Java tutorial

I. Bevezetés

Történelem

- Sun Microsystems, Inc.
- Java (1995)
- JDK 1.0 (1996)
- JDK 1.1 (1997)
- Java 2, JDK 1.2 (1998)
- JDK 1.3 (2000)

Áttekintés

- Objektum-elvű imperatív nyelv
- A C/C++ nyelvekhez hasonlító szintaxis
- Egyszerűség
- Interpretált jelleg, bájtkód
- Hordozhatóság ("write once, run everywhere")
- WEB technológia
- Appletek
- Lassúság, erőforrás-igényesség

Hello World!

```
class Hello {
  public static void main( String[] args ){
    System.out.println("Hello World!");
  }
}
```

Építőkockák

- Az OOP elemei
  - Osztályok (típusok)
    - Adattagok
    - Műveletek
- Tagolás magasabb szinten: csomagok
  - Azaz modularitás...
  - Blokkstrukturáltság részleges támogatása
- Párhuzamosság: végrehajtási szálak
- Végrehajtás: programok és appletek
- Kivételek

Az OOP elemei

- Objektumok, **osztályok** (adatközpontú)
- Eseményvezérelt programozás
  - vs. strukturált programozás
  - deklaratív / *imperatív*
- Adatabsztrakció (egységbe zárás, adatelrejtés)
- Polimorfizmus
- Öröklődés
- Dinamikus kötés

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Verem osztály

```
class Verem {

  Object[] adatok;
  int veremtető = 0;
  int maxméret;

  public Verem( int maxméret ){
    this.maxméret = maxméret;
    adatok = new Object[maxméret];
  }

  public void push( Object o )
  throws VeremMegteltException {
    if( veremtető < maxméret ){
      adatok[veremtető] = o;
      veremtető ++;
    } else throw new VeremMegteltException(o);
  }

  ...
}
```

Verem osztály

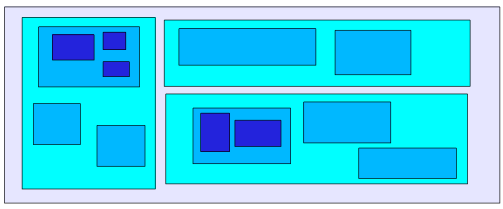
```
class Verem {

  Object[] adatok;
  int veremtető = 0;
  int maxméret;

  public Verem( int maxméret ){
    this.maxméret = maxméret;
    adatok = new Object[maxméret];
  }

  public void push( Object o )
  throws VeremMegteltException {
    if( veremtető < maxméret ){
      adatok[veremtető] = o;
      veremtető ++;
    } else throw new VeremMegteltException(o);
  }

  ...
}
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

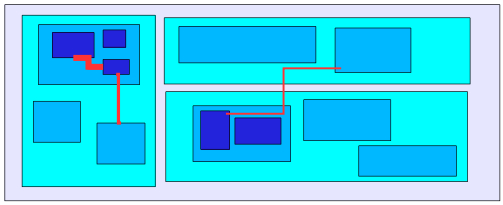
Tagolás: modularitás

- A programot komponensekre bontjuk
- A rendszert alrendszerekre bontjuk



Tagolás: modularitás

- A programot komponensekre bontjuk
- A rendszert alrendszerekre bontjuk



Java tutorial

-

Program tagolása Java-ban

- Csomagok: **`package`**
- Osztályok: **`class`**

- Egy osztály: szorosan összetartozó adatok és műveletek
- Az egy csomagba kerülő osztályok logikailag jobban összetartoznak, mint a különböző csomagokba kerülő osztályok

- Egyelőre ennyi elég... :-)

Példa csomagok használatára

```
package énkiscsomagom;
class Verem { ... }
```

```
package másikcsomagom;
import énkiscsomagom.*;
class A {
  Verem v;
  ...
}
```

Kódolási konvenciók: azonosítók neve

- változók neve
  `fizetés, nyugdíjKorhatár`
- "konstansok" neve
  `PI (Math.PI)` `TT_NUMBER`
- metódusok neve
  `lekérdez()` `fizetéstEmel()`
- típusok neve
  primitív: `int, boolean, ...`
  osztály: `Verem` `VeremMegteltException`
- csomagok neve
  `java.io`

Kódolási konvenciók: a kapcsos zárójelek

