

Programozás III

CSOMAG

CSOMAGOK

Az összetartozó osztályok és interfészek egy csomagba (package) kerülnek.



A Java is csomagok halmaza: csomagokban van a fejlesztő környezet és az osztálykönyvtárak is:

rt.jar fájl - A Java szabványos osztálygyűjteménye
jre\lib\rt.jar

CSOMAGOK – RT.JAR

c:\Program Files\Java\jdk1.6.0_03\jre\lib*						c:\temp*					
↑Név	Kit.	Méret	Dátum	Attr.		↑Név	Kit.	Méret	Dátum	Attr.	
content-t..	properti..	5 506	2007.10.27 19:26	-a-		↑Név	Kit.	Méret	Dátum	Attr.	
deploy	jar	2 738 010	2007.09.25 00:11	-a-		↑Név	Kit.	Méret	Dátum	Attr.	
flavormap	properti..	3 828	2007.10.27 19:26	-a-		[.]	<DIR>		2008.03.08 21:46	—	
fontconfig	bfc	3 058	2007.10.27 19:26	-a-		[com]	<DIR>		2008.03.08 21:42	—	
fontconfi..	bfc	2 710	2007.10.27 19:26	-a-		[java]	<DIR>		2008.03.08 21:44	—	
fontconfi..	src	7 591	2007.10.27 19:26	-a-		[javax]	<DIR>		2008.03.08 21:45	—	
fontconfi..	src	8 003	2007.10.27 19:26	-a-		[META-INF]	<DIR>		2008.03.08 21:42	-a-	
javaws	jar	722 634	2007.09.25 00:11	-a-		[org]	<DIR>		2008.03.08 21:45	—	
jce	jar	88 256	2007.10.27 19:26	-a-		[sun]	<DIR>		2008.03.08 21:46	—	
jsse	jar	617 098	2007.09.25 00:09	-a-		[sunw]	<DIR>		2008.03.08 21:46	—	
jvm.hprof	txt	2 807	2007.10.27 19:26	-a-							
logging	properti..	2 245	2007.10.27 19:26	-a-							
manage..	jar	382	2007.09.24 21:36	-a-							
meta-index		1 948	2007.10.27 19:26	-a-							
net	properti..	3 070	2007.10.27 19:26	-a-							
plugin	jar	1 048 014	2007.09.25 00:11	-a-							
psfont.pr..	ja	3 070	2007.10.27 19:26	-a-							
psfontj2d	propert..	10 669	2007.10.27 19:26	-a-							
resources	jar	1 071 615	2007.09.24 22:12	-a-							
rt	jar	46 154 433	2007.09.25 00:09	-a-							
sound	properti..	1 210	2007.10.27 19:26	-a-							
tzmappings		7 265	2007.10.27 19:26	-a-							

Az rt.jar kicsomagolva (esetleg át kell nevezni rt.zip-re)

CSOMAGOK – RT.JAR

c:\temp*						c:\temp\java*					
↑Név	Kit.	Méret	Dátum	Attr.		↑Név	Kit.	Méret	Dátum	Attr.	
↑Név	Kit.	Méret	Dátum	Attr.		↑Név	Kit.	Méret	Dátum	Attr.	
↑Név	Kit.	Méret	Dátum	Attr.		↑Név	Kit.	Méret	Dátum	Attr.	
[.]	<DIR>		2008.03.08 21:46	—		[.]	<DIR>		2008.03.08 21:44	—	
[com]	<DIR>		2008.03.08 21:42	—		[applet]	<DIR>		2008.03.08 21:46	—	
[java]	<DIR>		2008.03.08 21:44	—		[awt]	<DIR>		2008.03.08 21:46	—	
[javax]	<DIR>		2008.03.08 21:45	—		[beans]	<DIR>		2008.03.08 21:46	—	
[META-INF]	<DIR>		2008.03.08 21:42	-a-		[io]	<DIR>		2008.03.08 21:46	—	
[org]	<DIR>		2008.03.08 21:45	—		[lang]	<DIR>		2008.03.08 21:46	—	
[sun]	<DIR>		2008.03.08 21:46	—		[math]	<DIR>		2008.03.08 21:46	—	
[sunw]	<DIR>		2008.03.08 21:46	—		[net]	<DIR>		2008.03.08 21:46	—	
						[nio]	<DIR>		2008.03.08 21:46	—	
						[rmj]	<DIR>		2008.03.08 21:46	—	
						[security]	<DIR>		2008.03.08 21:46	—	
						[sql]	<DIR>		2008.03.08 21:44	—	
						[text]	<DIR>		2008.03.08 21:46	—	
						[util]	<DIR>		2008.03.08 21:46	—	

