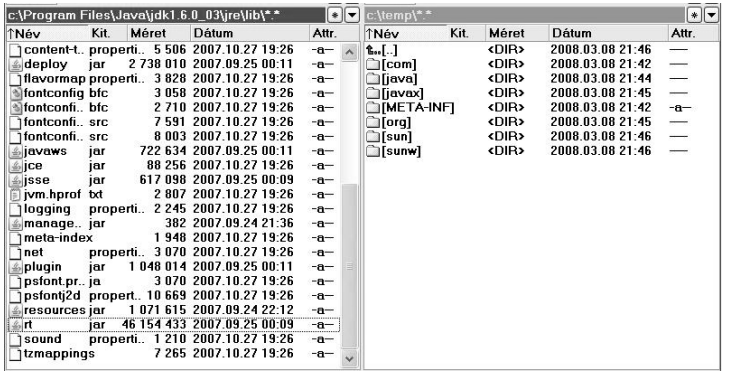
Programozás III

CSOMAG

CSOMAGOK – RT.JAR



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

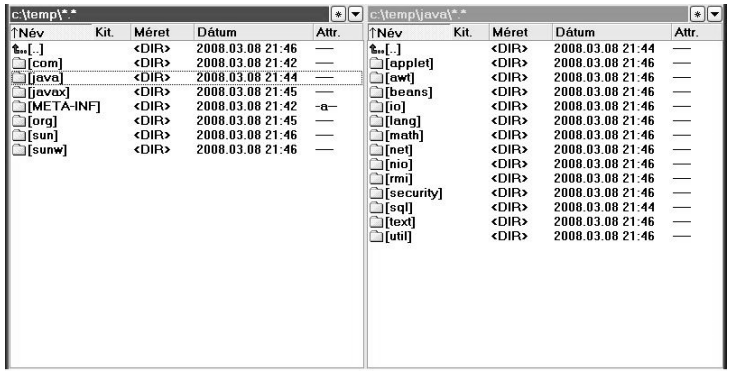
CSOMAGOK

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



Az rt.jar\java könyvtár tartalma

CSOMAGOK – RT.JAR

c:\temp\java\lang*.*

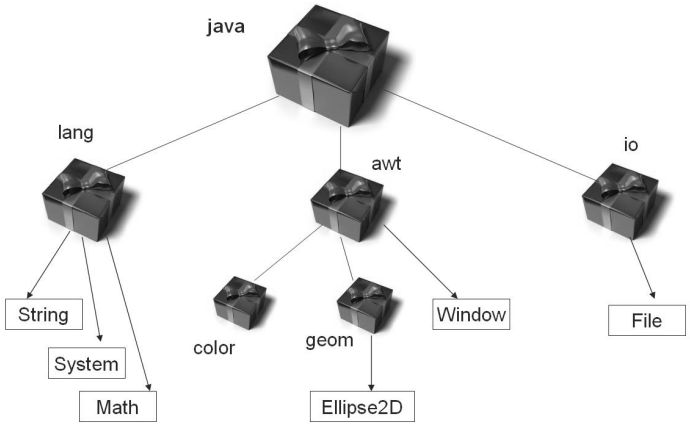
↑Név	Kit.	Méret	Dátum
[..]		<DIR>	2008.03.08
[annotation]		<DIR>	2008.03.08
[instrument]		<DIR>	2008.03.08
[management]		<DIR>	2008.03.08
[ref]		<DIR>	2008.03.08
[reflect]		<DIR>	2008.03.08
AbstractMethodError	class	330	2007.09.24
AbstractStringBuilder	class	10 332	2007.09.24
Appendable	class	344	2007.09.24
ApplicationShutdownH..	class	1 981	2007.09.24
ArithmeticException	class	318	2007.09.24
ArrayIndexOutOfBoun..	class	640	2007.09.24
ArrayStoreException	class	318	2007.09.24
AssertionError	class	1 403	2007.09.24
AssertionStatusDirecti..	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

c:\temp\java\math*.*

↑Név	Kit.	Méret	Dátum
[..]		<DIR>	2008.03.08 21
BigDecimal	class	23 230	2007.09.24 22
BigInteger	class	31 376	2007.09.24 22
BitSieve	class	1 991	2007.09.24 22
MathContext	class	3 403	2007.09.24 22
MutableBigInteger	class	12 883	2007.09.24 22
RoundingMode	class	1 577	2007.09.24 22
SignedMutableBigInte..	class	1 146	2007.09.24 22

A java.lang és a java.math könyvtár

CSOMAGOK - PÉLDA



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 - KÖNYVTÁRSTRUKTÚRA

Csomagok: logikai szerkezet

⇕

Könyvtárak: fizikai szerkezet

Csomagnév = könyvtárnév!!

