Curs 12
Organizarea fisierelor .java si .class

Organizarea fisierelor sursa (.java)

Orice aplicatie nebanala trebuie sa fie construita folosind o organizare ierarhica a surselor si fisierelor .class ale sale. Este recomandat ca strategia de organizare a fisierelor sursa sa respecte urmatoarele conventii:

1. Codul sursa al claselor si interfetelor sa se gaseasca în fisiere ale caror nume sa fie numele scurt al claselor/interfetelor si care sa aiba extenisa .java.

   Atentie
   Este obligatoriu ca o clasa/interfata publica sa se gaseasca într-un fisier având numele clasei(interfetei) si extenisa .java. Din acest motiv într-un fisier sursa nu pot exista doua clase publice. Pentru clasele care nu sunt publice acest lucru nu este obligatoriu ci doar recomandat. Intr-un fisier sursa pot exista oricâte clase care nu sunt publice.

2. Fisierele sursa trebuie sa se gaseasca în directoare care sa reflecte numele pachetelor în care se gasesc clasele si interfetele din acele fisiere sursa. Cu alte cuvinte un director va contine surse pentru clase si interfete din acelassi pachet iar numele directorului va fi chiar numele pachetului. Daca numele pachetelor sunt formate din mai multe unitati lexicale separate prin punct, atunci acestea trebuie de asemenea sa corespunda unor directoare ce vor descrie calea spre fisierele sursa ale caror clase/interfete fac parte din pachetele respective.

Vom clarifica modalitatea de organizare a fisierelor sursa ale unei aplicatii printr-un exemplu concret. Sa presupunem ca dorim crearea unui program java care sa reprezinte diverse notiuni matematice din domenii diferite cum ar fi geometrie, algebra, analiza, etc. Pentru a simplifica lucrurile sa presupunem ca dorim sa cream, clase care sa descrie urmatoarele notiuni: *poligon, cerc, poliedru, sfera, grup, functie*. O prima varianta ar fi sa construim câte o clasa java pentru fiecare si sa le plasam în acelasi director împreuna cu un program care sa le foloseasca, însa, având în vedere posibila extindere a aplicatiei cu noi reprezentari de notiuni matematice, aceasta abordare ar fi ineficienta.

O abordare eleganta ar fi aceea în care clasele care descriu notiuni din acelasi domeniu sa se gaseasca în pachete separate si directoare separate. Ierarhia fisierelor sursa ar fi:

```
/matematica
        /surse
                /geometrie
                        /plan
                                Poligon.java
                                Cerc.java
                        /spatiu
                                Poliedru.java
```

```
                        Sfera.java
        /algebra
                Grup.java
        /analiza
                Functie.java
        Matematica.java
```

Clasele descrise în fisierele de mai sus trebuie declarate în pachete denumite corespunzator cu numele directoarelor în care se gasesc:

Poligon.java	`package geometrie.plan;` `public class Poligon { . . . }`
Cerc.java	`package geometrie.plan;` `public class Cerc { . . . }`
Poliedru.java	`package geometrie.spatiu;` `public class Poliedru { . . . }`
Sfera.java	`package geometrie.spatiu;` `public class Sfera { . . . }`
Grup.java	`package algebra;` `public class Grup { . . . }`
Functie.java	`package analiza;` `public class Functie { . . . }`

`Matematica.java` este clasa principala a aplicatiei.
Numele lung al unei clase trebuie sa descrie calea spre acea clasa în cadrul fisierelor sursa ale unei aplicatii.

Organizarea fisierelor .class

In urma compilarii fisierelor sursa vor fi generate unitati de compilare pentru fiecare clasa si interfata din fisierele sursa. Pentru fiecare clasa/interfata va fi generat un fisier cu extensia .class si cu numele clasei/interfetei respective.
Ca si la fisierele .java, un fisier .class trebuie sa se gaseasca într-o serie de directoare care sa reflecte numele pachetului din care face parte. Initial, în urma compilarii fisierele sursa si unitatile de compilare (.class) se gasesc în acelasi director, însa ele pot fi apoi organizate separat.