Az rt.jar\java könyvtár tartalma

CSOMAGOK – RT.JAR

c:\temp\java\lang*.*				c:\temp\java\math*.*			
Név	Kit.	Méret	Dátum	Név	Kit.	Méret	Dátum
<DIR>			2008.03.08	<DIR>			2008.03.08 21
[annotation]	<DIR>		2008.03.08	BigDecimal	class	23 230	2007.09.24 22
[instrument]	<DIR>		2008.03.08	BigInteger	class	31 376	2007.09.24 22
[management]	<DIR>		2008.03.08	BitSieve	class	1 991	2007.09.24 22
[ref]	<DIR>		2008.03.08	MathContext	class	3 403	2007.09.24 22
[reflect]	<DIR>		2008.03.08	MutableBigInteger	class	12 883	2007.09.24 22
AbstractMethodError	class	330	2007.09.24	RoundingMode	class	1 577	2007.09.24 22
AbstractStringBuilder	class	10 332	2007.09.24	SignedMutableBigInte..	class	1 146	2007.09.24 22
Appendable	class	344	2007.09.24				
ApplicationShutdownH..	class	1 981	2007.09.24				
ArithmeticException	class	318	2007.09.24				
ArrayIndexOutOfBounds..	class	640	2007.09.24				
ArrayStoreException	class	318	2007.09.24				
AssertionError	class	1 403	2007.09.24				
AssertionStatusDirect..	class	366	2007.09.24				
Boolean	class	2 208	2007.09.24				
Byte\$ByteCache	class	437	2007.09.24				
Byte	class	3 353	2007.09.24				
Character\$CharacterC..	class	463	2007.09.24				
Character\$Subset	class	657	2007.09.24				
Character\$UnicodeBl..	class	16 911	2007.09.24				
Character	class	17 383	2007.09.24				
CharacterDataFlu...	class	25 745	2007.09.24				

A java.lang és a java.math könyvtár

CSOMAGOK

A csomagok egymásba ágyazhatóak.

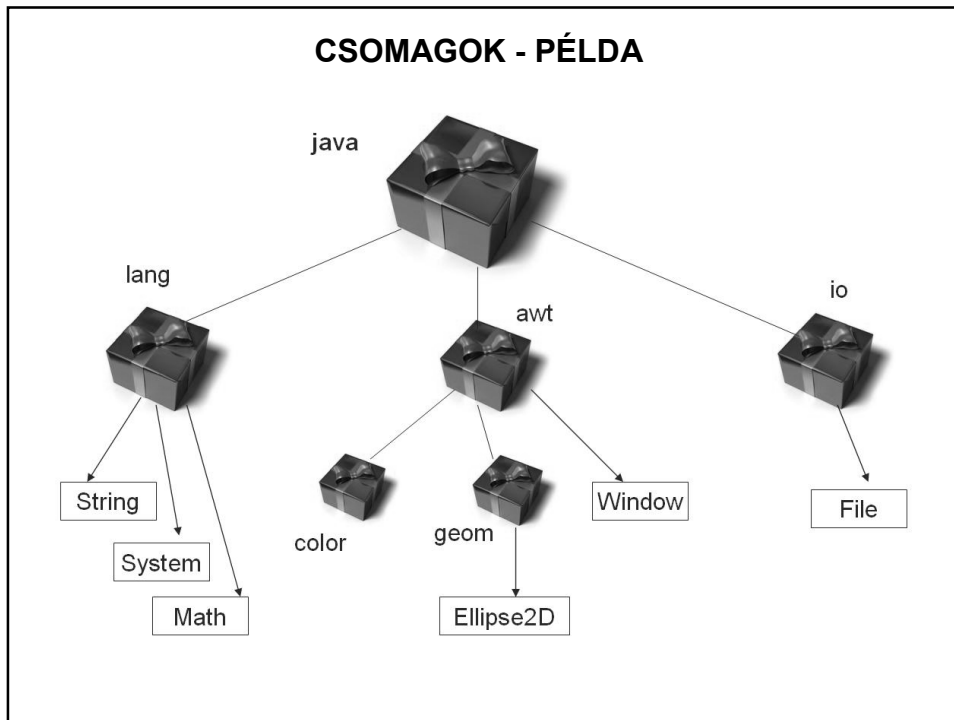
Tetszőleges mélységű csomagstruktúra kialakítható.

