Curs 13
Pachete

Crearea unui pachet

Definitie

> Un *pachet* este o colectie de clase si interfete înrudite. Sunt folosite pentru gasirea si utilizarea mai usoara a claselor, pentru a evita conflictele de nume si pentru a controla accesul la anumite clase. In alte limbaje de programare pachetele se numesc librarii.

Toate clasele si interfetele Java apartin la diverse pachete, grupate dupa functionalitatea lor: clasele de baza se gasesc în pachetul **`java.lang`**, clasele pentru intrari/iesiri sunt în **`java.io`**, clasele pentru grafica în **`java.awt`**, cele pentru construirea applet-urile în **`java.applet`**, etc. Crearea unui pachet se realizeaza prin scriere la începutul fisierelor sursa ce contin clasele si interfetele pe care dorim sa le grupam într-un pachet a instructiunii: **`package NumePachet;`** Sa consideram un exemplu: presupunem ca avem doua fisiere sursa `Graf.java` si `Arbore.java`

Graf.java	Arbore.java

`package grafuri;`
`class Graf {...}`
`class GrafPerfect extends Graf {...}`
	`package grafuri;`
	`class Arbore {...}`
	`class ArboreBinar extends Arbore {...}`

Clasele `Graf, GrafPerfect, Arbore, ArboreBinar` vor face parte din acelasi pachet `grafuri`. Instructiunea `package` actioneaza asupra întregului fisier sursa la începutul caruia apare. Cu alte cuvinte nu putem specifica faptul ca anumite clase dintr-un fisier sursa apartin unui pachet iar altele altui pachet.
Daca nu este specificat un anumit pachet, clasele unui fisier sursa vor face parte din pachetul implicit (care nu are nici un nume). In general, pachetul implicit este format din toate clasele si intefetele directorului curent.
Este recomandabil ca toate clasele si intefetele sa fie plasate în pachete. Pachetul implicit este folosit doar pentru aplicatii mici sau la începutul dezvoltarii unei aplicatii.

Denumirea unui pachet

Exista posibilitatea ca doi programatori care lucreaza la un proiect comun sa foloseasca acelasi nume pentru unele din clasele lor. De asemenea, se poate ca una din clasele unei aplicatii sa aiba acelasi nume cu o clasa a mediului Java. Acest lucru este posibil atât timp cât clasele cu acelasi nume se gasesc în pachte diferite, ele fiind diferentiate prin prefixarea lor cu numele pachetelor. Asadar numele complet al unei clase este format din numele pachetului la care apartine + numele sau:

`numePachet.NumeClasa`
`Ex: java.lang.String (java.lang=pachet, String=clasa)`

De exemplu sa presupunem ca în aplicatia noastra folosim o clasa numita `Stack`:

```
package my_package;
class Stack { ... }
```

Clasa `Stack` exista deja în pachetul `java.util`. Diferentierea între cele doua clase se va face prin specificarea numelui complet al clasei, adica numelePachetului.NumeleClasei:

```
java.util.Stack s1 = new  java.util.Stack();
my_package.Stack s2 = new  my_package.Stack();
```

Ce se întâmpla însa când doi programatori care lucreaza la un proiect comun folosesc clase cu acelasi nume ce se gasesc în pachete cu acelasi nume ? Pentru a evita acest lucru companiile folosesc inversul domeniului lor Internet în denumirea pachetelor implementate în cadrul companiei, cum ar fi `com.company.numePachet`. In cadrul unei aceeasi companii conflictele de nume trebuie rezolvate prin diverse conventii de uz intern.De exemplu, adresa mea de e-mail este `acf@infoiasi.ro`, ceea ce înseamna ca domeniul meu Internet este `infoiasi.ro`. Pachetele create de mine ar trebui denumite `ro.infoiasi.NumePachet`. Pentru a rezolva conflicte cu alti programatori din acelasi domeniu cu mine pachetele s-ar putea numi: `ro.infoiasi.acf.NumePachet`.

