Applet-uri, aplicaţii şi kit-ul Java

Cuprins

Dacă în capitolul anterior s-a realizat o introducere în tematica programării în Java, acum este timpul să trecem la treabă.
Mai precis să reamintim ce se poate realiza cu ajutorul programării în Java:

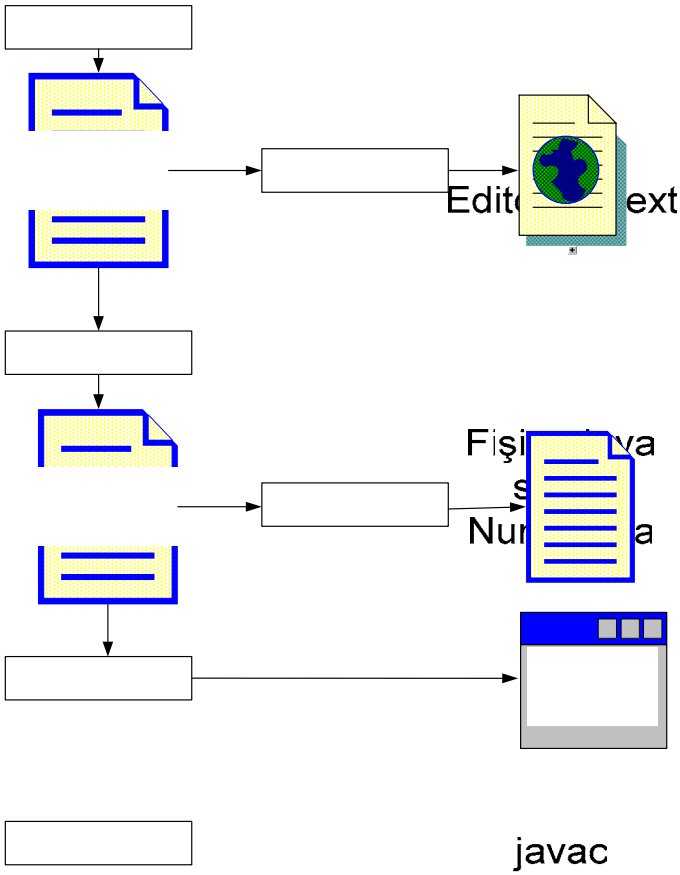
- Aplicaţii sau programe
- Applet-e
- Şi servlett-e

Se pare că v-am surprins cu ultimul tip de program de care nu s-a amintit până acum – servlett-ul. Acest tip de program este rulat de către server-ul WEB pentru implementarea unui protocol sau program de comunicaţie. În cadrul acestui curs nu ne vom referi la acest tip de program.
Să revenim la cele două tipuri de programe şi să identificăm diferenţele şi asemănările dintre ele.
Mai întâi încercând o diferenţiere sintetică vom putea schiţa următorul tabel:

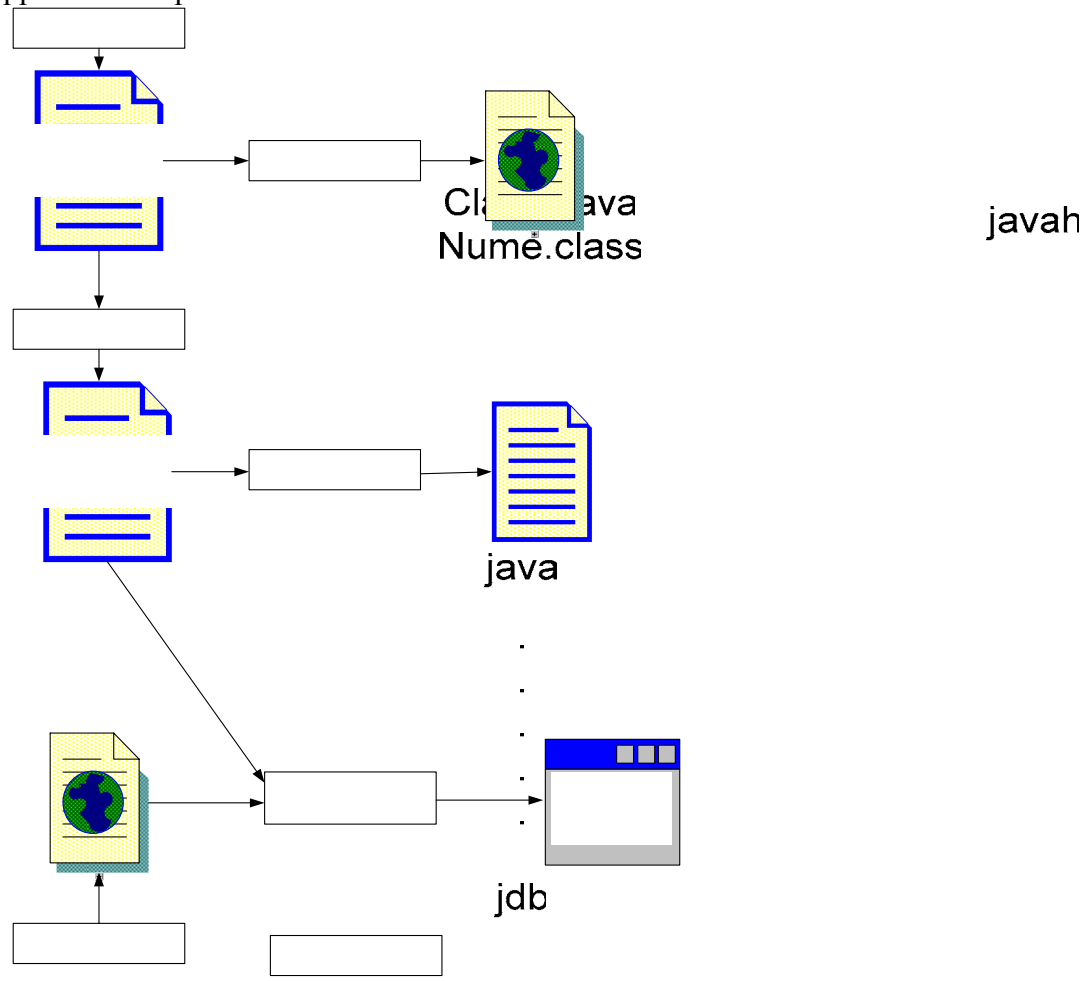
Atribute	Applet Java	Aplicaţie Java
Utilizează grafice	Este implicit o aplicaţie grafică	Opţional
Cerinţe referitoare la memorie	Cerinţele aplicaţiei Java plus cerinţele datorate browser-ului Web	Cerinţe minimale datorate aplicaţiei Java
Modul de distribuţie	Cuplat cu aplicaţia HTML, transportat prin protocolul HTTP	Încărcat ca fişier sistem sau de către un proces prin apelul unor clase Java
Mediul de intrare	Prin browser-ul client, specificând parametrii necesari server-ului pentru a localiza fişierul HTML apelat	Prin parametrii de tip argumente ale comenzii
Metode utilizate de către maşina virtuală – VM	Init – metoda de iniţializare Start – metoda de start program Stop – metoda de stop program Destroy – terminarea metodei Paint – metoda de desenare	Main – metoda de startup
Aplicaţii tipice	Sisteme publice de acces a clienţilor din Internet, prezentări online multimedia, animaţii Web page	Aplicaţii network server, multimedia, instrumente de dezvoltare aplicaţii, inzterfeţe utilizator şi aplicaţii pentru echipamente electrocasnice

Ca o concluzie a acestui tabel se remarcă faptul că applet-ul este legat prin excelența de Internet, de browser-e, în timp ce aplicațiile Java sunt programe independente de mediul Internet. Mai precis o aplicație Java distribuită pe Internet va fi complet funcțională, îm timp ce un applet Java va fi funcțional, numai dacă broswer-ul utilizat de client este unul care permite utilizarea appletelor Java.
Kit-ul Java conține următoarele elemente, a căror semnificație este indicată imediat:
javac – compilatorul de Java, care are rolul de a converti codul sursă în bytecodes. Mai precis transformă un fișier sursă cu extensia .java într-un fișier compilat cu extensia .class
java – interpreter-ul JIT (remember – **J**ust **I**n **T**ime) care execută bytecode-ul din fișierul class. Această comandă este aplicabilă **numai** aplicațiilor Java.
appletviewer – interpretor Java pentru execuția claselor applett-elor găzduite de paginile HTML
javadoc – crează documente HTML bazate pe sursele Java şi comentariile cuprinse în cadrul acestor surse
jdb – Debugger-ul Java. Permite rularea pas cu pas a programului, introducerea unor întreruperi de execuție şi examinarea unor variabile
javah – generează fişier Cce pot fi utilizate în cadrul unor rutine C sau construieşte rutine C ce pot fi apelate de programe Java
javap – Dezasamblor Java - afişează funcțiile accesibile şi datele dintr-o clasă complilată anterior
rmic – crează clase care permit Remote Method Invocation – RMI – accesul de la distanța a resurselor
rmiregistry - registru utilizat pentru a obține accesul la obiecte RMI situate pe o anumită maşină
serialver - utilitar de serializare, relativ la construcția unor obiecte
native2ascii – utilitar de conversie a caraterelor Latin-1 ,Unicode în alte structuri de caractere internaționale
jar – generatorul de arhivare a claselor ce vor fi distribuite cu aplicația Java
keytool – utilitar de generare a unor chei de securizare
jarsigner – utilitar de arhivare utilizând chei de securizare
policytool – permite configurarea politicii de securizare la nivelul utilizatorului