Egy szintre akárhány osztály/interfész kerülhet.

Lehet olyan csomag, amelyben nincsenek osztályok/interfészek pl.: **java** csomag

A csomagolás logikai szinten történik

CSOMAGOK - PÉLDA



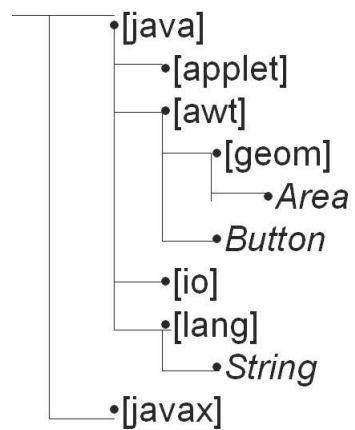
CSOMAGOK - KÖNYVTÁRSTRUKTÚRA

Csomagok: logikai szerkezet



Könyvtárak: fizikai szerkezet

Csomagnév = könyvtárnév!!



osztály: nagy kezdőbetűvel

csomag: kis kezdőbetűvel

CSOMAGOK - NEVEK

Ugyanaz vonatkozik rájuk, mint a könyvtárnevekre, vagyis:

Egy csomagon belül nem lehet azonos nevű csomag és osztály.

Az osztályra teljes útvonallal hivatkozunk – **minősített név**

pl.: *java.lang.Math*

Vigyázat!!! nincsenek relatív minősített nevek

pl.: *lang.Math* nem elegendő

De itt sem kell mindig a teljes útvonallal hivatkozni.

CSOMAGOK - NEVEK

Csomagok deklarációja:

A fordítási egység elején a **package** kulcsszóval jelöljük, hogy melyik csomagba tartozik:

```
package csomag;  
public class A { }
```

Egy fordítási egységben csak egy package deklaráció lehet.

A csomagot teljes elérési útvonallal írjuk.

CSOMAGOK - OSZTÁLYHIVATKOZÁSOK

Definícióval megegyező csomagban egyszerű névvel hivatkozunk.

Definíciótól eltérő csomagban minősített nevekkel (teljes útvonal)

vagy: egyenkénti import deklaráció

vagy: tömbösített (igény szerinti) import deklaráció

vagy: automatikus import deklaráció – egyetlen ilyen csomag van, a **java.lang** csomag.

CSOMAGOK - IMPORT

Egyenkénti import: minden osztályt egyenként importálunk
pl.: `import [csomag1.[csomag2]...].Osztály;`

Igény szerinti import:

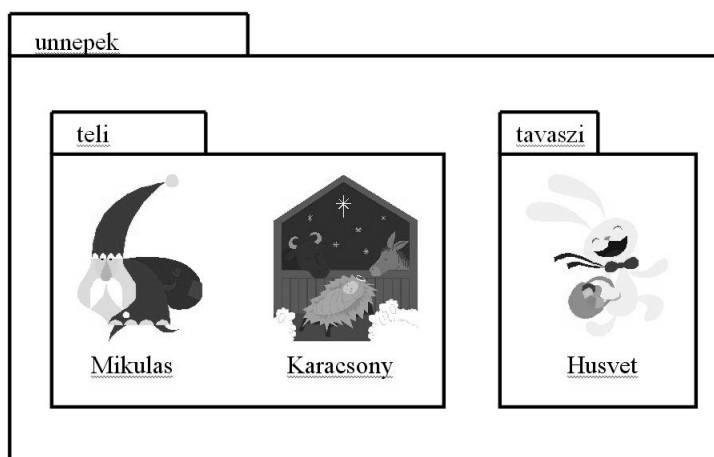
`import [csomag1.[csomag2]...].*;`

a * (joker) karakterrel az összes osztályt importáljuk – ugyanúgy használható, mint könyvtár/fájl struktúrák esetén!

