda **parsInt()** pentru a prelua argumente din cadrul unor câmpuri de tip text.Totodată sunt introduse o serie de câmpuri de tip text care oferă argumente pentru desenarea arcului de cerc. Explicarea elementelor noi se va realiza în applet-ul ***Arc1***:

```
import java.awt.*;
import java.applet.*;

public class Arc1 extends Applet {
//sunt declarate variabilele de clasă

TextField  unghiStart,  numarGrade;

public  void  init()

{//se indică folosirea unor câmpuri Text  pentru preluarea unor parametri

unghiStart =  new TextField(10);

numarGrade =  new TextField(10);
//la  canvas  sunt  adăugate  cele  2  câmpuri  de  tip  text

add(unghiStart);
add(numarGrade);

//iar apoi sunt inițializate cu anumite valori unghiStart.setText("0");
numarGrade.setText("360");
}
```

```
public void paint(Graphics g)
{
      //este lansată funcţia de desenare
      String ungh = unghiStart.getText();

      //valoarea  din  cadrul  câmpului  text  este  memorată  în
      variabila  ungh

      int start = Integer.parseInt(ungh);
      //variabila ungh de tip String este convertită la întreg


      grd = numarGrade.getText();
      int grade=Integer.parseInt(grd);

      g.drawArc(35, 50, 125, 180, start, grade);

      //este lansată desenarea arcului cu punctul de start 35,50,
      dreptunghiul de încadrare de 125,180, având ca unghi de start
      valoarea indicată de variabila start, iar ca numar de grade variabila
      grade

}


public boolean action(Event event, Object arg)
      {
      //la evenimentul apasare tasta ENTER este   generată redesenarea
      appletului

      repaint();
      return true;
      }
}
```



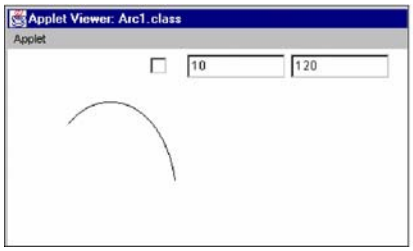
Obs. Experimentaţi modificarea argumentelor în clipa utilizării altor evenimente – spre exemplu GOT_FOCUS.

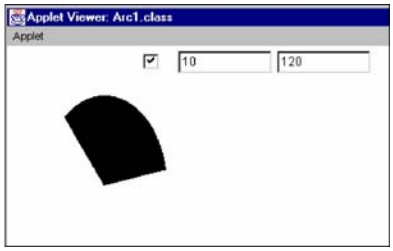
Aplicatia de desenare a unui arc de cerc in cadrul unui applet Java - varianta 3

Următorul exemplu va adăuga la exemplul anterior un obiect de tip **Checkbox**, la activarea căruia va fi lansată desenarea unui arc de cerc al cărui interior este colorat. Se observă că evenimentul de tip validare căsuţă este captat prin metoda **getState().**

```
import java.awt.*;
import java.applet.*;
```

```
public class Arc1 extends Applet
{//sunt declarate variabilele de clasă

TextField unghiStart, numarGrade;

Checkbox c = new Checkbox();


public void init()

{//se indică folosirea unor câmpuri Text pentru preluarea unor
parametri

unghiStart = new TextField(10);
numarGrade = new TextField(10);

//la canvas sunt adăugate cele 2 câmpuri de tip text si un
checkbox

add(c);
add(unghiStart);
add(numarGrade);
//iar apoi sunt inițializate cu anumite valori unghiStart.setText("0");
numarGrade.setText("360");

}

public void paint(Graphics g)
{//este lansată funcţia de desenare
String ungh = unghiStart.getText();
//valoarea din cadrul câmpului text este memorată în variabila
ungh

int start = Integer.parseInt(ungh);
//variabila ungh de tip String este convertită la întreg
String grd =numarGrade.getText();
int grade =Integer.parseInt(grd);
boolean s = c.getState();


if (s==true) g.fillArc(35, 50, 125, 180, start, grade);
else g.drawArc(35, 50, 125, 180, start, grade);

//este lansată desenarea arcului cu punctul de start 35,50, dreptunghiul
de încadrare de 125,180, având ca unghi de start valoarea indicată de
variabila start, iar ca numar de grade variabila grade

}

public boolean action(Event event, Object arg)

{//la orice eveniment este generată redesenarea appletului
repaint();
return true;
}
}
```