```
public Object pop() throws ÜresVeremException {
    if( veremtető > 0 ){
      veremtető --;
      return adatok[veremtető];
    } else throw new ÜresVeremException();
}
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Párhuzamosság

- Egy programon belül több végrehajtási szál lehet
- A feladat logikai darabolásából származnak
  - Kényelmesebb lehet leírni úgy a programot, hogy több szálat írunk
- (Végrehajtási) szál: thread
- Támogatás: a Thread osztályon keresztül
- Kérdések: ütemezés, kommunikáció, interferencia
- Például:
  - az egyik szál a felhasználói felülettel foglalkozik
  - a másik szál a hálózaton keresztül kommunikál valakivel

Példa párhuzamosságra

```
class A extends Thread {
  public String név;
  public void run(){
    System.out.println(név);
  }
}


class B {
  public static void main(String[] args){
    A x = new A(); x.név = "Jancsi"; x.start();
    A y = new A(); y.név = "Juliska"; y.start();
  }
}
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Programvégrehajtás

Program (applikáció, application)

- olyasmi, mint egy közönséges program
- igaz, hogy egy interpreter segítségével, de azért mégiscsak hagyományosan futtatjuk

Applet (applet)

- HTML oldalba ágyazott kis programocska
- A Web-böngésző futtatja

Szervlet (servlet)

- HTTP szerverbe ágyazott kis programocska
- A HTTP szerver futtatja

Kivételek

- *a kivételkezelést támogató nyelvi elemek nagyon hasznosak*
- kivétel: a programot a normális / átlagos végrehajtási menetétől eltérítő esemény (pl. futási idejű hiba)
- a kivételek lekezelését végző programkód szeparálása a normális / átlagos végrehajtást leíró kódtól
  - olvashatóság, karbantarthatóság, továbbfejleszthetőség
  - megbízhatóság, stabilitás
- kivételfajták megkülönböztetése
  - reprezentálása Java-ban: osztályokkal
- hierarchiába szervezhetők ezáltal
- kivételek: mint egy speciális visszatérési érték

Verem osztály

```
class Verem {

  Object[] adatok;
  int veremtető = 0;
  int maxméret;

  public Verem( int maxméret ){
    this.maxméret = maxméret;
    adatok = new Object[maxméret];
  }

  public void push( Object o )
  throws VeremMegteltException {
    if( veremtető < maxméret ){
      adatok[veremtető] = o;
      veremtető ++;
    } else throw new VeremMegteltException(o);
  }

  ...
}
```

Verem osztály

```
class Verem {

  Object[] adatok;
  int veremtető = 0;
  int maxméret;

  public Verem( int maxméret ){
    this.maxméret = maxméret;
    adatok = new Object[maxméret];
  }

  public void push( Object o )
  throws VeremMegteltException {
    if( veremtető < maxméret ){
      adatok[veremtető] = o;
      veremtető ++;
    } else throw new VeremMegteltException(o);
  }

  ...
}
```

Kivétel osztály definiálása

```
class VeremMegteltException extends Exception {

  public Object nemFértBele;