Revenind la exemplul de mai sus, putem avea urmatoarea organizare:

```
/matematica
        /clase
                /geometrie
                        /plan
                                Poligon.class
                                Cerc.class
                        /spatiu
                                Poliedru.class
                                Sfera.class
                /algebra
                        Grup.class
                /analiza
                        Functie.class
                Matematica.class
```

Crearea acestei structuri ierarhice poate fi facuta automat de catre compilator. In directorul aplicatiei (`matematica`) cream subdirectorul `clase` si dam comanda:

```
        javac -sourcepath surse surse/Matematica.java -d clase
sau
        javac -classpath surse surse/Matematica.java -d clase
```

Curs 12

Necesitatea organizarii fisierelor

Organizarea fisierelor sursa este necesara deoarece în momentul când compilatorul întâlneste un nume de clasa el trebuie sa poata identifica acea clasa, ceea ce înseamna ca trebuie sa gaseasca fiserul sursa care o contine. Similar, fisierele .class sunt organizate astfel pentru a da posibilitatea interpretorului sa gaseasca o anumita clasa în timpul executiei programului. Insa aceasta organizare nu este suficienta deoarece specifica numai partea finala din calea catre fisierele .java si .class : `/matematica/clase/geometrie/plan/Poligon.class`. Pentru aceasta, atât la compilare cât si la interpretare trebui specificata lista de directoare în care se gasesc fisierele aplicatiei. Aceasta lista se numeste *cale de cautare (classpath)*.

Definitie

O *cale de cautare* este o lista de directoare sau arhive în care vor fi cautate fisierele necesare unei aplicatii. Fiecare director din calea de cautare este directorul imediat superior structurii de directoare formate de organizarea claselor în directoare corespunzatoare pachetelor, astfel încât compilatorul si interpretorul sa poata construi calea completa spre clasele aplicatiei. Implicit calea de cautare este formata doar din directorul curent.

Sa consideram clasa principala a aplicatiei `Matematica.java`:

```
import geometrie.plan.*;
import algebra.Grup;
import analiza.Functie;

public class Matematica {
        public static void main(String args[]) {
                Poligon a = new Poligon();
                geometrie.spatiu.Sfera = new geometrie.spatiu.Sfera();
                //...
        }
}
```

Identificarea unei clase referite în program se face în felul urmator:

1. La directoarele aflate în calea de cautare se adauga subdirectoarele specificate în import sau în numele lung al clasei
2. In directoarele formate este cautat un fisier cu numele clasei. In cazul în care nu este gasit nici unul sau sunt gasite mai multe va fi semnalata o eroare.

Setarea caii de cautare (CLASSPATH)

Se poate face în doua modalitati:

1. Setarea variabile de mediu **CLASSPATH** (nerecomandat)

```
2.   UNIX:
3.           SET CLASSPATH = cale1 : cale2 : ...
4.           Recomandat sa apara si directorul curent .
5.   DOS shell (Windows 95/NT):
6.           SET CLASSPATH = cale1 ; cale2 ; ...
7.           Recomandat sa apara si directorul curent .
```

8. Folosirea optiunii **-classpath** la compilarea si interpretarea programelor

```
9.           javac - classpath <cale de cautare> <fisier.java>
10.          java  - classpath <cale de cautare> <fisier.class>
```

Lansarea în executie a aplicatiei noastre, din directorul aplicatiei, s-ar putea face astfel:

```
java -classpath clase Matematica.java
```

O organizare eficienta a fisierelor aplicatiei ar arata astfel:

```
/matematica
```

```
/surse
/clase
compile.bat     (javac -sourcepath surse surse/Matematica.java -d clase)
run.bat         (java  -classpath clase Matematica.java)
```

Arhive JAR (Java ARchive)

Definitie

Sunt arhive în format ZIP folosite pentru compresarea mai multor fisere în unul singur. Diferenta consta în faptul ca un fisier JAR contine, pe lânga fiserele arhivate, si un director denumit `META-INF`, ce contine diverse informatii auxiliare.

