**6.1. Crearea applet-urilor.**

Unul dintre scopurile de dezvoltare Java: crearea applet-uri - mici programe care se execută pe Web-browser. Deoarece applet-urile trebuie să fie secure, ele sunt limitate în capacitatea lor, deşi ramin în continuare un instrument puternic pentru programarea Web de partea clientului.

/ / Exemplu # 1: Un applet simplu: //

**import** java.applet.Applet;

**import** java.awt.\*;

**public** **class** FirstApplet **extends** Applet {

**private** String date;

**public** **void** init() {

date = **new** java.util.Date().toString();

}

**public** **void** paint(Graphics g) {

g.drawString("Apletul este activ:", 50, 15);

g.drawString(date, 50, 35);

}}

Pentru a afişa ora şi data curentă, în acest exemplu a fost folosit obiectul **Date** din pachetul **java.util**. Metodă **ToString()** este folosită pentru a converti informaţia din obiect într-un şir de caractere extras într-un applet folosind metoda **drawString()**. Parametrii acestei metode cunt coordonatele textului, începând cu colţul de sus din stânga a appletului.

Applet-ul nu are nevoie de o metodă **main ()** – codul de rulare pote fi plasat în metoda **init ()** sau **paint ()**. Pentru a rula applet-ul trebuie de întrodus referinţa la clasa lui în HTML-document şi să vizualizaţi acest document cu Web-browserul care acceptă Java. În acest caz, se poate de făcut un fragment foarte simplu (tag) **<applet>** într-un document HTML în **view.htm**l:

**<html><body>**

**<applet code= FirstApplet.class width=300 height=300>**

**</applet>**

**</body></html>**

Fişierul **FirstApplet.class** pentru un astfel de activare trebuie să fie în acelaşi director cu documentul HTML. Realizarea HTML-documentului are loc în browser-ul Microsoft Internet Explorer sau Netscape Navigator.

Exemplu applet-ului, in care se utilizeaza medoda init()**paint()**, medoda setColor() care atribue culoare caracterelor, metoda drawString() desenarea şirurile.

/ \* Exemplu # 1: exstragerea şirurilir cu diferite culori: \* /

**import** java.applet.\*;

**import** java.awt.\*;

**public** **class** ColorConstants **extends** Applet {

**public** **void** init(){ }

**public** **void** paint(Graphics g){

g.setColor(Color.yellow);

g.drawString("<Yellow>", 5, 30);

// culoare galbenă

g.setColor(Color.blue);

g.drawString("<Blue>", 5, 60);

//culoare albastră

g.setColor(Color.green);

g.drawString("<Green>", 5, 90);

//culoare verde

}}

Majoritatea metodelor grafice utilizate in applet-uri sunt metode abstracte din clasa de baza **Graphics** din pachetul **java.awt.** Metodele applet-ului primesc obiectul de tip **Graphics** in calitate de parametru, şi impreuna cu el primeste culoarea curentă, fontul, poziţia cursorului. Setarea contextului se realizează de obicei de metoda **udate()** sau **paint()**.

Cele mai des utilizate metode din aceasta clasa:

**drawLine(int x1, int y1, int x2, int y2)** – desena linie

**drawRect(int x, int y, int width, int height)**şi  **fillRect(int x, int y, int** **width, int height)** − deseană un dreptunghi un colorat;

**draw3DRect(int x, int y, int width, int height, boolean raised)** −deseană dreptunghi tridimensional;

**drawRoundRect(int x, int y, int width, int height, int** **arcWidth, int arcHeight)** −deseană un dreptunghi rotunjit;

**drawOval(int x, int y, int width, int height)** − deseană un oval;

**drawPolygon(int[] xPoints, int[] yPoints, int nPoints)** − deseană un poligon dat de matricele x si y;

**drawPolygon(Polygon p)** − poligon, având în vedere obiectul Poligon;

**paint()**, **drawPolyline(int[] xPoints, int[] yPoints, int nPoints)** −deseană secvenţă de linii, date de matricele x si y;

**drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)** − deseană un arc

**drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)** − adaugă imaginea;

**drawString(String str, int x, int y)** −deseană un texst;

**setColor(Color c), getColor()** −setează şi returnează culoarea curentă;

**getFont()** −returnează fontul curent;

**setFont(Font font)** −setează fontul specificat.

Exemplele de mai jos, demonstrează modul de utilizare a metodelor clasei **Graphics** pentru a afişa imagini grafice într-o fereastră applet.

