Curs 10
Applet-uri

Ce este un applet ?

Definitie

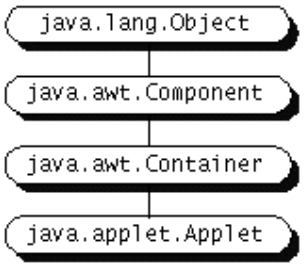
> Un *applet* reprezinta o suprafata de afisare (container) ce poate fi inclusa într-o pagina Web si gestionata printr-un program Java. Un astfel de program se mai numeste *miniaplicatie* sau, prin abuz de limbaj, *applet*.

Codul unui applet poate fi format din una sau mai multe clase. Una dintre acestea este *principala* si extinde clasa **Applet**, fiind clasa ce trebuie specificata în documentul HTML ce descrie pagina de Web în care dorim sa includem appletul.
Diferenta fundamentala dintre un applet si o aplicatie consta în faptul ca, un applet nu poate fi executat independent, ci va fi executat de browserul în care este încarcata pagina Web ce contine appletul respectiv. O aplicatie independenta este executata prin apelul interpretorului *java*, având ca parametru numele clasei principale a aplicatiei, clasa principala fiind cea care contine metoda *main*. Ciclul de viata al unui applet este complet diferit, fiind dictat de evenimentele generate de catre browser la vizualizarea documentului HTML ce contine appletul.

Pachetul care ofera suport pentru creearea de appleturi este **java.applet**.

Crearea unui applet

Orice applet este implementat prin crearea unei subclase a clasei **Applet**. Ierarhia claselor din care deriva `Applet` este prezentata în figura de mai jos:



Fiind derivata din clasa `Container`, clasa `Applet` descrie de fapt suprafete de afisare, asemenea claselor `Frame` sau `Panel`.

Curs 10

Un applet simplu

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

public class AppletSimplu extends Applet {
        public void paint(Graphics g) {
                g.setFont(new Font("Arial", Font.BOLD, 16));
                g.drawString("Hello", 0, 30);
        }
}
```

Uzual, clasa principala va fi salvata într-un fisier cu acelasi nume si extensia `.java`. Asadar, vo salva clasa de mai sus într-un fisier `AppletSimplu.java`.

Compilarea

Compilarea se face la fel ca si la aplicatiile independente, apelând compilatorul **javac** pentru clasa principala a appletului (cea care extinde `Applet`).

```
javac AppletSimplu.java
```

In cazul în care compilarea a reusit va fi generat fisierul `AppletSimplu.class`.

Executia (vizualizarea)

Pentru a vizualiza acest applet trebuie sa cream un document HTML, sa-i spunem `demo.html`, în care sa specificam cel putin urmatoarele informatii

- clasa ce contine codul appletului
- latimea si înaltimea suprafetei alocate pe pagina Web

```
<HTML>
<HEAD>
<TITLE> Un applet simplu </TITLE>
</HEAD>
<APPLET CODE="AppletSimplu.class" WIDTH=100 HEIGHT=50></APPLET>
</HTML>
```

Vizualizarea acestui document se poate face cu orice browser (Internet Explorer, Netscape, etc), sau cu utilitarul `appletviewer` ce vine în pachetul JDK.

```
appletviewer demo.html
```

Ciclul de viata al unui applet

Executia unui applet începe în momentul în care un browser afiseaza o pagina Web în care este inclus appletul respectiv si poate trece prin mai multe etape. Fiecare etapa este strâns legata de un eveniment generat de catre browser si determina apelarea unei metode specifice din clasa ce implementeaza appletul.

1. **Incarcarea in memorie**
   Este creata o instanta a clasei principale a appletului si încarcata în memorie.
2. **Initializarea**
   Este apelata metoda `init` ce permite initializarea diverselor variabile, citirea unor parametri de intrare, etc.
3. **Pornirea**
   Este apelata metoda `start`

4. **Executia propriu-zisa**
   Consta în interactiunea dintre utilizator si componentele afisate pe suprafata appletului sau în executarea unui anumit cod într-un fir de executie. In unele situatii întreaga executie a appletului se consuma la etapele de initializare si pornire.
5. **Oprirea temporara**
   In cazul în care utilizatorul paraseste pagina Web în care ruleaza appletul este apelata metoda `stop` a acestuia, dându-i astfel posibilitatea sa se opreasca temporar cât timp nu este vizibil, pentru a nu consuma inutil din timpul procesorului. Acelasi lucru se întâmpla daca fereastra browserului este minimizata. In momentul când pagina Web ce contine appletul devine din nou activa, va fi reapelata metoda `start`.
6. **Oprirea definitiva**
   La închiderea tuturor instantelor browserului folosit pentru vizualizare, appletul va fi eliminat din memorie si va fi apelata metoda `destroy` a acestuia, pentru a-i permite sa elibereze resursele detinute. Apelul metodei `destroy` este întotdeauna precedat de apelul lui `stop`.

Metodele specifice appleturilor

Asadar, exista metode specifice appletului ce sunt apelate automat la diverse evenimente generate de catre browser. Acestea sunt date în tabelul de mai jos:

Metoda	Situatia în care este apelata
init	La initializarea appletului; teoretic, aceasta metoda ar trebui sa se apeleze o singura data, la prima afisare a appletului în pagina, însa, la unele browsere, este posibil ca ea sa se apeleze de mai multe ori.
start	Imediat dupa initializare si de fiecare data când appletul redevine activ, dupa o oprire temporara.
stop	De fiecare data când appletul nu mai este vizibil (pagina Web nu mai este vizibila, fereastra browserului este minimizata, etc) si înainte de `destroy`.
destroy	La închiderea ultimei instante a browserului care a încarcat în memorie clasa principala a appletului.
Aceste metode sunt apelate automat de browser si nu trebuie apelate explicit din program !	

Structura generala a unui applet