Pentru a obține o aplicație Java se procedează în felul următor:

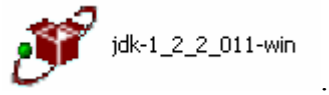


Pentru a obține un applet Java se procedează în felul următor:

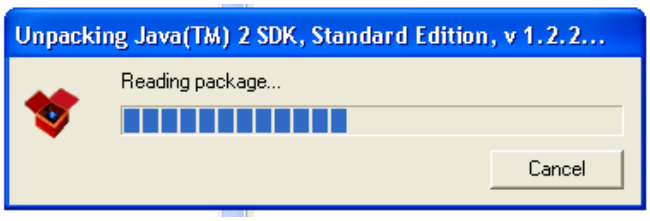


Instalarea Kit-ului de Java

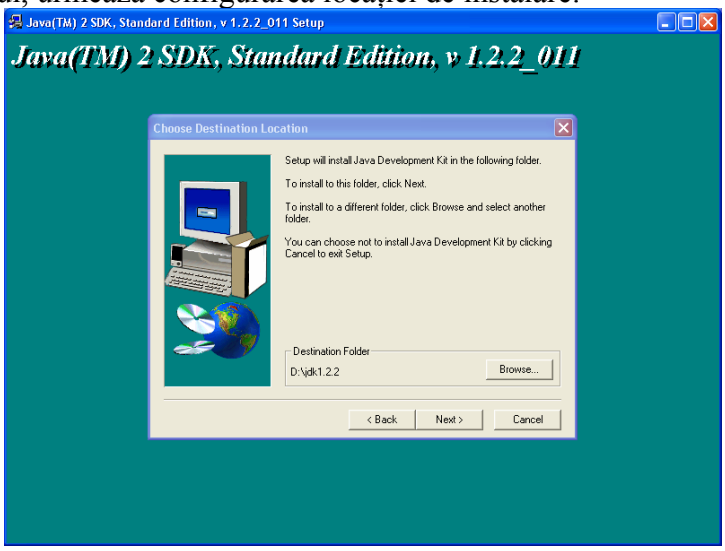
Fiserul care permite realizarea instalării pachetului Java poate fi încărcat de la adresa www.sun.com, prin FTP sau HTTP. Încărcarea acestui pachet Java 2 SDK, Standard Edition, v.1.2.2.011 durează în medie, la viteza rețelelor universitare, între o oră şi 30 de minute. Pachetul încărcat este un fişier self-extract, varianta pentru procesoarele Intel, sisteme de operare Windows, care poartă numele jdk-1_2_2_011-win.exe, având 19,3 Mb, respectiv următorul icon:



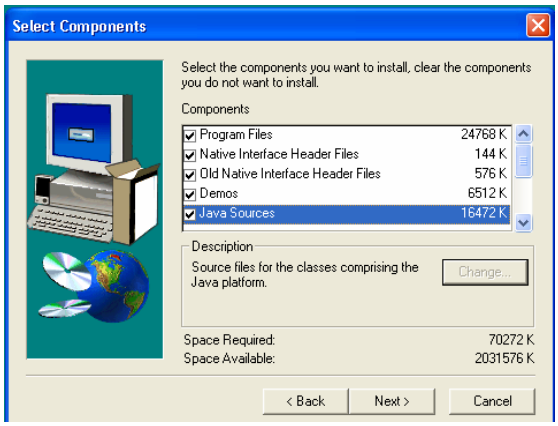
La accesarea acestui fişire, disponibil pe site-ul robotics.ucv.ro, buton Thesis&Papers, hyperlink-ul Java, va apare ecranul care realizează pregătirea instalării pachetului de bază, utilizând o aplicație de tipul self-extract:



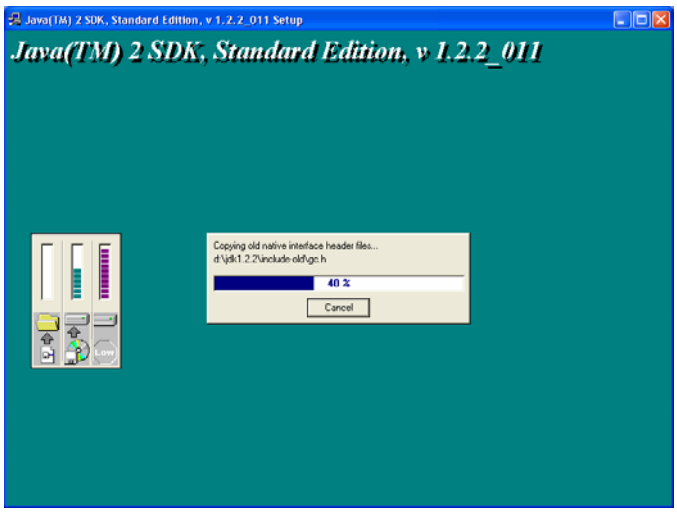
După citirea Agreement-ului legat de clauzele de folosire a acestui pachet şi exprimarea acceptului din partea utilizatorului, urmează configurarea locației de instalare:



După ce utilizatorul a indicat partiția şi directorul în care acesta doreşte ca aplicația să fie instalată, are loc selecția componentelor ce se doresc a fi instalate:

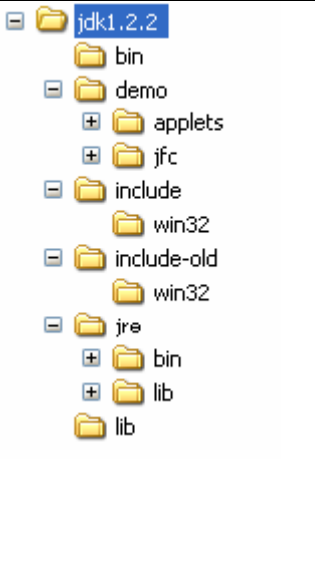


Odată ce a fost realizată această selecție se lansează în mod automat procesul de instalare:

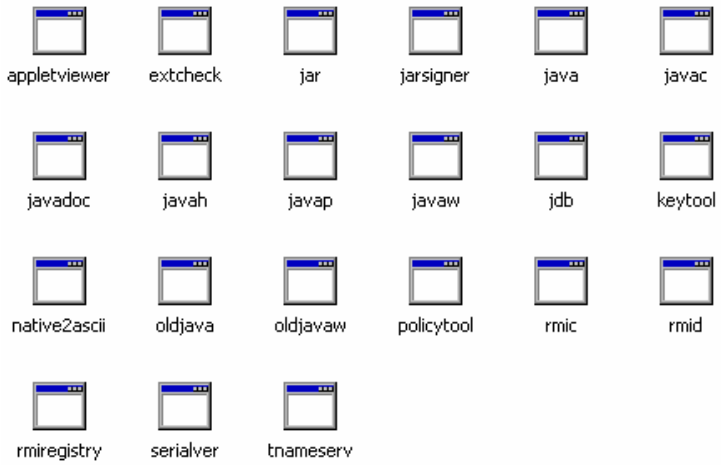


Încheierea acestui process de instalare este indicată de apariția indicatorului de încărcare 100% şi a ecranului în care apăsarea butonului de Finish conduce la încheierea instalării.

Structura directorilor , formată în urma procesului de instalare este următoarea:



iar aplicațiile specifice limbajului Java, care se află în directorul jdk1.2.2/bin, sunt:



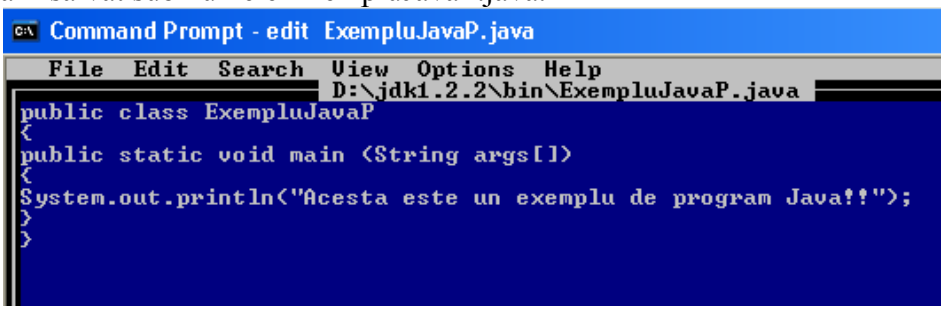
Realizarea unei aplicații Java

Desigur că în mintea majorității persoanelor care încep să studieze limbajul Java, aceştia asociază limbajul cu Internet-ul. Şi au dreptate în proporție de 80%. Dar limbajul Java permite şi realizarea unor aplicații de sine stătătoare, care pot fi rulate independent de existența unui browser.
Un exemplu deosebit de simplu, care va ilustra această afirmație, va fi descris în continuare.