/ / Exemplu # 2: desenarea poligonului:

**import** java.applet.\*;

**import** java.awt.\*;

**public** **class** DrawPoly **extends** Applet {

**int** poly1\_x[] = { 40, 80, 0, 40};

**int** poly1\_y[] = { 5, 45, 45, 5};

**int** poly2\_x[] = {140, 180, 180, 140, 100, 100, 140};

**int** poly2\_y[] = {5,25,45,65,45,25,5};

**int** poly3\_x[] = { 240, 260, 220, 260, 220, 240};

**int** poly3\_y[] = {5, 65, 85, 25, 25, 5};

**public** **void** paint(Graphics g) {

g.drawPolygon(poly1\_x, poly1\_y, poly1\_x.length);

g.drawPolygon(poly2\_x, poly2\_y, poly2\_x.length);

g.drawPolygon(poly3\_x, poly3\_y, poly3\_x.length);

}}

/ / Exemplu # 3: dreptunghi tridimensional:

**import** java.applet.\*;

**import** java.awt.\*;

**public** **class** ThreeDRect **extends** Applet {

**public** **void** draw3DRect(Graphics g, **int** x, **int** y, **int** width, **int** height,

**boolean** raised) {

g.draw3DRect(x,y,width-1, height-1, raised);

g.draw3DRect(x+1, y+1, width-3, height-3, raised);

g.draw3DRect(x+2, y+2, width-5, height-5, raised);

}

**public** **void** fill3DRect(Graphics g, **int** x, **int** y, **int** width, **int** height,**boolean** raised)

{ g.draw3DRect(x, y, width-1, height-1, raised);

g.draw3DRect(x + 1, y + 1, width - 3, height - 3, raised);

g.draw3DRect(x + 2, y + 2, width - 5, height - 5, raised);

g.fillRect(x + 3, y + 3, width - 6, height - 6);

}

**public** **void** paint(Graphics g){

g.setColor(Color.gray);

draw3DRect(g, 10, 5, 80, 40, **true**);

draw3DRect(g, 130, 5, 80, 40, **false**);

fill3DRect(g, 10, 55, 80, 40, **true**);

fill3DRect(g, 130, 55, 80, 40, **false**);

} }

/ / Exemplu # 4: exstragerea în ferestră a GIF-imaginei:

**import** java.applet.\*;

**import** java.awt.\*;

**public** **class** DrawImage **extends** Applet {

Image img;

**public** **void** init() {

img = getImage(getCodeBase(), "cow.gif"); }

**public** **void** paint(Graphics g){

g.drawImage(img, 0, 0, **this**);

}}

Când sunt utilizate proprietăţile tag-ul <applet> este posibil de a trimite parametrii din HTML-document în codul applet-ul.

/ / Exemplu # 5: Transmiterea parametrilor pentru applet:

**import** java.awt.\*;

**import** java.applet.\*;

**public** **class** ReadParam **extends** Applet{

**int** b;

String name = "OK";

**boolean** statement;

**public** **void** start() {//citirea parametrilor

String param;

param = getParameter("statement");

**if**(param != **null**)

statement = Boolean.valueOf(param).booleanValue();

**try** {

param = getParameter("bNumber");

**if**(param != **null**) //dacă nu este găsit

b = Integer.parseInt(param);

} **catch**(NumberFormatException e) {

b = 0;

statement = **false**;

}}

**public** **void** paint(Graphics g) {

**double** d = 0;

**if**(statement)

d = Math.pow(b,2);

**else**

name = "FALSE !";

g.drawString("Name: " + name, 0, 11);

g.drawString("Value b: " + b, 0, 45);

g.drawString("b power 2: " + d, 0, 62);

g.drawString("Statement: "+statement,0,79);

} }

Dacă parametru nu este disponibil, **getParameter ()** returnează null. HTML-documentul pentru acest exemplu ar putea arata ca:

**<html><head><title>Parametrii Apletului</title></head>**

**<body>**

**<applet code=ReadParam.class width=250 height= 300>**

**<param name = bNumber value = 4>**

**<param name = statement value = true>**

**</applet></body> </html>**

Pentru a utiliza componente în applet-uri se foloseasc clase interfete în care componentele de pe ecran se crează prin mijloacele Java şi în măsura minimă depinde de platforma şi echipament. Acest tip de clase componente au fost adunate în biblioteca **Swing.**

/ / Exemplu # 6: applet cu componente:

**import** javax.swing.\*;

**import** java.awt.\*;

**public** **class** MyJApplet **extends** JApplet {

JLabel lbl=

**new** JLabel("Swing-applet!");

**public** **void** init() {

Container c = getContentPane();

c.add(lbl);