Desenarea unor poligoane

Poligoanele sunt figuri geometrice cu mai multe laturi. Spre exemplu triunghiul este cel mai simplu poligon. În Java se poate desena un poligon creând un şir de puncte ce reprezintă coordonatele fiecărui capăt al liniilor ce compun poligonul. Spre exemplu dacă se doreşte desenarea unui poligon cu 6 laturi se va crea un şir cu 7 elemente:

```
int x[]={35, 150, 60, 140, 60, 150, 35};
//coordonatele  x

int y[]={50, 80, 110, 140, 170, 200, 230};
//coordonatele y
int numPts = 7;
```

Primul şir x[] memorează coordonatele X ale punctelor, în timp ce y[], memorează coordonatele Y. În aceste condiții desenarea unui poligon se poate realiza prin metoda de forma:

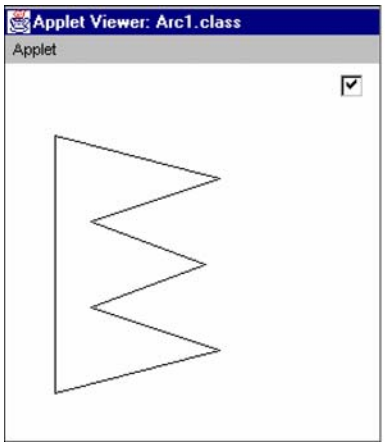
```
g.drawPolygon(x, y, numPts);
```

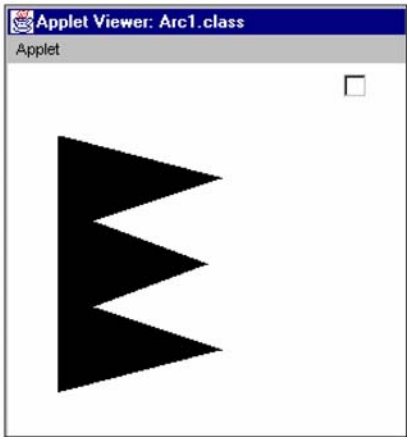
Ca o observație, pentru a obține un contur închis, ultimul punct va trebui să coincidă cu primul, dacă se utilizează metoda **drawPolygon**, iar în cazul utilizării metodei **fillPolygon**, acest lucru nu mai este necesar deoarece culoarea ce va colora interiorul desenului va da impresia de poligon închis.

Aplicatie de desenare a unor poligoane a caror umplere este controlata prin activarea sau dezactivarea unui obiect de tip Chekbox

```
import java.awt.*;
import java.applet.*;

public class Poly extends Applet
{//sunt declarate variabilele de clasă

int x[]={35, 150, 60, 140, 60, 150, 35};
//coordonatele x
int y[] = {50, 80, 110, 140, 170, 200, 230};
//coordonatele y
int numPts = 7;

Checkbox c=new Checkbox();

public void init()
 {
      add(c);
 }

public  void  paint(Graphics  g)
 {
      boolean s=c.getState();
```

```
      if (s==true) g.drawPolygon(x, y, numPts);
      else g.fillPolygon(x, y, numPts);
 }


public boolean action(Event event, Object arg)
 {
      repaint();
      return true;
 }
}
```