  public VeremMegteltException( Object o ){
    nemFértBele = o;
  }

}
```

Tervek a továbbiakra

- A nyelv maga
  - alapok: deklarációk, utasítások, kifejezések, vezérlési szerkezetek
  - az objektum-elvű programozás eszközei
  - csomagok
  - kivételkezelés
  - párhuzamosság
- Könyvtárak
  - bemenet/kimenet
  - grafikus felhasználói felületek
  - appletek
  - hálózatkezelés
  - stb.

Java tutorial

-

II. A programozási környezet

Letöltés

- Az eredeti, Sun, azaz JavaSoft: http://java.sun.com/ vagy http://www.javasoft.com/
  - Install anyag (Linux, Windows, Solaris) http://www.javasoft.com/j2se/1.3/
  - Dokumentáció a szabványos könyvtárakról http://www.javasoft.com/j2se/1.3/docs.html
  - Online olvasható dokumentáció http://www.javasoft.com/j2se/1.3/docs/
- Linux-os változathoz install, dokumentáció
  - http://www.blackdown.org/
  - ftp://xenia.sote.hu/pub/mirrors/java.blackdown.org/
- jikes: egy gyors Java fordító http://oss.software.ibm.com/developerworks/opensource/jikes/

Olvasnivaló

- Java 2 Útikalauz programozóknak Nyékyné G. J. (ed.) et al.
- Hálózati alkalmazások készítése (Csizmazia Balázs)
- http://java.sun.com/docs/books/
- Java Felhasználók Társasága http://java.sch.bme.hu/

Az első program: Hello World!

```
class Hello {
  public static void main( String[] args ){
    System.out.println("Hello World!");
  }
}
```

- Elmenteni a **Hello.java** nevű fájlba.

```
$ javac Hello.java
$ ls
Hello.class    Hello.java
$ java Hello
Hello World!
$
```

javac

java

Fordítás és futtatás

- Fájlnév: osztálynév.`java`
- `javac` osztálynév.`class` bájtkódra fordít, klassz fájl
- `java` interpreter

```
$ javac Hello.java
$ ls
Hello.class    Hello.java
$ java Hello
Hello World!
$
```

Fordítás és futtatás

- Fájlnév: osztálynév.`java`
- `javac` osztálynév.`class` bájtkódra fordít, class fájl
- `java` interpreter

```
$ javac Hello.java
$ ls
Hello.class    Hello.java
$ java Hello
Hello World!
$
```

Rövidítések

- Fejlesztői környezet
  - **J**ava **D**evelopment **K**it
  - Java 2 **S**tandard **D**evelopment **K**it
- Futtató rendszer: **J**ava **R**un-time **E**nvironment
- Interpreter: Java Virtuális Gép (**J**ava **V**irtual **M**achine)

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

III. Az alapok

Tartalom

A procedurális programozás alapfogalmai

- Azonosítók
- Alaptípusok
- Literálok
- Kifejezések
- Utasítások
- Változódeklarációk
- Értékadások
- Vezérlési szerkezetek
- Alprogramok
- Megjegyzések

Már megint: Hello World!

```
class Hello {
  public static void main( String[] args ){
    System.out.println("Hello World!");
  }
}
```

A C(++) nyelvhez hasonlító szintaxis

```
class Hello {
  public static void main( String[] args ){
    System.out.println("Hello World!");
  }
}
```

```
#include <stdio.h>
int main(){
    printf("Hello World!\n");
    return 1;
}
```

A C(++) nyelvhez hasonlító szintaxis

```
import java.io.*;
class Hello {
  public static void main( String[] args ){
    System.out.println("Hello World!");
  }
}
```

```
#include <stdio.h>
void main( int argc, char* argv[] ){
  fprintf(stdout,"Hello World!\n");
}
```

Hogyan is írjunk programot?

```
class Hello {
  public static void main( String[] args ){
    System.out.println("Hello World!");
  }
}
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Szám kiírása a képernyőre

```
class Hello {
  public static void main( String[] args ){
    System.out.println(42);
  }
}
```

- Szövegek
- Számok
- Logikai értékek
  (Próbáld ki pl. true-val!)
- stb.

Absztrakt programok kódolása

- Állapottér — változók neve és típusa
- Értékadás
- Programkonstrukciók — utasítások
  - szekvencia
  - elágazás
  - ciklus
- Struktogrammok — alprogramok
- Típusmegvalósítás

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Azonosítók

- karakterek: betűk _ $ számok
  - első karakter: betű _ $
- betű: két bájton, **Unicode**
  - pl. `nevező` `nevez\u0151`
  - nem `nevezo` vagy `nevező`
- foglalt szavak:
  - kulcsszavak (`if`, `while`, `int`, stb.)
  - nem használt: `const`, `goto`
  - predefinit literálok: `null`, `true`, `false`