Un fisier JAR poate fi creat folosind utilitarul **jar** sau metode ale pachetului **java.util.jar**.

Beneficii

- portabilitate - este singurul format de arhivare independent de platforma
- compresarea fisierelor est optimizata pentru fisiere de tip `class`
- minimizarea timpului de incarcare a unui applet : daca appletul (fisiere class, resurse, etc) este compresat intr-o arhiva jar, el poate fi incarcat intr-o singura tranzactie HTTP, fara a fi deci nevoie de a se deschide o conexiune noua pt. fiecare fisier.
- securitate - arhivele JAR pot fi "semnate" electronic
- mecanismul pentru lucrul cu fisiere JAR este parte integrata a platformei Java.

Folosirea utilitarului jar

Arhivatorul jar se gaseste în subdirectorul `bin` al directorului în care este instalat mediul Java. In tabelul de mai jos sunt sumarizate operatiile principale:

Operatie	Comanda
Crearea unei arhive	`jar cf nume-arhiva fisier(e)-intrare`
Vizualizare continutului unei arhive	`jar tf nume-arhiva`
Extragerea continutului unei arhive	`jar xf nume-arhiva`
Extragerea doar a unor fisiere dintr-o arhiva	`jar xf nume-arhiva fisier(e)-arhivate`
Executarea unei aplicatii împachetate într-un fisier jar (JDK 1.1)	`jre -cp app.jar ClasaPrincipala`
Executarea unei aplicatii împachetate într-un fisier jar (JDK 1.2) Alicatia trebuie compresata in asa fel incat interpretorul sa stie unde se gaseste clasa principala	`java -jar app.jar`
Deschiderea într-un browser a unui applet compresat într-un fisier jar	`<applet code=NumeClasaApplet.class` **`archive`**`="NumeArhiva.jar"` `width=latime height=înaltime> </applet>`

Exemple:

- arhivarea a doua fisiere class: `jar cf classes.jar A.class B.class`
- arhivarea tuturor fisierelor din directorul curent: `jar cvf allfiles.jar *`

Executarea aplicatiilor împachetate într-o arhiva JAR

Curs 12
Pentru a rula o aplicatie împachetata într-o arhiva JAR trebuie sa facem cunoscuta interpretorului numele clasei principale a aplicatiei. Sa consideram urmatorul exemplu, în care dorim sa arhivam clasele : `GraphEditor.class, Graph.class, Edge.class, Vertex.class` , clasa principala fiind `GraphEditor`.
Vom scrie:

```
jar cvfm editor.jar *.class
```

In urma acestei comenzi vom obtine arhiva `editor.jar`. Daca vom încerca sa lansam în executie aceasta arhiva prin comanda `java -jar editor.jar` vom obtine urmatoarea eroare:

```
"Failed to load Main-Class manifest from editor.jar"
```

. Aceasta înseamna ca în fiserul `Manifest.mf` ce se gaseste în directorul `META-INF` trebuie sa înregistram clasa principala a aplicatiei. Acest lucru îl vom face în doi pasi:

1. se creeaza un fisier cu un nume oarecare (ex: mymanifest) în care vom scrie:
   `Main-Class : GraphEditor.java`
2. adaugam aceasta completare la fisierul manifest al arhivei editor.jar:
   `jar cvfm editor.jar mymanifest`

Ambele operatii puteau fi executate într-un singur pas:

```
jar cvfm editor.jar mymanifest *.class
```

Fisiere JAR executabile

Pe sistemele Win32, platforma Java 2 va asocia extensiile .jar cu interpretorul java, ceea ce înseamna cs facând dublu-click pe o arhiva jar va fi lansata în executie aplicatia împachetata în acea arhiva (daca exista o clasa principala).