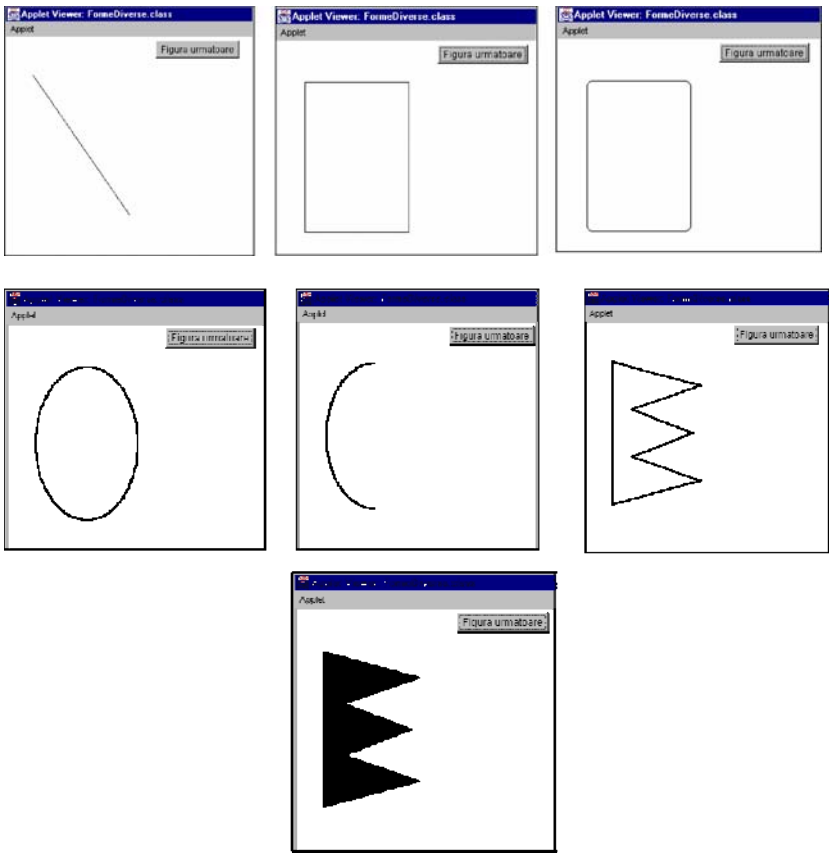


Aplicatie de desenar a unor forme diverse

În continuare va fi prezentat un program care reuneşte toate elementele grafice prezentate în cadrul aplicaţiilor de desenare, utilizînd totodată şi o serie de elemente de tip interfaţă grafică.

```
import java.awt.*;
import java.applet.*;


public  class  FormeDiverse  extends  Applet
{
int forma;
Button buton;


public  void  init()

 {
      forma = 0;
      buton = new Button("Figura urmatoare");
      add(buton);
      }



public void paint(Graphics g)
{

      int x[] = {35, 150, 60, 140, 60, 150, 35};
      int y[] = {50, 80, 110, 140, 170, 200, 230};
```

```
        int numPts = 7;
        switch(forma)
        {
case 0:g.drawLine(35, 50, 160, 230); break;
case 1:g.drawRect(35, 50, 125, 180); break;

case 2:g.drawRoundRect(35,50,125,180,15,15);break;
case 3:g.drawOval(35, 50, 125, 180); break;
case 4:g.drawArc(35, 50, 125, 180, 90, 180); break;
case 5:g.drawPolygon(x, y, numPts); break;
case 6:g.fillPolygon(x,  y,  numPts);break;

        }
 }

public  boolean  action(Event event, Object arg)

{
        ++forma;
        if (forma == 7) forma = 0;
        repaint();
        return true;
 }
}
```



Probleme propuse

1. Pentru applet-ul Arc1 experimentați modificarea argumentelor în clipa utilizării altor evenimente – spre exemplu GOT_FOCUS.
2. Scrieți codul necesar desenării unui dreptunghi cu lungimea de 200 şi înălțimea de 100, având ca punct de start punctul de coordonate 50,75.
3. Scrieți un applet care va desena un pătrat înscris într-un cerc.
4. Scrieți un applet care va permite utilizatorului să introducă punctul de start, lungimea şi înălțimea unui dreptunghi a cărei desenare va fi lansată la activarea unui buton de OK. Desenul realizat anterior apăsării butonului OK va fi şters.
5. Modificați appletul Arc astfel încât utilizatorul să poată introduce ca opțiune nu numai punctul de start al arcului, dar şi dimensiunea arcului.
6. Scrieți un applet care va desena o față umanioidă, realizată din elipse şi arce de cerc.