Utilizând un editor de text de tip NotePad, WorPad sau mai sigur comanda edit din MsDos, se va introduce următorul program:

```
public class ExempluJavaP
{ public stativ void main (String args [])
        {System.out.println("Acesta este un exemplu de program Java!!");
        }
}
```

Acest fişier va fi salvat sub numele ExempluJavaP.java.



Acest cod conţine o singură metodă denumită *main()*. De fapt toate aplicaţiile Java vor trebui să definească această funcţie main de forma:

public static void main (String args [])

args este un şir, text de variabile de tip String.

Observaţie

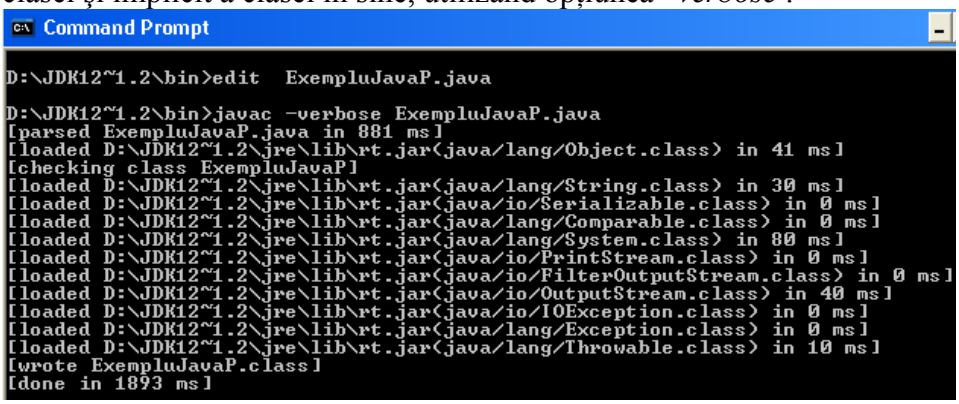
Numele fişierului sursă Java nu este arbitrar. Acesta trebuie să fie identic cu numele clasei care este definită a fi publică. Prin urmare, în cadrul unei aplicaţii Java numai o singură clasă poate fi publică, în tip ce altele, ce pot fi utilizate în structura programului vor fi non publice.

Javac

Compilarea acestei aplicaţii se realizează utilizând comanda ***javac ExempluJavaP.java***

```
D:\JDK12~1.2\bin>javac ExempluJavaP.java
```

Rezultatul acestei comenzi, daca editarea aplicaţiei a fost realizată fără erori, va fi fişierul de tip class ExempluJavaP.class. Deoarece în cadrul aplicaţiilor Java, componenta timp devine extrem de importantă, se poate verifica timpul în care au fost încărcate componentele utilizate la creearea clasei şi implicit a clasei în sine, utilizând opţiunea *–verbose* :



Odată cu creearea clasei, aplicaţia este disponibilă pentru a fi rulată pe maşina virtuală.

Java

Rularea unei aplicaţii Java se realizează utilizând comanda ***java ExemplujavaP***.

Atenţie

Nu construiţi comanda sub forma java ExemplujavaP.,java deoarece veţi avea erori pe care nu vi le veţi putea explica, de tipul :

```
D:\JDK12~1.2\bin>java ExempluJavaP.java
Exception in thread "main" java.lang.NoClassDefFoundError: ExempluJavaP/java
```

când de fapt explicația este că ați uitat că numai fişierele de tip opcode pot fi rulate, iar fițierele java pot fi compilare pentru a da naştere fişierelor opcode class.
Utilizarea argumentului –prof, în cadrul comenzii java : ***java –prof ExemplujavaP*** va conduce la creearea unui fişier java.prof care va indica cîte milisecunde sunt utilizate pentru apelul fiecărei metode sau funcții. Dacă utilizatorul are încărcate foarte multe clase şi metode, acest fişier va indica timpul de rulaj al tuturor claselor de pe respectiva maşină, fapt care va conduce la un fişier prof deosebit de mare şi din păcate prea puțin folositor!!
În prima parte a fişierului sunt indicate metodele apelate în ordinea descrescătoare a frecvenței de apelare, în partea a două este indicată dimensiunea memoriei utilizate, iar în partea a treia sunt listate tipul de variabile utilizate şi numarul de byți necesari pentru stocare. Acest fişier se adresează programatorilor avansați, care studiind fişierul pot duce la optimizarea timpului de execuție, prin realizarea unor noi algoritmi care vor reduce numărul de apeluri ale celor mai utilizate metode.