```
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() {
        }

        public void paint(Graphics g) {
        }
```

```
}
```

Interfata grafica cu utilizatorul

Dupa cum am vazut, clasa `Applet` este o extensie a superclasei `Container`, ceea ce înseamna ca appleturile sunt, înainte de toate, suprafete de afisare. Plasarea componentelor, gestionarea pozitionarii lor si tratarea evenimentelor generate se realizeaza la fel ca si în cazul aplicatiilor. Uzual, adaugarea componentelor pe suprafata appletului precum si stabilirea obiectelor responsabile cu tratarea evenimentelor generate sunt operatiuni ce vor fi realizate în metoda `init`.
Gestionarul de pozitionare implicit este `FlowLayout`, însa acesta poate fi schimbat prin metoda `setLayout`.

Desenarea pe suprafata unui applet

Exista o categorie întreaga de appleturi ce nu comunica cu utilizatorul prin intermediul componentelor ci, executia lor se rezuma la diverse operatiuni de desenare executate în metoda **paint**. Reamintim ca metoda `paint` este responsabila cu definirea aspectului grafic al oricarei componente. Implicit, metoda `paint` din clasa `Applet` nu realizeaza nimic, deci, în cazul în care dorim sa desenam direct pe suprafata unui applet va fi nevoie sa supradefinim aceasta metoda.
In cazul în care este aleasa aceasta solutie, evenimentele tratate uzual vor fi cele generate de mouse sau tastatura.

Definirea si folosirea parametrilor

Parametrii sunt pentru appleturi ceea ce argumentele de la linia de comanda sunt pentru aplicatiile independente. Ei permit utilizatorului sa personalizeze aspectul sau comportarea unui applet fara a-i schimba codul si recompila clasele.

Definirea parametrilor se face în cadrul tagului `APPLET` din documentul HTML ce contine appletul si sunt identificati prin atributul **PARAM**. Fiecare parametru are un nume, specificat prin **NAME** si o valoare, specificata prin **VALUE**, ca în exemplul de mai jos:

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

Ca si în cazul argumentelor trimise aplicatiilor de la linia de comanda, tipul parametrilor este **sir de caractere**, indiferent daca valoarea este între ghilimele sau nu.

Fiecare applet are si un set de parametri prestabiliti ale caror nume nu vor putea fi folosite pentru definirea de noi parametri folosind metoda de mai sus. Acestia apar direct în corpul tagului `APPLET` si definesc informatii generale despre applet. Exemple de astfel de parametri sun `CODE`, `WIDTH` sau `HEIGHT`. Lista lor completa va fi prezentata la descrierea tagului `APPLET` . fi pre

Folosirea parametrilor primiti de catre un applet se face prin intermediul metodei **getParameter** care primeste ca argument numele unui parametru si returneaza valoarea acestuia. In cazul în care nu exista nici un parametru cu numele specificat, metoda întoarce `null`, caz în care programul trebuie sa atribuie o valaore implicita variabilei în care se dorea citirea respectivului parametru.
Sa rescriem apletul considerat initial (`AppletSimplu`) astfel încât acesta sa afiseze textul primit ca parametru, folosind un font cu numele si dimeniunea specificate de asemenea ca parametri.

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

public class AppletSimplu 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);
        }
}
```

Orice applet poate pune la dispozitie o "documentatie" referitoare la parametrii pe care îi suporta, pentru a veni în ajutorul utilizatorilor care doresc sa includa appletul într-o pagina Web. Aceasta se realizeaza prin supradefinirea metodei **getParameterInfo**, care returneaza un vector format din triplete de siruri. Fiecare element al vectorului este de fapt un vector cu trei elemente de tip `String`, cele trei siruri reprezentând *numele* parametrului, *tipul* sau si o *descriere* a sa.

```
        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;
        }
```

Informatiile furnizate de un applet pot fi citite din browserul folosit pentru vizualizare prin metode specifice acestuia. De exemplu, în Netscape se foloseste optiunea *Page info* din meniul *View*.