CSOMAGOK - LÁTHATÓSÁG

- **+ public**
tetszőleges csomagból látható
- **nincs módosító (package private)**
csak a saját csomagja láthatja az osztályt
- **- private**
csak a deklaráció osztály érheti el
- **# protected**
saját csomagból bárki és az utódok érhetik el

CSOMAGOK - PÉLDA



Csomag UML jelölése: téglalap kis „füllel”

CSOMAGOK - PÉLDA

A Mikulas.java állomány

```
package unnep.teli;

public class Mikulas{

    public Mikulas(){
        System.out.println("Ne felejtse el kirakni a csizmádat!");
    }
}
```

A Karacsony.java állomány

```
package unnep.teli;

public class Karacsony{

    public Karacsony(){
        System.out.println("Áldott karácsonyi ünnepeket!");
    }
}
```

CSOMAGOK - PÉLDA

A Husvet.java állomány

```
package unnep.tavaszi;

public class Husvet{

    public Husvet(){
        System.out.println("Vidám, örömteli húsvétot!");
        System.out.println("Józan locsolókat,");
        System.out.println("Pirostojást ajándékozó lányokat!");
    }
}
```


CSOMAGOK - PÉLDA

Az Indit.java állomány

```
import unnep.teli.*;
import unnep.tavaszi.Husvet;

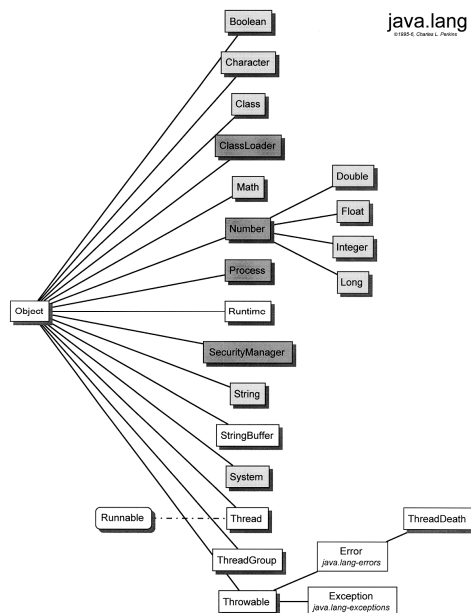
public class Indit{

    public static void main(String[] args){
        Husvet h = new Husvet();
    }
}
```

Eredmény:

```
Vidám, örömteli húsvétot!
Józan locsolókat,
Pirostojást ajándékozó lányokat!
```

CSOMAGOK – PÉLDA: A JAVA.LANG CSOMAG



Olyan alapvető típusokat tartalmaz, amelyekre szükség van egy program futtatásához.

Az `Object` osztály minden osztály közös őse.

A JAVA.LANG CSOMAG – NÉHÁNY OSZTÁLY

Csomagoló osztályok (wrapper classes):

Boolean, Character, Byte, Short, Integer, Long, Float, Double.

Ezek a megfelelő primitív típusokat (boolean, char, int, float, stb.) foglalják osztályokba, és ezzel lehetővé teszik, hogy a primitív értékeket objektumként kezeljük.

(Például a konvertáláshoz a megfelelő osztály megfelelő metódusa szükséges.)

A numerikus osztályok közös absztrakt őse: a Number.

(Az absztrakt osztály csak örökítési célokat szolgál, belőle nem lehet példányt létrehozni.)

KITÉRŐ: A CSOMAGOLÓ OSZTÁLYOK HASZNÁLATA

Sokszor szükség lehet rá, hogy pl. int típusból Integer-t hozzunk létre, stb. Ezt a folyamatot nevezik boxing-nak. (A fordítottja az unboxing.)

Autoboxing: a nyelv automatikusan elvégzi.

Mikor lehet rá szükség?

Pl. egészekből álló listát szeretnénk

<http://stackoverflow.com/questions/27647407/why-do-we-use-autoboxing-and-unboxing-in-java>

A JAVA.LANG CSOMAG – NÉHÁNY OSZTÁLY

Math osztály:

matematikai konstansokat és függvényeket definiál.
(Pl. abs, cos, sin, exp, stb. – részleteket ld. a help-ben.)
Ez egy final osztály – emiatt nem örökíthető, és nincs publikus konstruktora (kizárólag statikus deklarációkat tartalmaz), ezért nem példányosítható.

