

Appleturi

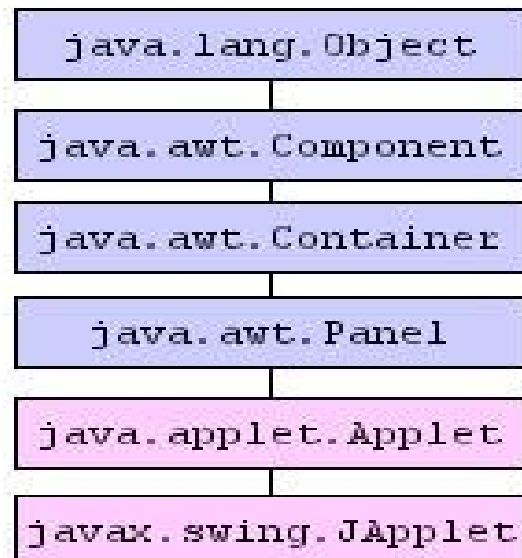
- Ce este un applet ?
- Crearea unui applet simplu
- Ciclul de viață
- Interfața grafică cu utilizatorul
- Definirea și folosirea parametrilor
- Tag-ul APPLET
- Folosirea firelor de execuție
- Alte metode oferite de clasa Applet
- Arhivarea appleturilor
- Restricții de securitate
- Appleturi care sunt și aplicații

Ce este un applet ?

Un *applet* reprezintă un program Java ce gestionează o suprafață de afișare (container) ce poate fi inclusă într-o pagină Web.

Pachete: **java.applet**

Clasa principală extinde **Applet**



Un applet nu poate fi executat independent

El este executat de către **browser**-ul în care a fost încărcată pagina Web ce conține appletul sau de către programe specializate.

Browsere

- Internet Explorer
- Netscape
- Mozilla
- Opera

Programe specializate - **appletviewer**
Utilitar inclus în distribuția J2SDK.

Crearea unui applet simplu

1. Scrierea codului sursa

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

public class FirstApplet extends Applet {
    Image img;
    public void init() {
        img = getImage(getCodeBase(), "taz.gif");
    }
    public void paint (Graphics g) {
        g.drawImage(img, 0, 0, this);
        g.drawOval(100,0,150,50);
        g.drawString(
            "Hello! My name is Taz!", 110, 25);
    }
}
```

2. Salvarea fisierelor sursă FirstApplet.java.

3. Compilarea

```
javac FirstApplet.java
```

Va fi generat: `FirstApplet.class`

4. Rularea appletului

- **Crearea unui fișier HTML**

```
<html>
<head>
  <title>Primul applet Java</title>
</head>
<body>
  <applet code=FirstApplet.class
          width=400 height=400>
  </applet>
</body>
</html>
```

- **Vizualizarea appletului**

`appletviewer simplu.html` sau folosind un browser.

Ciclul de viață

- **Incărcarea în memorie**
Este instanțiată clasa principală
- **Inițializarea**
Este apelată metoda `init`
- **Pornirea**
Este apelată metoda `start`
- **Execuția propriu-zisă**
- **Oprirea temporară**
Este apelată metoda `stop` a acestuia
- **Oprirea definitivă**
Este apelată metoda `destroy`

Structura generală

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

public class StructuraApplet extends Applet {

    public void init() {
    }
    public void start() {
    }
    public void stop() {
    }
    public void destroy() {
    }
}
```

Aceste metode sunt **apelate automat de browser** și nu trebuie apelate explicit din program !

Interfața grafică cu utilizatorul

Folosind componente

Plasarea componentelor, gestionarea poziționării lor și tratarea evenimentelor generate se realizează la fel ca și în cazul aplicațiilor.

Uzual, operații executate în `init`.

Gestionar de poziționare: `FlowLayout`

Folosind desenarea

```
public void paint(Graphics g) {  
    // Desenare  
    ...  
}
```

Evenimentele tratate sunt cele generate de mouse sau tastatură.

Definirea și folosirea parametrilor

Permit **personalizarea aspectului sau comportării** unui applet fără a-i schimba codul.

Definirea

```
<APPLET CODE="TestParametri.class" WIDTH=100 HEIGHT=50
  <PARAM NAME=textAfisat VALUE="Salut">
  <PARAM NAME=numeFont VALUE="Times New Roman">
  <PARAM NAME=dimFont VALUE=20>
</APPLET>
```

Folosirea - `getParameter`

”Documentarea” parametrilor `getParameterInfo` returnează un vector de triplete: (*numele*, *tipul*, *descriere*).