Tag-ul <APPLET>

```
        < APPLET
                [CODEBASE = directorApplet]
                CODE = clasaApplet
                [ALT = textAlternativ]
                [NAME = numeInstantaApplet]
                WIDTH = latimeInPixeli
                HEIGHT = înaltimeInPixeli
                [ALIGN = aliniere]
                [VSPACE = spatiuVertical]
                [HSPACE = spatiuOrizontal]
        >
        [< PARAM NAME = numeParametru1 VALUE = valoare1 >]
        [< PARAM NAME = numeParametru2 VALUE = valoare2 >]
        . . .
```

```
        [text HTML alternativ]
        </APPLET>
```

Atributele puse între paranteze patrate sun optionale.

CODEBASE = *directorApplet*
: Specifica URL-ul în care se gaseste clasa appletului. Uzual se exprima relativ la directorul documentului HTML. In cazul în care lipseste, se considera implicit URL-ul documentului.

CODE = *clasaApplet*
: Numele fisierului ce contine clasa principala a appletului. Acesta va fi cautat în directorul specificat de `CODEBASE`. Nu poate fi absolut.

ALT = *textAlternativ*
: Specifica textul ce trebuie afisat daca browserul întelege tagul `APPLET` dar nu poate rula appleturi Java.

NAME = *numeInstantaApplet*
: Ofera posibilitatea de a da un nume respectivei instante a appletului, astfel încât mai multe appleturi aflate pe aceeasi pagina sa comunice între ele folosindu-se de numele lor.

WIDTH = *latimeInPixeli*

HEIGHT = *înaltimeInPixeli*
: Specifica latimea si înaltimea suprafetei în care va fi afisat appletul.

ALIGN = *aliniere*
: Semnifica modalitatea de aliniere a appletului în pagina Web. Acest atribut poate primi una din urmatoarele valori: `left, right, top, texttop, middle, absmiddle, baseline, bottom, absbottom` , seminificatiile lor fiind aceleasi ca si la tagul `IMG`.

VSPACE = *spatiuVertical*

HSPACE = *spatiuOrizontal*
: Specifica numarul de pixeli dintre applet si marginile suprafetei de afisare.

< PARAM NAME = *numeParametru1* **VALUE** = *valoare1* **>**
: Tag-urile <PARAM> sunt folosite pentru specificarea parametrilor unui applet.
vezi "Definirea si folosirea parametrilor"

text HTML alternativ
: Este textul ce va fi afisat în cazul în care browserul nu întelege tagul `APPLET`. Browserele care înteleg Java vor ignora acest text.

Folosirea firelor de executie în appleturi

Fiecare applet aflat pe o pagina Web se executa într-un fir de executie propriu. Acesta este creat de catre browser si este responsabil cu desenarea appletului (apelul metodelor `update` si `paint`) precum si cu transmiterea mesajelor generate de catre componentele appletului. In cazul în care dorim sa realizam si alte operatiuni consumatoare de timp este recomandat sa le realizam într-un alt fir de executie, pentru a nu bloca interactiunea utilizatorului cu appletul sau redesenarea acestuia.

Structura unui applet care doreste sa lanseze un fir de executie poate avea doua forme. In prima situatie appletul porneste un fir de executie la initialzarea sa iar acesta va rula, indiferent daca appletul mai este sau nu vizibil, pâna la oprirea sa naturala (terminarea metodei `run`)

```
import java.applet.Applet;

class AppletThread1 extends Applet implements Runnable {
        Thread appletThread = null;

        public void init() {
                if (appletThread == null) {
                        appletThread = new Thread(this);
```

```
                        appletThread.start();
                }
        }
        public void run() {
                //codul firului de executie
        }
}
```

In cazul în care firul de executie pornit de applet efectueaza operatii ce au sens doar daca appletul este vizibil, cum ar fi animatie, ar fi de dorit ca acesta sa se opreasca atunci când appletul nu mai este vizibil (la apelul metodei `stop`) si sa reporneasca atunci când appletul redevine vizibil (la apelul metodei `start`).

```
import java.applet.Applet;

public class StructuraApplet extends Applet implements Runnable {
        Thread appletThread = null;
        boolean running = false;

        public void start() {
                //reporneste firul de executie
                if (appletThread == null) {
                        appletThread = new Thread(this);
                        running = true;
                        appletThread.start();
                }
        }

        public void stop() {
                //opreste firul de executie
                running = false;
                appletThread = null;
        }

        public void run() {
                while (running) {
                        //codul firului de executie
                }
        }
}
```

Un applet este considerat *activ* imediat dupa apelul metodei `start` si devine inactiv la apelul metodei `stop`.Pentru a afla daca un applet este activ se foloseste metoda **isActive**.