}}

In acest program se realizează plasarea etichetei text **JLabel** pe formă in applet. Constructorul clasei **JLabel** primeste obiectul **String** şi foloseşte valoarea ei la crearea etichetei. Apelat în mod automat la încărcarea applet-ului metoda **init()**, responsabilă de initializarea câmpurilor şi introducerea componentei pe formă. Pentru acesta se apeleaza metoda **add()** din clasa **Container**, care introduce componentele într-un container. Metoda **add()** nu este activată imediat, ca şi în biblioteca **AWT**. Pachetul **Swing** impune ca toate componentele sa fie adăugate in panoul formei ContentPane, de aceia trebuie iniţial de activat metoda **getContentPane() a** clasei **JApplet** pentru a crea o referinţă la obiect, ca parte a procesului **add().**

/\* exemlu #7 : lucrul metodei init(), start(), stop() \*/

**import** java.awt.\*;

**import** javax.swing.\*;

**public** **class** InitDemo **extends** JApplet {

**private** **int** i;

**private** String msg = **null**;

**private** Color c;

**public** **void** init() {

c = **new** Color(0, 0, 255);

**this**.setBackground(c);

**this**.setForeground(Color.white);

setFont(**new** java.awt.Font(

"Courier", 1, 14));

msg = " initialization";

i = 1;

}

**public** **void** start() {

**int** j = i \* 25;

**if** (j < 255){

c = **new** Color (j, j, 255 - j);

}

setBackground(c);

String str = Integer.toString(i);

msg += " " + str;

}

**public** **void** paint(Graphics g) { g.drawString(msg, 30, 30);

}

**public** **void** stop() {

i++;

msg = "start() - stop()";

}}

Atunci când se lucrează cu fonturi, se poate de determinat care dintre ele sunt disponibile şi pot fi utilizate. Pentru a obţine această informaţie, se foloseste metoda **getAvailableFontFamilyNames**() din clasa **GraphicsEnvironment**. Metoda returneaza un array de siruri de caractere care conţine numele de fonturi disponibile.

/ \* Exemplu # 8: acces la fonturile sistemului de operare: \* /

**import** javax.swing.\*;

**import** java.awt.\*;

**public** **class** FontDemo **extends** JApplet {

**private** **int** i;

**private** String msg = **null**;

**private** String[] fonts;

**public** **void** init() {

GraphicsEnvironment ge = GraphicsEnvironment. getLocalGraphicsEnvironment();

fonts = ge.getAvailableFontFamilyNames();

}

**public** **void** start() {

**int** j = i;

**if** (j > fonts.length)

i = 0;

**else**

setFont(**new** Font(fonts[j], 1, 14));

String str = Integer.toString(i);

msg = fonts[j] + " " + str;

i++;

}

**public** **void** paint(Graphics g) {

g.drawString(msg, 30, 30);

} }

Biblioteca **Swing**, precum şi biblioteca **AWT**, susţine un set de metode grafice din clasa **Graphics**. Toate elementele grafice se desenează în fereastră. Apariţia in ferestră se face printr-un context grafic folosind metoda **paint()** sau metoda **update()**. Clasa **Graphics**, obiectul careia este trecut la context, defineşte funcţiile de desen.

**6.2. Frame-uri**

Spre deosebire de applet-uri, aplicaţii grafice care extind clasa **java. awt. Frame** sau subclasa **jawax.swing.JFrame**, nu au nevoie de browser. Pentru crearea interfeţei grafice a aplicatiei **GUI**, trebuie să-i ofere un obiect **Frame** sau **JFrame**, in care vor fi adunate componentele utilizate de aplicaţia **GUI**.

Astfel de aplicaţie se lanseaza cu ajutorul metodei **main()** şi este responsabil pentru afişarea în fereastră. **Frame**-urile sunt utilizate pentru construirea unor sisteme distribuite, care funcţionează în reţele locale şi private.

/ / Exemplu # 9: afişarea textului, liniilor, şi ovalelor:

**import** java.awt.\*;

**import** java.awt.event.\*;

**public** **class** WindowApp **extends** Frame{

String msg = "Window-Application";

**int** x1 = 30, y1 = 50, x2 = 200, y2 = 50;

**public** **void** paint(Graphics g){

g.drawString(msg, 30, 40); /\*exstragerea şirului din poziţia х=30 y=40\*/

g.drawLine(x1, y1, x2, y2);// linie

**int** x = 30, y = 200, width = 150, height = 100;

Color c = **new** Color(255, 100, 100);

/\* culoarea roşie \*/

g.setColor(c);

g.drawOval( x, y, width, height);//oval

g.drawArc(x + 100, y + 50, width - 50, height, 0, 360);//sector

}

**public** **static** **void** main(String args[]){

WindowApp fr = **new** WindowApp();

fr.setSize(**new** Dimension(500, 400));

/\* dimensiunile ferestrei\*/

fr.setTitle("awt-Application");//titlul

fr.setVisible(**true**); // vizibilitatea

fr.repaint();

//redesenare - activînd paint()

} }

Metoda **main()** activează metoada setarii parametrilor ferestrei şi metoda de redisenare a ferestrei. Un număr mare de metode a clasei **Frame** moşteneşte ierarhic de la clasele **Component, Container si Window**. Clasa  **JFrame** din biblioteca **Swing** este o subclasă a clasei **Frame**.