- [java]
  - [applet]
  - [awt]
    - [geom]
      - *Area*
    - *Button*
  - [io]
  - [lang]
    - *String*
  - [javax]

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 - 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 - 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 - 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 deklaráló osztály érheti el
- **# protected**
  saját csomagból bárki és az utódok érhetik el

CSOMAGOK - PÉLDA

A **Mikulas.java** állomány

```
package unnep.teli;

public class Mikulas{

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

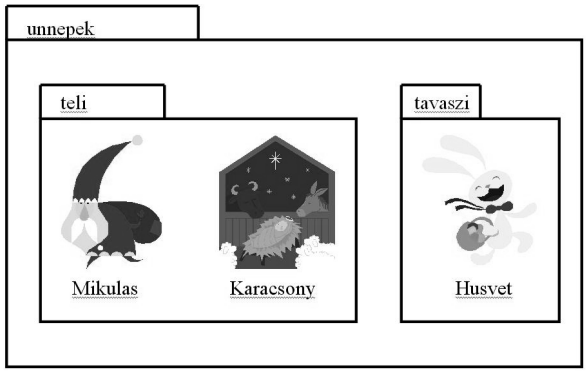
A **Karacsony.java** állomány

```
package unnep.teli;

public class Karacsony{

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

CSOMAGOK - PÉLDA



Csomag UML jelölése: téglalap kis „füllel”

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!
```

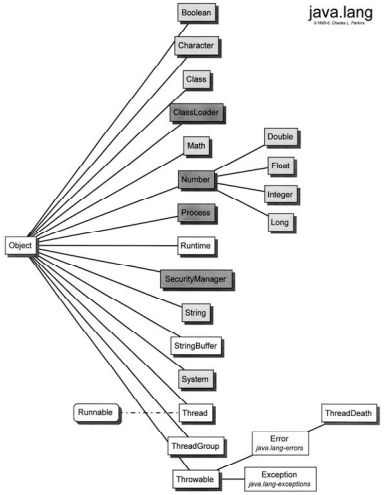
A JAVA.LANG CSOMAG – NÉHÁNY OSZTÁLY

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 evvel 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.)

CSOMAGOK – PÉLDA: A JAVA.LANG CSOMAG



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

Az Object osztály minden osztály közös őse.

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
```

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 (***J***ava ***AR***chive) állományok
szabványos ZIP formátumú tömörített állományok
(*zip* helyett *jar* kiterjesztéssel).

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 leállását 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

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 (***J***ava ***AR***chive) 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:

bájtkódokat (class állományokat) amelyek osztályokat, interfészeket tartalmaznak

könyvtárakat (amelyek fizikailag valósítják meg a csomagok hierarchiáját)

erőforrásokat (képeket, dokumentumokat, hangokat, stb.)

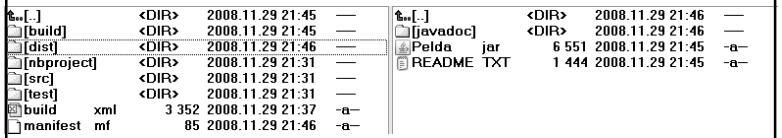
Bizonyos JAR állományok futtathatók, mások nem.
A futtatható JAR állományoknak kell, hogy legyen egy belépési pontja, vagyis egy statikus main metódust tartalmazó főosztálya, amelyről a JAR aláírás-állománya (**manifest file**) ad információt.

JAR ÁLLOMÁNYOK

JAR készítése:

b/ A Netbeans automatikusan elkészíti ☺

Clean + Build után a project dist mappája:



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

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 …