Alte metode oferite de clasa Applet

Clasa `Applet` ofera metode specifice applet-urilor pentru:

Punerea la dispozitie a unor informatii despre applet

Similara cu metoda `getParameterInfo` ce oferea o "documentatie" despre parametrii pe care îi suporta un applet, exista metoda **getAppletInfo** ce permite specificarea unor informatii legate de applet cum ar fi numele, autorul, versiunea, etc. Metoda returneaza un sir de caractere continând informatii despre applet.

```
        public String getAppletInfo() {
                return "Cel mai simplu applet, autor necunoscut, ver 1.0";
        }
```

Aflarea unor adrese URL referitoare la applet

Se realizeaza cu metodele:

- **getCodeBase** - ce returneaza URL-ul directorului ce contine clasa appletului
- **getDocumentBase** - returneaza URL-ul directorului ce contine documentul HTML în care este inclus appletul respectiv.

Sunt foarte utile deoarece permit specificarea relativa a fisierelor folosite de un applet.

Afisarea imaginilor

Afisarea imaginilor într-un applet se face fie prin intermediul unei componente ce permite acest lucru, cum ar fi o suprafata de desenare de tip `Canvas`, fie direct în metoda `paint` a applet-ului, folosind metoda `drawImage` a clasei `Graphics`. In ambele cazuri, încarcarea imaginii în memorie se va face cu ajutorul metodei **getImage** din clasa `Applet`. Aceasta poate primi ca argument fie adresa URL absoluta a fisierului ce contine imaginea, fie calea sa relativa la o anumita adresa URL, cum ar fi cea a directorului în care se gaseste documentul HTML ce contine appletul (`getDocumentBase`) sau a directorului în care se gaseste clasa appletului (`getCodeBase`).

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

public class AppletImagine extends Applet  {
        Image img = null;

        public void init() {
                img = getImage(getCodeBase(), "taz.gif");
        }
        public void paint(Graphics g) {
                g.drawImage(img, 0, 0, this);
        }
}
```

Afisarea unor mesaje în bara de stare a browserului

Acest lucru se realizeaza cu metoda **showStatus**

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

Aflarea contextului de executie

Contextul de executie al unui applet se refera la pagina în care acesta ruleaza si este descris de interfata **AppletContext**. Crearea unui obiect ce implementeaza aceasta interfata se realizeaza de catre browser, la apelul metodei **getAppletContext** a clasei `Applet`. Prin intermediul acestei interfete un applet poate "vedea" în jurul sau, putând comunica cu alte applet-uri aflate pe aceeasi pagina sau cere browser-ului sa deschida diverse documente.

```
                AppletContext env = getAppletContext();
```

Afisarea unor documente în browser

Se face cu metoda **showDocument** ce primeste adresa URL a fisierului ce contine documentul dorit (text, html, imagine, etc). Aceasta metoda se gaseste în interfata `AppletContext`.

```
                try {
                        URL doc = new URL("http://www.infoiasi.ro");
                        getAppletContext().showDocument(doc);
                } catch(MalformedURLException e) {}
```

Comunicarea cu alte applet-uri aflate pe aceeasi pagina

Aceasta comunicare implica de fapt identificarea unui applet aflat pe aceeasi pagina si apelarea unei metode sau setare unei variabile publice a acestuia. Identificarea se face prin intermediu numelui pe care orice instanta a unui applet îl poate specifica prin atributul **NAME**. (vezi tag-ul <APPLET>).
Obtinerea unei instante a unui applet al carui nume îl cunoastem sau obtinerea unei enumerari a tuturor applet-urilor din pagina se fac cu metodele definite de interfata `AppletContext` **getApplet** si **getApplets**.

Reproducerea unor sunete

Probleme de securitate

Un applet nu poate sa:

- Citeasca sau scrie fisiere pe calculatorul pe care a fost încarcat (client)
- Deschida conexiuni cu alte masini în afara de cea de pe care provine (host)
- Porneasca programe pe masina client
- Citeasca diverse proprietati ale sistemului de operare al clientului

Ferestrele folosite de un applet, altele decât cea a browserului, vor arata altfel decât într-o aplicatie obisnuita.

Programe care sunt atât appleturi cât si aplicatii

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

public class AppletApp extends Applet  {

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

        public static void main(String args[]) {

                AppletApp applet = new AppletApp();

                Frame f = new Frame("Aplicatie si applet");
                f.setSize(200, 200);

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

                f.show();
        }
}
```