Listing 1: Folosirea parametrilor

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

public class TestParametri extends Applet {

    String text, numeFont;
    int dimFont;

    public void init() {
        text = getParameter("textAfisat");
        if (text == null)
            text = "Hello"; // valoare implicita

        numeFont = getParameter("numeFont");
        if (numeFont == null)
            numeFont = "Arial";

        try {
            dimFont = Integer.parseInt(getParameter("dimFont"));
        } catch (NumberFormatException e) {
            dimFont = 16;
        }
    }

    public void paint(Graphics g) {
        g.setFont(new Font(numeFont, Font.BOLD, dimFont));
        g.drawString(text, 20, 20);
    }

    public String [][] getParameterInfo() {
        String [][] info = {
            // Nume      Tip          Descriere
            {"textAfisat", "String", "Sirul ce va fi afisat"},
            {"numeFont", "String", "Numele fontului"},
            {"dimFont", "int", "Dimensiunea fontului"}
        };
        return info;
    }
}
```

Tag-ul APPLET

```
<APPLET
  CODE = clasaApplet
  WIDTH = latimeInPixeli
  HEIGHT = inaltimeInPixeli

  [ARCHIVE = arhiva.jar]
  [CODEBASE = URLApplet]
  [ALT = textAlternativ]
  [NAME = numeInstantaApplet]
  [ALIGN = aliniere]
  [VSPACE = spatiuVertical]
  [HSPACE = spatiuOrizontal] >

  [< PARAM NAME = parametru1 VALUE = valoarea1 >]
  [< PARAM NAME = parametru2 VALUE = valoarea2 >]
  ...
  [text HTML alternativ]

</APPLET>
```

Folosirea firelor de execuție

Fiecărui applet îi este creat automat un fir de execuție responsabil cu apelarea metodelor acestuia. Acestea vor rula concurent.

Dpdv GUI, fiecare applet are acces la un **același** fir de execuție, creat de asemenea automat, responsabil cu desenarea și cu transmiterea mesajelor generate de către componente.

Operațiunile consumatoare de timp este recomandat să le realizăm într-un alt fir de execuție

Listing 2: Incorect: blocarea metodei paint

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

public class AppletRau1 extends Applet {
    public void paint(Graphics g) {
        while(true) {
            int x = (int)(Math.random() * getWidth());
            int y = (int)(Math.random() * getHeight());
            g.drawString("Hello", x, y);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
        }
    }
}
```

Listing 3: Incorect: appletul nu termină inițializarea

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

public class AppletRau2 extends Applet {
    int x, y;

    public void init() {
        while(true) {
            x = (int)(Math.random() * getWidth());
            y = (int)(Math.random() * getHeight());
            repaint();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
        }
    }
    public void paint(Graphics g) {
        g.drawString("Hello", x, y);
    }
}
```

Listing 4: Corect: folosirea unui fir de execuție propriu

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

public class AppletCorect1 extends Applet implements Runnable
{
    int x, y;
    Thread fir = null;

    public void init() {
        if (fir == null) {
            fir = new Thread(this);
            fir.start();
        }
    }

    public void run() {
        while(true) {
            x = (int)(Math.random() * getWidth());
            y = (int)(Math.random() * getHeight());
            repaint();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
        }
    }

    public void paint(Graphics g) {
        g.drawString("Hello", x, y);
    }
}
```

Listing 5: Folosirea metodelor start și stop

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

public class AppletCorect2 extends Applet implements Runnable
{
    int x, y;
    Thread fir = null;
    boolean activ = false;
    int n = 0;

    public void start() {
        if (fir == null) {
            fir = new Thread(this);
            activ = true;
            fir.start();
        }
    }

    public void stop() {
        activ = false;
        fir = null;
    }

    public void run() {
        while(activ) {
            x = (int)(Math.random() * getWidth());
            y = (int)(Math.random() * getHeight());
            n ++;
            repaint();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
        }
    }

    public void paint(Graphics g) {
        g.drawString("Hello " + n, x, y);
    }
}
```

Alte metode din clasa Applet

getAppletInfo

```
public String getAppletInfo() {  
    return  
        "Applet simplist, autor necunoscut, ver 1.0";  
}
```

Aflarea adreselor URL

- getCodeBase
- getDocumentBase

Afişarea unor mesaje

```
public void init() {  
    showStatus("Initializare applet...");  
}
```

Afișarea imaginilor

- Folosind o suprafață de desenare (**Canvas**) sau
- Desenând direct pe suprafața appletului

Obținerea unei referințe la o imagine: `getImage` din clasa `Applet`

Listing 6: Afișarea imaginilor

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

public class Imagini extends Applet {
    Image img = null;

    public void init() {
        img = getImage(getCodeBase(), "taz.gif");
    }

    public void paint(Graphics g) {
        g.drawImage(img, 0, 0, this);
    }
}
```