Használata pl.:

```
double x = Math.PI;  
double y = Math.sin(x);           // az x sinusát adja vissza  
double z = Math.random();        // 0 ≤ x < 1 véletlen szám
```

A JAVA.LANG CSOMAG – NÉHÁNY OSZTÁLY

System osztály:

a rendszer működésével kapcsolatos alapvető metódusokat és objektumokat tartalmazza. Ez sem örökíthető és nem is példányosítható.

Ebben vannak pl. az in és out objektumok – a kiíratáshoz, beolvasáshoz, vagy itt van pl. a rendszer azonnali leállítását eredményező exit() metódus, stb.

String, StringBuffer osztályok:

szövegek tárolására, manipulálására alkalmas osztályok.

A String típusú objektum állapota nem változtatható, a StringBuffer típusú objektumok állapota változtatható.

JAR ÁLLOMÁNYOK

A kész Java programot át kell adni a felhasználónak.

Kényelmes megoldás: a futtatáshoz szükséges fájlokat összecsomagoljuk, és egyetlen állományként adjuk át a megrendelőnek.



JAR (Java ARchive) állományok
szabványos ZIP formátumú tömörített állományok
(*zip* helyett *jar* kiterjesztéssel).

JAR ÁLLOMÁNYOK

Egy Java alkalmazás (applet) csak akkor fut, ha az illető gépen telepítve van a JRE.

Elvileg lehet exe állományt készíteni, de a kész Java programot .jar fájlként szokás átadni a felhasználónak:
JAR (Java ARchive) állományok (szabványos ZIP formátumú tömörített állományok)

A JAR formátum előnyei:

Biztonságos: Ellátható digitális aláírással.

Egyszerű letölthetőség.

Tömörítés: Hatékony tárolás, különböző meta-információkkal.

stb...

JAR ÁLLOMÁNYOK

Egy JAR állomány tartalmazhat:

bájt kódokat (class állományokat) amelyek osztályokat, interfészeket tartalmaznak

könyvtárakat (amelyek fizikailag valósítják meg a csomagok hierarchiáját)

erőforrásokat (képeket, dokumentumokat, hangokat, stb.)

Bizonyos JAR állományok futtathatók, mások nem.

A futtatható JAR állományoknak kell, hogy legyen egy belépési pontja, vagyis egy statikus main metódust tartalmazó főosztálya, amelyről a JAR aláírás-állománya (**manifest file**) ad információt.

JAR ÁLLOMÁNYOK

JAR készítése:

a/ A JDK jar.exe programja segítségével:

Készítés parancs módban:

- kell egy **manifest** nevű állomány, amely megmondja, hogy mit lehet kezdeni a jar tartalmával. Ennek minimális tartalma:

Main-Class: SajátMainOsztályNeve

- Ezek után kiadható a következő parancs:

jar cvfm SajátProgi.jar manifest SajátProgiKönyvtára

vagy:

jar cvfm SajátProgi.jar manifest SajátFájl1 SajátFájl2 ...

JAR ÁLLOMÁNYOK

JAR készítése:

b/ A Netbeans automatikusan elkészíti ☺

Clean + Build után a project dist mappája:

...	<DIR>	2008.11.29 21:45	—	...	<DIR>	2008.11.29 21:46	—
[build]	<DIR>	2008.11.29 21:45	—	[javadoc]	<DIR>	2008.11.29 21:46	—
[dist]	<DIR>	2008.11.29 21:46	—	[Pelda.jar]	jar	6 551	2008.11.29 21:45 -a-
[nbproject]	<DIR>	2008.11.29 21:31	—	[README.TXT]	TEXT	1 444	2008.11.29 21:45 -a-
[src]	<DIR>	2008.11.29 21:31	—				
[test]	<DIR>	2008.11.29 21:31	—				
build.xml	xml	3 352	2008.11.29 21:37 -a-				
manifest.mf	mf	85	2008.11.29 21:46 -a-				

Megjegyzés: Konzolos program .jar állománya parancsmódból futtatható. (java -jar fajNev.jar)

