1. Applet-uri

1.1. Ce este un applet?

Unul dintre scopurile limbajului Java a fost crearea unor programe mici (applet) care să ruleze în interiorul unui browser Web.

Un *applet* reprezintă o suprafaţă de afişare (container) ce poate fi inclusă într-o pagina Web şi gestionată printr-un program Java. Un astfel de program se mai numeşte *miniaplicaţie*.

Codul unui applet poate fi format din una sau mai multe clase. Una dintre acestea este *principală* şi extinde clasa **JApplet**, fiind clasa ce trebuie specificată în documentul HTML ce descrie pagina de Web în care dorim să includem applet-ul.
Diferenţa fundamentală dintre un applet şi o aplicaţie constă în faptul că, un applet nu poate fi executat independent, ci va fi executat de browser-ul în care este încărcată pagina Web ce conţine applet-ul respectiv. O aplicaţie independentă este executată prin apelul interpretorului *java*, având ca parametru numele clasei principale a aplicaţiei, clasa principală fiind cea care conţine metoda *main*.

Un applet nu se poate atinge de hardisk-ul local prin citiri sau scrieri. Scrierea este împiedicată din cauza viruşilor care s-ar putea instala, iar citirea deoarece nu se doreşte preluarea de informaţii de pe staţia locală.

Un neajuns al applet-urilor ar putea fi timpul destul de lung necesar încărcării lor. O metodă de înlăturare a acestui neajuns ar fi arhivarea applet-urilor într-un fişier JAR (Java ARchive) care să cuprindă toate componentele. Astfel, fişierul compresat este download-at la o singură tranzacţie cu server-ul.

Un avantaj al folosirii applet-urilor este lipsa necesităţii instalării lor. De fapt, instalarea este automată de câte ori utilizatorul încarcă pagina Web care conţine applet-ul.

Pachetul care oferă suport pentru crearea de applet-uri este **javax.swing**. Clasa `JApplet` furnizează tot ce este necesar pentru construirea şi întreţinerea unui applet. Crearea unui applet implică implementarea metodelor puse la dispoziţie de clasa `JApplet` care ne ajută să descriem comportamentul dorit al applet-ului.

Ierarhia de clase este:

```
java.lang.Object
   |
   +--java.awt.Component
          |
          +--java.awt.Container
                |
```

1.2. Funcţiile unui applet

Execuţia unui applet începe în momentul în care un browser afişează o pagină Web în care este inclus applet-ul respectiv şi poate trece prin mai multe etape. Fiecare etapă este strâns legată de un eveniment generat de către browser şi determină apelarea unei metode specifice din clasa ce implementează appletul.

- **Încărcarea în memorie** – se creează o instanţă a clasei principale a applet-ului şi se încarcă în memorie.
- **Iniţializarea** – se apelează metoda `init` ce permite iniţializarea diverselor variabile, citirea unor parametri de intrare, etc. Metoda `init` este responsabilă şi pentru aşezarea tuturor componentelor pe formă.
- **Pornirea** – se apelează metoda `start`
- **Execuţia propriu-zisă** – constă în interacţiunea dintre utilizator şi componentele afişate pe suprafaţa applet-ului.
- **Oprirea temporară** – În cazul în care utilizatorul părăseşte pagina Web în care rulează applet-ul se apelează metoda `stop` a acestuia, dându-i astfel posibilitatea să se oprească temporar cât timp nu este vizibil, pentru a nu consuma inutil din timpul procesorului. Acelaşi lucru se întâmplă dacă fereastra browser-ului este minimizată. În momentul când pagina Web ce conţine applet-ul devine din nou activă, va fi reapelată metoda `start`.
- **Oprirea definitivă** – La închiderea tuturor instanţelor browser-ului folosit pentru vizualizare, applet-ul va fi eliminat din memorie şi va fi apelată metoda `destroy` a acestuia, pentru a-i permite să elibereze resursele deţinute. Apelul metodei `destroy` este întotdeauna precedat de apelul lui `stop`.

1.3. Structura generală a unui applet

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

public class StructuraApplet extends JApplet {

      public void init() {
        /*  codul descrie acţiunile care dorim să fie efectuate la instanţierea clasei applet-ului.
   */
      }

      public void start() {
        /*  codul descrie acţiunile care dorim să fie executate la lansarea applet-ului în
    execuţie sau la reîntoarcerea în pagina applet-ului */
      }

      public void paint(Graphics g) {
        /*  codul descrie acţiunile care dorim să fie executate la fiecare redesenare a ferestrei
    applet-ului */
 }

      public void stop() {
        /*  codul descrie acţiunile care dorim să fie executate la oprirea temporară a applet-
    ului (pagina Web nu mai este vizibilă, fereastra browser-ului este minimizată, etc) */
 }

      public void destroy() {
        /*  codul descrie acţiunile care dorim să fie executate la distrugerea applet-ului
    (browser-ul părăseşte documentul .html din care a fost apelat applet-ul)  */
 }
}
```

Observaţie: Aceste metode sunt apelate automat de browser şi nu trebuie apelate explicit din program !

1.4. HTML

Un browser Web interpretează conţinutul (textul) unui fişier .html (Hyper Text Markup Language). Limbajul HTML constă într-o colecţie de marcaje (tag-uri). Marcajele au rolul de a descrie modul în care

va apărea afişat textul, de a comanda browser-ului să utilizeze şi alte resurse Internet, aflate în fişiere diferite.

Sintaxa unui marcaj este:

```
<NumeTag [parametri ]>text </NumeTag>
```

Parametrii se scriu sub forma:

```
NumeParametru = valoare
```

Structura unui document `.html` este:

```
<HTML>
      <HEAD>
         <TITLE>
                  titlul documentului

         </TITLE>
      </HEAD>
         <BODY>
            . . .
            <APPLET parametrii>
                  <PARAM parametrii>
                  <PARAM parametrii>
                  . . .
                  <PARAM parametrii>
            </APPLET>
            . . .
         </BODY>
</HTML>
```

Considerăm cunoscute noţiunile de bază ale HTML. Ceea ce ne interesează pentru construirea applet-urilor este marcajul <APPLET>. Acesta are parametrii obligatorii şi opţionali.

Parametrii obligatorii:

- CODE – valoarea lui este numele fişierului care conţine clasa applet-ului: `NumeClasa.class`;
- WIDTH – valoarea lui este lăţimea ferestrei atribuită de browser applet-ului la afişarea documentului .html;
- HEIGHT – valoarea lui este înălţimea ferestrei atribuită de browser applet-ului la afişarea documentului .html;

Parametrii opţionali:

- CODEBASE – valoarea lui este adresa URL (Universal Resource Locator) sau calea la fişierul cu clasa applet-ului. Dacă parametru lipseşte, căutarea clasei se face în directorul curent (cel din care a fost încărcat fişierul `.html`);
- VSPACE – valoarea lui este înălţimea zonei (exprimată în pixeli) care se lasă liberă deasupra şi dedesubtul ferestrei applet-ului;
- HSPACE – valoarea lui este lăţimea zonei (exprimată în pixeli) care se lasă liberă în stânga şi în dreapta ferestrei applet-ului;
- ALT – Specifică textul ce trebuie afişat dacă browser-ul înţelege marcajul <APPLET> dar nu poate rula applet-uri Java.
- NAME – Oferă posibilitatea de a da un nume respectivei instanţe a applet-ului, astfel încât mai multe applet-uri aflate pe aceeaşi pagină să comunice între ele folosindu-se de numele lor.
- ALIGN – Semnifică modalitatea de aliniere a applet-ului în pagina Web. Acest atribut poate primi una din următoarele valori: `left`, `right`, `top`, `texttop`, `middle`, `absmiddle`, `baseline`, `bottom`, `absbottom`, semnificaţiile lor fiind aceleaşi ca şi la marcajul `IMG`.

Marcajul <PARAM> nu este obligatoriu şi poate să apară atunci când se doreşte ca applet-ul să preia parametrii. Un parametru este definit prin:

- NAME – reprezintă numele variabilei recunoscut de applet;
- VALUE – reprezintă valoarea recepţionată de applet; este de tip String.

1.5. Exemple

Exemplu 1: Exemplifică ordinea apelării metodelor.

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

public class app1 extends JApplet{
  public void init(){
          System.out.println("Sunt in init");
  }

  public void start(){
          System.out.println("Sunt in start");
```

```
  }

  public void paint(Graphics g){
      g.drawString("Sunt in paint", 20, 120);
           System.out.println("Sunt in paint");
  }

  public void stop(){
           System.out.println("Sunt in stop");
  }
  public void destroy(){
           System.out.println("Sunt in destroy");
  }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Primul Applet </TITLE>
</HEAD>
<BODY>
<APPLET CODE="app1.class" WIDTH=300   HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

Exemplu 2: Afişează cateva figuri geometrice.

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

public class app2 extends JApplet{
    public void paint(Graphics g){
      g.setColor(Color.red);
      g.drawRect(10,10,100,200);
      g.setColor(new Color(200,100,255));
      g.fillRect(20,20,50,50);
      g.setColor(Color.blue);
      g.drawOval(60,60,50,50);
    }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Figuri geometrice </TITLE>
```

```
</HEAD>
<BODY>
<APPLET CODE="app2.class" WIDTH=300   HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

Exemplu 3: Afişează lista tuturor font-urilor cunoscute de sistem.

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

public class app3 extends JApplet
{
      public void init(){
            Container c=getContentPane();
            c.setLayout(new FlowLayout());
            JTextArea ta=new JTextArea();
            JScrollPane sp=new JScrollPane(ta);
            sp.setPreferredSize(new Dimension(100, 100));
            c.add(sp, BorderLayout.CENTER);
            GraphicsEnvironment gr=GraphicsEnvironment.
                        getLocalGraphicsEnvironment();
            Font []f=gr.getAllFonts();
            for (int i=0;i<f.length;i++)
                  ta.append(f[i].getFontName()+"\n");
      }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Ce de mai font-uri !!! </TITLE>
</HEAD>
<BODY>
<APPLET CODE="app3.class" WIDTH=300   HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

Exemplu 4: Setează un font şi îi tipăreşte caracteristicile.