Aflarea contextului de execuție

Contextul de execuție = pagina în care rulează appletul.

```
AppletContext contex =  
    getAppletContext();
```

Afișarea unor documente în browser

```
try {  
    URL doc = new URL("http://www.infoiasi.ro");  
    getAppletContext().showDocument(doc);  
} catch(MalformedURLException e) {  
    System.err.println("URL invalid! \n" + e);  
}
```

Comunicarea cu alte applet-uri

- Identificarea unui applet aflat pe aceeași pagina

getApplet, getApplets

- Apelarea unei metode sau setarea unei variabile publice a acestuia

Redarea sunetelor - în format **.au**

- Folosind metoda **play** din clasa **Applet**
- Folosind un obiect de tip

AudioClip: start, loop, stop

Listing 7: Redarea sunetelor

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

public class Sunete extends Applet implements ActionListener{
    Button play = new Button("Play");
    Button loop = new Button("Loop");
    Button stop = new Button("Stop");
    AudioClip clip = null;

    public void init() {
        // Fisierul cu sunetul trebuie sa fie in acelasi
        // director cu appletul
        clip = getAudioClip(getCodeBase(), "sunet.au");
        add(play);
        add(loop);
        add(stop);

        play.addActionListener(this);
        loop.addActionListener(this);
        stop.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        Object src = e.getSource();
        if (src == play)
            clip.play();
        else if (src == loop)
            clip.loop();
        else if (src == stop)
            clip.stop();
    }
}
```

Arhivarea appleturilor

Reprezintă cea mai **eficientă** modalitate de a distribui un applet.

Arhivarea fișierelor

```
jar cvf arhiva.jar ClasaPrincipala.class  
                AltaClasa.class  
                imagine.jpg  
                sunet.au
```

// sau

```
jar cvf arhiva.jar *.class *.jpg *.au
```

Includerea unui applet arhivat într-o pagină Web

```
<applet archive=arhiva.jar  
        code=ClasaPrincipala  
        width=400 height=200 />
```

Restricții de securitate

Browser-ul instalează un manager de securitate (**SecurityManager**) care va ”superviza” activitatea appletului, aruncând excepții de tip **SecurityException** pentru operații nepermise.

Un applet **nu poate să:**

- Citească sau să scrie fișiere pe calculatorul pe care a fost încărcat (client).
- Deschidă conexiuni cu alte mașini în afară de cea de pe care provine (host).
- Pornească programe pe mașina client.
- Citească diverse proprietăți ale sistemului de operare al clientului.

Semnarea appleturilor

- **Arhivarea:**

```
> jar cf myjar.jar MyClass.class
```

- **Crearea unui certificat cu keytool:**

```
> keytool -genkey  
-alias alias  
-keystore .keystore
```

- **Semnarea cu jarsigner:**

```
> jarsigner -keystore .keystore  
-storepass password  
myjar.jar alias
```

Permisuni

Crearea unui fisier de permisiuni (policy file) se face cu utilitarul `policytool`:

- `CodeBase=URL`
- `SignedBy`

Fisierul rezultat va fi de tipul:

```
grant codeBase "file:///d:/java/curs/20/ex/" {  
    permission java.io.FilePermission "test.txt", "write";  
};
```

Folosirea unui fisier de permisiuni

Implicit managerul de securitate instalat va folosi fisierele de securitate specificate in:

```
java.home/lib/security/java.security.
```

Acestea sunt:

```
policy.url.1=file:${java.home}/lib/security/java.policy
```

```
policy.url.2=file:${user.home}/.java.policy
```

```
java.home=c:/jdk1.5.0/jre
```

```
user.home=c:/windows
```

Varianta 1

```
appletviewer -J-Djava.security.policy=mypolicy test.html
```

Varianta 2

```
policy.url.3=file:d:/java/curs/20/ex/mypolicy
```

Appleturi care sunt și aplicații

- Adăugăm metoda `main` clasei care descrie appletul, în care vom face operațiunile următoare.
- Creăm o instanță a appletului și o adăugăm pe suprafața unei ferestre.
- Apelăm metodele `init` și `start`, care ar fi fost apelate automat de către browser.
- Facem fereastra vizibilă.

Listing 8: Applet și aplicație

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

public class AppletAplicatie extends Applet {

    public void init() {
        add(new Label("Applet si aplicatie"));
    }

    public static void main(String args[]) {
        AppletAplicatie applet = new AppletAplicatie();
        Frame f = new Frame("Applet si aplicatie");
        f.setSize(200, 200);

        f.add(applet, BorderLayout.CENTER);
        applet.init();
        applet.start();

        f.show();
    }
}
```