***Lucrare de laborator nr. 6***

**1. Tema lucrării:**

Programarea applet-elor

**2.**  **Scopul lucrării:**

* Însuşirea modalităţilor de creare şi realizare a applet-ului în Java;

**3. Etapele de realizare:**

1. Crearea applet-ului;
2. Metode de activare a applet-ului;
3. Crearea obiectelor grafice în applet;
4. Crearea interfeţii programului;
5. Prezentarea lucrării.

**4. Exemplu de realizare:**

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Main extends Applet {

boolean t = true ;

int y = 9;

int x = 10;

double i = 0.03;

int r = 200;

int b = 0;

int g = 0;

public Color c = new Color(10, 0, 200);

String mesaj=

"Programarea Orientata pe Obiecte - APPLET";

public void start(){t = true;}

public void stop (){t = false;}

public void init (){

super.setSize(500, 500);

}

public void parametri(){

x = x+2;

y = y+3;

}

public void paint(Graphics g){

if(t){

parametri();

g.setColor(Color.white);

g.clearRect(0, 0, getWidth(), getHeight());

if((y%9)==0)

g.setColor(Color.red);

else{

if((y%7)==0)

g.setColor(Color.BLUE);

else{

if((y%5)==0)

g.setColor(Color.GREEN);

else{

if((y%3)==0)

g.setColor(Color.MAGENTA);

else{

if((y%2) == 0)

g.setColor(Color.darkGray);

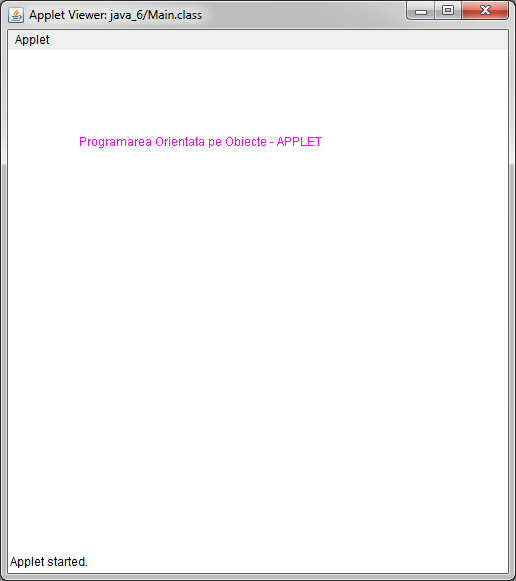
}}}}

g.drawString(mesaj, x, y);

repaint();

}}}

Rezultatul realizării programului:



**5. Probleme propuse:**

Pentru toate variantele de criat un applet care realizează sarcina indicată.

1. Sa se creeze mişcare pe ecran a şirurilor de caractere (unul după altul) dintr-un tablou de şiruri. Direcţia de mişcare în applet şi consecutivitatea fiecărui rând este selectat aleatoriu.
2. De creat un şir de caractere pe ecran, unde caracterele apar din diferite parti ale appletului, şi sa se alinieze unul după altul. Procesul trebuie să fie repetat ciclic.
3. Sa se creeze o mişcare a cercului în applet, astfel încât, atunci când atingeţi limitele ecranului, cercul este reflect, cu efect de comprimare elastică.
4. De prezentat într-un applet apropierea mingei şi deplasarea ei. Mingea trebuie să se mişte, cu o viteza constantă.
5. Sa se reprezinte în applet un segment, ce se roteste în planul ecranului în jurul unuia din punctele finale ale lui. Segmentul trebui să-şi schimbe culoarea de la o poziţie la alta.
6. Sa se reprezinte în applet un segment, ce se roteste în planul ecranului în jurul punctului, care se deplasează de-a lungul segmentului.
7. Să se reprezinte un dreptunghi, care se roteste în planul ecranului în jurul centrului său de greutate.
8. Să se reprezinte un dreptunghi, ce se roteste în planul ecramului în jurul unuia dintre virfuri.
9. Reprezintă un triunghi, ce roteste în plan ecranului în jurul centrului său de greutate.
10. Sa se creeze mişcare pe ecran a imaginilor dintr-un directoriu. Direcţia de mişcare în applet şi consecutivitatea fiecării imagini este selectată aleatoriu.
11. Sa se reprezinte în applet un ceas electronic cu timpul current.
12. Sa se reprezinte în applet calendarul lunii curente.