```
import javax.swing.JApplet;
import java.awt.Graphics;
import java.awt.Font;
```

```
public class app6 extends JApplet {
      private Font fon;

      public void init()
      {
        fon=new Font("Courier", Font.ITALIC + Font.BOLD, 24);
      }
      public void paint (Graphics g)
      {
            int stil, dim;
            String str, nume;

            g.setFont (fon);
            stil = fon.getStyle();

            if ( stil == Font.PLAIN)
                  str = "Plin";
            else if (stil == Font.BOLD)
                  str = "Bold";
            else if (stil == Font.ITALIC)
                  str = "Italic";
            else
                  str = "Bold italic";

            dim = fon.getSize();
            str += dim + " puncte ";
            nume = fon.getName();
            str += nume;

            g.drawString (str, 20, 40);
            g.drawString ("Familia de font-uri este " +
fon.getFamily(), 20, 60);

      }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Ce font frumos !!! </TITLE>
</HEAD>
<BODY>
<APPLET CODE="app6.class" WIDTH=700   HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

Exemplu 5: Desenează un poligon şi îl copiază într-o altă zonă a ferestrei.

```
import javax.swing.JApplet;
import java.awt.*;

public class app7 extends JApplet {
      int xValues[] = {20, 40, 50, 30, 20, 15, 20};
      int yValues[] = {20, 20, 30, 50, 50, 30, 20};

      private Polygon p2;

      public void init ()
      {
            p2 = new Polygon();
            p2.addPoint (70, 70);
            p2.addPoint (90, 70);
            p2.addPoint (100, 80);
            p2.addPoint (80, 100);
            p2.addPoint (70, 100);
            p2.addPoint (65, 80);
            p2.addPoint (60, 60);
      }
      public void paint (Graphics g)
      {
            //deseneaza conturul unui poligon
            g.drawPolygon (xValues, yValues, 7);
            //deseneaza un poligon
            g.fillPolygon (p2);
            //copiaza cele doua poligoane la noile coordonate
            g.copyArea (0, 0, 100, 100, 10,10 );
      }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Ce poligoane... </TITLE>
</HEAD>
<BODY>
<APPLET CODE="app7.class" WIDTH=300   HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

Exemplu 6: Desenează un poligon şi îl copiază într-o altă zonă a ferestrei.

```
import javax.swing.JApplet;
```

```
import java.awt.*;

public class app7 extends JApplet {
     int xValues[] = {20, 40, 50, 30, 20, 15, 20};
     int yValues[] = {20, 20, 30, 50, 50, 30, 20};

     private Polygon p2;

     public void init ()
     {
          p2 = new Polygon();
          p2.addPoint (70, 70);
          p2.addPoint (90, 70);
          p2.addPoint (100, 80);
          p2.addPoint (80, 100);
          p2.addPoint (70, 100);
          p2.addPoint (65, 80);
          p2.addPoint (60, 60);
     }
     public void paint (Graphics g)
     {
          //deseneaza conturul unui poligon
          g.drawPolygon (xValues, yValues, 7);
          //deseneaza un poligon
          g.fillPolygon (p2);
          //copiaza cele doua poligoane la noile coordonate
          g.copyArea (0, 0, 100, 100, 10,10 );
     }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Ce poligoane... </TITLE>
</HEAD>
<BODY>
<APPLET CODE="app7.class" WIDTH=300   HEIGHT=200>
</APPLET>
</BODY>
</HTML>
```

Exemplu 7: Suma a două numere folosind preluarea parametrilor.

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

public class suma extends JApplet
{
```

```
  int m,n;
  String s;

  public void init(){
    String sm=getParameter("m"),
           sn=getParameter("n");
    m=Integer.parseInt(sm);
    n=Integer.parseInt(sn);
    s=new Integer(m+n).toString();
  }

  public void paint(Graphics g){
    g.drawString("Cmmdc = "+s, 50, 60 );
  }
}
```

Documentul .html este:

```
<HTML>
<HEAD>
<TITLE> Suma... </TITLE>
</HEAD>
<BODY>
<APPLET CODE="suma.class" WIDTH=300   HEIGHT=200>
          <PARAM name="m" value="2">
          <PARAM name="n" value="3">
</APPLET>
</BODY>
</HTML>
```