

# Pachete

- Ce este un pachet ?
- Pachetele standard (J2SDK)
- Importul unei clase sau interfețe
- Importul la cerere
- Importul static
- Crearea unui pachet
- Organizarea fișierelor
- Setarea căii de căutare
- Arhive JAR
- Executarea aplicațiilor arhivate

## Ce este un pachet ?

**Pachet** = Colecție de clase și interfețe

### Scopul:

- Organizarea lucrului
- Găsirea și utilizarea mai ușoară a claselor
- Evitarea conflictelor de nume
- Controlul accesului

# Pachetele standard (J2SDK)

- **java.lang** - clasele de bază ale limbajului Java
- **java.io** - intrări/ieșiri, lucrul cu fișiere
- **java.util** - clase și interfețe utile
- **java.applet** - dezvoltarea de appleturi
- **java.awt** - interfața grafică cu utilizatorul
- **java.awt.event** - tratare evenimente
- **java.beans** - scrierea de componente reutilizabile
- **java.net** - programare de rețea
- **java.sql** - lucrul cu baze de date
- **java.rmi** - execuție la distanță
- **java.security** - mecanisme de securitate
- **java.math** - operații matematice cu numere mari
- **java.text** - lucrul cu texte, date și numere independent de limbă
- **java.lang.reflect** - introspecție
- **javax.swing** - interfața grafică cu utilizatorul, mult îmbogățită față de AWT.

## Folosirea membrilor unui pachet

Folosirea unei clase publice dintr-un pachet:

- specificarea numelui complet
- importul clasei respective
- importul întregului pachet

### **numePachet.NumeClasa.**

Button                   - numele scurt al clasei  
java.awt                 - pachetul din care face parte  
java.awt.Button         - numele complet al clasei

//Problema:

```
java.awt.Button b1 = new java.awt.Button("OK");  
java.awt.Button b2 = new java.awt.Button("Cancel");  
java.awt.TextField tf1 =  
    new java.awt.TextField("Neplacut");  
java.awt.TextField tf2 =  
    new java.awt.TextField("Tot neplacut");
```

## Importul unei clase sau interfețe

```
import numePachet.numeClasa;

//Pentru exemplul nostru:
import java.awt.Button;
import java.awt.TextField;
...
Button b1 = new Button("OK");
Button b2 = new Button("Cancel");
TextField tf1 = new TextField("Placut");
TextField tf2 = new TextField("Foarte placut");

//Problema:
import java.awt.Button;
import java.awt.TextField;
import java.awt.Rectangle;
import java.awt.Line;
import java.awt.Point;
import java.awt.Polygon;
```

## Importul la cerere

```
import numePachet.*;

//Pentru exemplul nostru:
import java.awt.*;
...
Button b = new Button("OK");
Point p = new Point(0, 0);
```

```
import java.awt.C*; = eroare
```

Importate automat:

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

# Ambiguități

```
import java.awt.*;  
// Contine clasa List
```

```
import java.util.*;  
// Contine interfata List
```

```
...
```

```
List x; //Declaratie ambigua
```

```
java.awt.List a = new java.awt.List(); //corect  
java.util.List b = new ArrayList(); //corect
```

## Importul static

Referirea constantelor statice ale unei clase

```
import static numePachet.NumeClasa.*;
    NumeClasa.CONSTANTA
        CONSTANTA

// Inainte de versiuna 1.5
import java.awt.BorderLayout.*;
...
fereastră.add(new Button(), BorderLayout.CENTER);

// Incepand cu versiunea 1.5
import java.awt.BorderLayout.*;
import static java.awt.BorderLayout.*;
...
fereastră.add(new Button(), CENTER);
```

## Crearea unui pachet

```
package numePachet;
```

```
//Fisierul Graf.java  
package grafuri;  
class Graf {...}  
class GrafPerfect extends Graf {...}
```

```
//Fisierul Arbore.java  
package grafuri;  
class Arbore {...}  
class ArboreBinar extends Arbore {...}
```

Pachetul implicit = directorul curent

## Organizarea fișierelor sursă

Organizarea surselor = foarte importantă

### Recomandări:

- **clasa - fisier**  
sursa clasei C - fisierul C.java  
obligatoriu pentru clase publice
- **pachet - director**  
clasele pachetului - fisierele directorului

```
/matematica
  /surse
    /geometrie
      /plan
        Poligon.java
        Cerc.java
      /spatiu
        Poliedru.java
        Sfera.java
    /algebra
      Grup.java
    /analiza
      Functie.java

Matematica.java
```

```
// 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 { . . . }
```

# Organizarea unităților de compilare

Obligatoriu:

- **clasa - fisier**

clasa C - fisierul C.class

- **pachet - director**

directorul fisierului C.class **trebuie** să reflecte pachetul clasei C

```
//Compilare completa:
```

```
javac -sourcepath surse
```

```
    surse/Matematica.java -d clase
```

```
//Partial:
```

```
javac surse/geometrie/plan/*.java -d clase
```

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

Matematica.class
```

## Necesitatea organizării fișierelor

Identificarea fișierelor sursă /  
.class

**Sufix:**

geometrie/spatiu/Sfera.class

**Prefix:**

c:/java/aplicatii/matematica/clase

**cale de cautare (classpath) = listă  
de directoare sau arhive**

Implicit, calea de căutare este formată  
doar din directorul curent.

```
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();
        //...
    }
}
```

- La directoarele aflate în calea de căutare se adaugă subdirectoarele specificate în import sau în numele lung al clasei
- În directoarele formate este căutat un fișier cu numele clasei. În cazul în care nu este găsit nici unul sau sunt găsite mai multe va fi semnalată o eroare.

# Setarea căii de căutare

- Variabila **CLASSPATH**

UNIX:

```
SET CLASSPATH = cale1:cale2:...
```

DOS shell (Windows 95/NT/...):

```
SET CLASSPATH = cale1;cale2;...
```

- Opțiunea **-classpath**

```
javac - classpath <cale de cautare>  
      <surse java>
```

```
java  - classpath <cale de cautare>  
      <clasa principala>
```

```
java -classpath clase Matematica
```

```
java -classpath c:\java\aplicatii\matematica\clase  
      Matematica
```

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

## Arhive JAR

**Java Archive** = arhive ZIP + META-INF

Crearea unei arhive:

- Utilitarul **jar**
- Clase suport din **java.util.jar**

Beneficii:

- compresare
- portabilitate
- minimizarea timpului de încărcare din rețea
- securitate - ”semnare” electronică
- parte integrată a platformei Java

## Folosirea utilitarului jar

- Crearea unei arhive  
`jar cf arhiva.jar fișier(e)-intrare`
- Vizualizare conținutului  
`jar tf nume-arhiva`
- Extragerea conținutului  
`jar xf arhiva.jar`
- Extragerea doar a unor fișiere  
`jar xf arhiva.jar fișier(e)-arhivate`
- Executarea unei aplicații  
`java -jar arhiva.jar`
- Deschiderea unui applet arhivat  
`<applet code=A.class archive="arhiva.jar"  
...>`

Exemple:

- Arhivarea a două fișiere class:  
`jar cf classes.jar A.class B.class`
- arhivarea tuturor fișierelor din directorul curent:  
`jar cvf allfiles.jar *`

# Executarea aplicațiilor arhivate

- Crearea unui fisier manifest

```
//manifest.txt  
Main-Class: Matematica
```

- Crearea arhivei

```
jar cvfm mate.jar manifest.txt  
    geometrie analiza algebra Matematica.class
```

- Executia

```
java -jar mate.jar
```