Javadoc

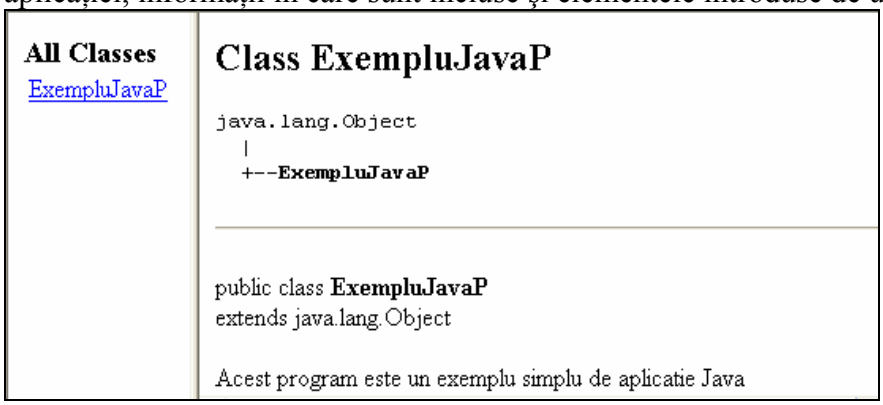
Adăugarea unor comentarii la codul Java, permite, prin folosirea comenzii javadoc să fie generat în mod automat o documentație HTML pentru aplicația construită.
Spre exemplu dacă se rescrie fişierul ExempluJavaP.java sub forma:

```
/** Acest program este un exemplu simplu de aplicatie Java*/
 public class ExempluJavaP
/** Aceasta metoda este apelata prima de catre interpretorul Java
* ducand la afisarea unui mesaj pe display*/
// comentariile obisnuite vor fi ignorate de catre comanda javadoc
{ public stativ void main (String args [])
/** Se face apel la functia main – obligatory pentru aplicatii Java*/
/* si acest tip de comentariu va fi ignorat de javadoc*/
        {System.out.println("Acesta este un exemplu de program Java!!");
        }
}
```

Este salvat fişierul şi apoi este apelată comanda ***javadoc ExempluJavaP.java***

```
D:\JDK12~1.2\bin>javadoc  ExempluJavaP.java
Loading source file ExempluJavaP.java...
Constructing Javadoc information...
Building tree for all the packages and classes...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating index.html...
Generating packages.html...
Generating ExempluJavaP.html...
Generating serialized-form.html...
Generating package-list...
Generating help-doc.html...
Generating stylesheet.css...
```

Rezultatul este un fişier html, index.html care oferă o serie de informații utile privind structura aplicației, informații în care sunt incluse şi elementele introduse de utilizator:



Jdb

O altă aplicație utilă este Java debugger-ul jdb. Acesta permite inserarea unor breakpoints în interiorul clasei şi implicit inspectarea unor elemente din cadrul aplicației. Utilizarea, pentru exemplul realizat, a acestei aplicații se realizează astfel:
Se încarcă ExempluJavaP.class în cadrul debugger-ului jdb: ***jdb ExempluJavaP***
Se inserează punctul de breakpoint la instrucțiunea main: ***stop in ExempluJavaP.main***
Se lansează rularea clasei cu breakpoint-ul inserat: ***run***
La atingerea punctului de stop se listează instrucțiunile rulate până în acel moment: ***list***
Se şterge punctul de stop: ***clear ExempluJavaP.main***
Se continuă rularea programului : ***cont***

```
D:\JDK12~1.2\bin>jdb ExempluJavaP
Initializing jdb...
0xb0:class(ExempluJavaP)
> stop in ExempluJavaP.main
Breakpoint set in ExempluJavaP.main
> run
run ExempluJavaP
running ...

Breakpoint hit: main[1] ExempluJavaP.main (ExempluJavaP:10)
main[1] list
6           { public static void main (String args[])
7           /** Se face apel la functia main - obligatorie pentru aplicatii Java*
/
8           /* si acest tip de comentariu va fi ignorat de javadoc */
9           {
10      => System.out.println("Acesta este un exemplu de program Java!!");
11          }
12          }
main[1] clear ExempluJavaP.main
Breakpoint cleared at ExempluJavaP.main
main[1] cont
Acesta este un exemplu de program Java!!
main[1]
Current thread "main" died. Execution continuing...
>
ExempluJavaP exited
```

Javap

Aplicația **javap** oferă posibilitatea examinării opcod-ului realizat la compilarea unei clase. Rapotul creeat va indica nu numai funcțiile şi variabilele utilizate, dar şi codul rezultat. Astfel lansarea comenzii ***javap ExempluJavaP*** va afişa metodele utilizate: main şi ExempluJavaP.

```
D:\JDK12~1.2\bin>javap ExempluJavaP
Compiled from ExempluJavaP.java
public class ExempluJavaP extends java.lang.Object {
    public ExempluJavaP();
    public static void main(java.lang.String[]);
}
```