Folosirea membrilor unui pachet

Conform specificatiilor de acces ale unei clase si ale mebrilor ei doar clasele publice si membrii declarati publici ai unei clase sunt accesibili în afara pachetului în care acestea se gasesc. (vezi "Specificatori de acces pentru membrii unei clase")
Pentru a folosi o clasa publica dintr-un pachet sau pentru a apela o metoda publica a unei clase public a unui pachet exista trei solutii:

- specificarea numelui complet al clasei
- importul clasei respective
- importul întregului pachet în care se gaseste clasa

Specificarea numelui complet al calsei se face, asa cum am vazut, prin prefixarea numelui clasei cu numele pachetului: `numePachet.NumeClasa`. Aceasta metoda este recomandata doar pentru cazul în care folosirea acelei clase se face o singura data sau foarte rar. De exemplu ar fi extrem de neplacut sa scriem de fiecare data când vrem sa declaram un sir de caractere sau sa folosim un obiect grafic secvete de genul;

```
java.lang.String s = "neplacut";
java.awt.Rectangle r = new  java.awt.Rectangle();
java.awt.Circle c = new  java.awt.Circle();
```

In aceste situatii vom importa (include) clasa respective sau întreg pachet din care face parte in aplicatia noastra. Acest lucru se realizeaza prin instructiunea **`import`**, care trebuie sa apara la începutul fisierelor sursa, imediat dupa instructiunea `package`.

Importul unei clase sau interfete

Se face printr-o instructiune import în care specificam numele clasei (interfetei) pe care dorim sa o folosim dintr-un pachet:

```
import java.awt.Rectangle;
```

Din acest moment vom putea folosi în clasele fisierului în care am plasat instructiunea de import numele scurt al clasei `Rectangle`

```
Rectangle r = new Rectangle(0,0,100,100);
```

Aceasta abordare este eficienta în cazul în care nu avem nevoie decât de acea clasa sau doar de câteva clase din pachetul respectiv. Daca în exemplul nostru am avea nevoie si de clasele `Circle, Line, Point, Polygon` ar trebui sa avem câte o instructiune de import pentru fiecare dintre ele:

```
import java.awt.Rectangle;
import java.awt.Circle;
```

```
import java.awt.Line;
import java.awt.Point;
import java.awt.Polygon;
```

In aceasta situatie ar fi recomandat importul întregului pachet si nu al fiecarei clase în parte.

Importul unui pachet (importul la cerere)

Se face printr-o instructiune import în care specificam numele pachetului ale carui clase si interfete dorim sa le folosim dintr-un pachet, urmat de simbolul **'*'**. Se mai numeste *import la cerere* deoarece încarcarea claselor se face dinamic în momentul apelarii lor. Este cel mai uzual tip de import.

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

Din acest moment vom putea folosi în clasele fisierului în care am plasat instructiunea de import numele scurt al tuturor claselor pachetului importat:

```
Rectangle r = new Rectangle();
Circle c = new Circle(); ...
```

Atentie

***** nu are semnificatia uzuala de wildcard (masca) si nu poate fi folosit decât ca atare.

```
import java.awt.C*; //eroare de compilare
```

In cazul în care sunt importate doua pachete care contin o clasa cu acelasi nume atunci referirea la ea trebuie facuta folosind numele complet al clasei respective.

```
//Stack.java
        package my_package;
        class Stack { ... }
//alt fisier sursa
        import java.util.*;
        import my_package.*;
        ...
        Stack s = new Stack(); //ilegal -> conflict de nume
        java.util.Stack s1 = new  java.util.Stack(); //corect
        my_package.Stack s2 = new  my_package.Stack();//corect
```

Mediul Java importa automat trei pachete pentru toate fisierele sursa:

- pachetul `java.lang`
- pachetul curent
- pachetul implicit (fara nume)

Pachetele JDK

Limbajul Java se bazeaza pe o serie de biblioteci (pachete) cu ajutorul carora se pot construi aplicatiile. Exista deci un set de clase deja implementate, ceea ce reduce timpul de dezvoltare a unui program. Cele mai importante sunt:

`java.applet`	suport pt scrierea de appleturi
`java.awt`	suportul pentru grafica(Abstract Windowing Toolkit)
`java.beans`	suport pentru scrierea de componente reutilizabile
`java.io`	intrari/iesiri, acces la fisiere
`java.lang`	clasele de baza ale limbajului
`java.math`	operatii matematice
`java.net`	acces la retea

`java.rmi`	executie la distanta (Remote Message Interface)
`java.security`	mecanisme de securitate : criptare, autentificare
`java.sql`	interogari SQL
`java.text`	suport pentru formatarea textelor
`java.util`	clase utile : Vector, Stack, Random, etc

Pachetul `java.lang` contine elementele de baza ale limbajului si este importat automat.