Apelarea aceleiaşi comenzi cu argumentul –c va oferi şi imaginea programului prin opcod-ul fişierului:

```
D:\JDK12~1.2\bin>javap -c ExempluJavaP
Compiled from ExempluJavaP.java
public class ExempluJavaP extends java.lang.Object {
    public ExempluJavaP();
    public static void main(java.lang.String[]);
}

Method ExempluJavaP()
   0 aload_0
   1 invokespecial #6 <Method java.lang.Object()>
   4 return

Method void main(java.lang.String[])
   0 getstatic #7 <Field java.io.PrintStream out>
   3 ldc #1 <String "Acesta este un exemplu de program Java!!">
   5 invokevirtual #8 <Method void println(java.lang.String)>
   8 return
```

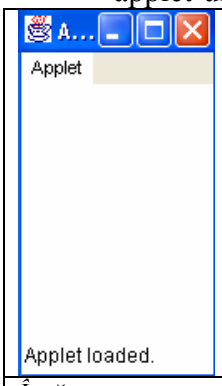
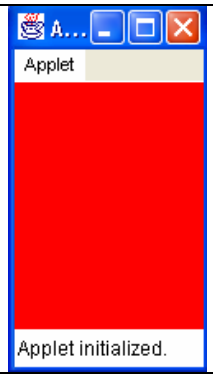
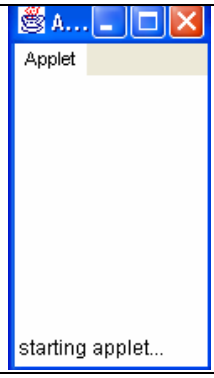
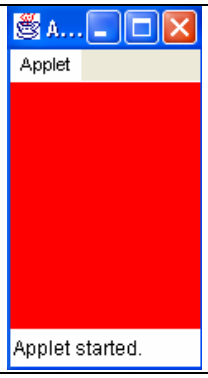
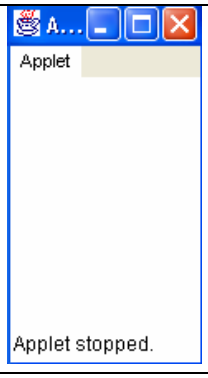
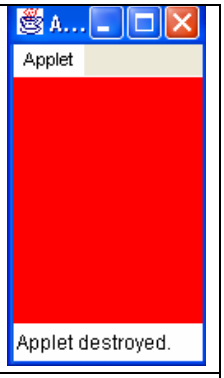
Realizarea unui applet Java

Crearea unui applet Java impune folosirea unui browser Internet şi prin urmare a unui fişier html ca suport pentru respectivul browser. Principalele cuvinte cheie pentru limbajul HTML şi câteva exemple simple le puteți gasi în cadrul cursului Internet şi Intranet.
Presupunând că sunteți familirizați cu documentele html să creem un exemplu de fişier html – ex1.html :
<HTML>

<Title>
Exemplu de inserare a unui applet Java in document HTML
</Title>
<Body>
<H1>
Acesta este un exemplu de inserare a unui applet intr-un document HTML
<H1>
<APPLET CODE = "Dreptunghicolor.class" width=100 height=150>
<PARAM NAME = color value="rosu">
</applet>
</Body>
</HTML>

Pentru a insera un applet Java, în cadrul unui fişier HTML este folosit cuvântul cheie <APPLET> şi </APPLET>. Această declaraţie are doi parametrii obligatorii:
WIDTH – lungime în pixeli
HEIGHT – laţime în pixeli
Aceste elemente care stabilesc dimensiunea applet-ului Java, pot fi completate cu alţi parametri pe care utilizatorul îi poate transmite applet-ului Java. Structura necesară transmiterii acestor parametrii este <PARAM>.
Procesarea unui applet Java cuprinde următoarele etape:

- Browser-ul rezervă spaţiu pentru afişarea applet-ului Java, în funcţie de parametrii HEIGHT şi WIDTH ai APPLET-ului
- Browser-ul citeşte ceilalţi parametri PARAM
- Maşina virtuală – **VM** - este pornită pentru a încărca şi iniţializa applet-ul. Acesta are acces la valorile definite în cadrul comenzii PARAM
- Maşina virtuală creează şi rulează o copie a applet-ului bazată pe fişierul clasă
- Browser-ul apelează funcţia init pentru a se autoiniţializa applet-ul
- VM apelează metoda start când applet-ul a fost pornit şi metoda draw pentru a desena applet-ul

					
Încărcarea appletului	*Iniţializarea appletului*	*„Start-ul„ appletului*	*Rularea appletului*	*Oprirea appletului*	*„Distrugerea” appletului*

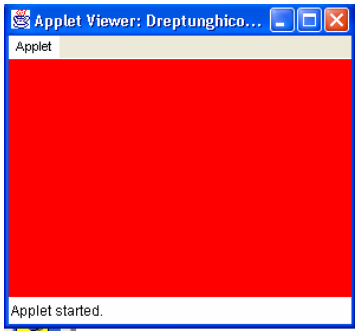
- Dacă applet-ul trebuie redesenat, în momentul în care utilizatorul vizualizează o pagină mare, care necesită scroll, browser-ul apelează metoda paint.
- Dacă se realizează trecerea la o altă pagină HTML, brower-ul apelează metoda stop
- Când browser-ul elimină applet-ul din memorie este appletată metoda destroy.

Un exemplu de applet, cu parametrii este următorul:

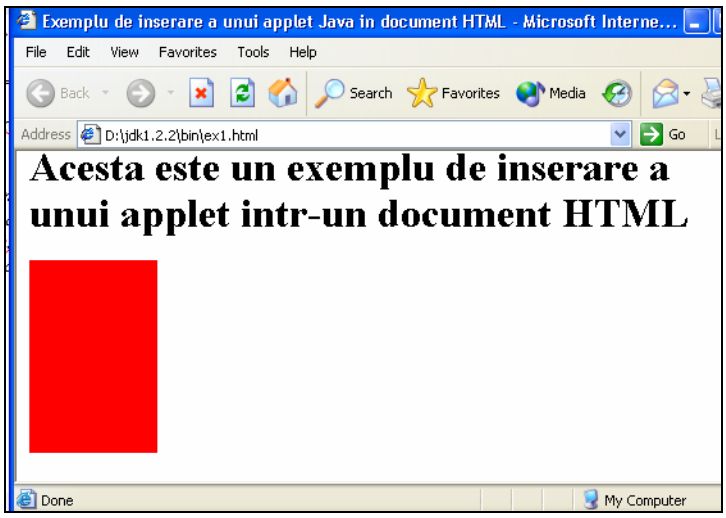
import java.awt.;*
import java.applet.Applet;
*/** Acest applet va afisa un dreptunghi colorat,*

```
*in functie de dorinta utiliyatorului*/
public class Dreptunghicolor extends Applet
{
//Urmatoarea variabila va memora culoarea specificata in documentul HTML
Color boxColor;
public void init()
{ String s;
s = getParameter("color");
//Culoarea implicita este gri
boxColor = Color.gray;
/** Este preluat din documetul HTML un parametru numit Culoare
* a carui valoare poate fi rosu, alb sau albastr. Daca parametrul
* lipseste atunci variabila s va fi null*/
if ( s!=null)
{       if (s.equals("rosu")) boxColor = Color.red;
        if (s.equals("alb")) boxColor = Color.white;
        if (s.equals("albastru")) boxColor = Color.blue;
}
}
/** In continuare urmeay[ comenzile care asigura desenarea
* dreptunghiului in zona rezervata appletului, cu culoarea
* indicata in cadrul documentului HTML*/
public void paint(Graphics g)
       { g.setColor(boxColor);
         g.fillRect(0,0,size().width,size().height);
       }
}
```

Fişierul astfel construit va fi salvat cu numele Dreptunghicolor.java. Urmează compilarea acestui fişier java: javac Dreptunghicolor.java.
In cadrul rularii va apare un avertisment care însă nu influenţează asupra applet-ului.
Vizualizarea rezultatului se realizează cu ajutorul comenzii ***appletviewer exe1.html***.



respectiv prin utilizarea unui browser:



Atenție

Uneori din cauza diferitelor variante de kit-uri Java şi browser-e, precum şi datorită unor probleme de implementare această comandă va furniza o serie de erori. Pentru a verifica rezultatul se poate utiliza cu mult mai multă eficiență un browser compatibil Java care va oferi imaginea portabilă a applet-ului şi a fişierului html.

Tema nr. 2

1. Care este deosebirea dintre un applet Java şi o aplicație Java?
2. Care este structura minimală a unei aplicații Java? Comentați-o!
3. Care sunt principalele aplicații din cadrul mediului Java? Descrieți efectul fiecărei aplicații(javac,...).
4. Care sunt caracteristicile limbajului Java?
5. Ce este un protocol?
6. Descrieți modul în care se obține o aplicație Java.
7. Descrieți modul prin care se obține un applet Java.
8. Rulați şi verificați toate comenzile din cadrul acestui curs, iar pentru data viitoare pentru ca această temă să fie contorizată prezentați o discheta cu toate exemplele din curs rulate. Absența dischetei şi a acestei activitați de rulare duce la anularea îndeplinirii temei.