Karakterkészlet

- Betűk ábrázolása két bájton, **Unicode**
- pl. `nevező`
  `nevez\u0151`
- nem `nevezo` vagy `nevezô`

- ASCII (7 bit) Latin-1 és Latin-2 (8 bit)
- C-ben Latin-1 szövegliterálok
- Ada95-ben Latin-1 azonosítók

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Típusok

- primitív típusok
  - boolean (true vagy false)
  - char (16 bites Unicode karakter)
  - byte (8 bites előjeles egész)
  - short (16 bites előjeles egész)
  - int (32 bites előjeles egész)
  - long (64 bites előjeles egész)
  - float (32 bites lebegőpontos szám)
  - double (64 bites lebegőpontos szám)

`!  &&  ||`

`+  -  *  /  %`
`<  >  <=  >=`
`&  |  <<  >>`

- osztályok, interfészek

Változó-deklarációk

- `int i;`
- `int i, j;`
- `int i = 1;`
- `int i, j = 0;` `int i; int j = 0;`

deklaráció-utasítás

inicializáló értékadás

olvasás írás előtt: fordítási hiba

Példa: olvasás írás előtt

```
int x;              int x;
int y = x;          x = 3;
x = 3;              int y = x;
```

```
int i;
if (42==42) i=33;
System.out.println(i);
```

A programszöveg statikus elemzése...

Literálok
számliterálok

- Egész számok (int)
  dec.: `255` okt.: `0377` hex.: `0xff`
  long: `255L` (vagy `255l`, vagy `0xffL`)
  short, byte: explicit konverzió, pl. `(short)255`
- Lebegőpontos számok (double)
  `12.3` `12.3e4` `12.3e4D`
  float: `12.3e4F`
  Nulla: `0.0` és `-0.0` is lehet (lásd `1.0/-0.0`)

Még literálok

- Logikai értékek: `true` és `false`
- Karakterek: `'K'`

`\n`	újsor	`\\`	\
`\t`	tabulátor	`\'`	aposztróf
`\b`	backspace	`\"`	idézőjel
`\r`	sorvissza	`\ooo`	karakter oktálisan
`\f`	lapdobás	`\uhhhh`	Unicode, hexadecim.

- Szövegek: "Helló Világ"

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

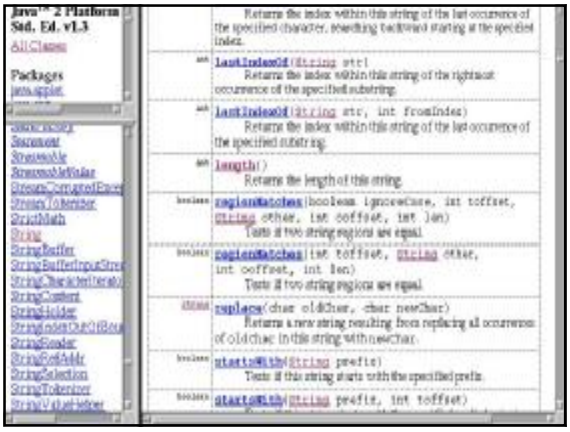
String hossza

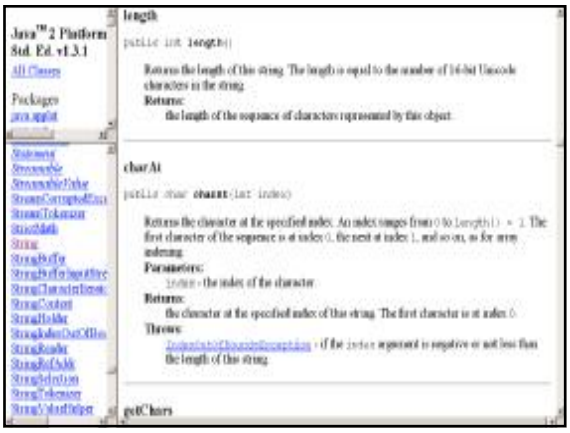
```
class Hossz {
  public static void main( String args[] ){
    System.out.println(args[0].length());
  }
}
```

```
$ javac Hossz.java
$ java Hossz
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException
    at Hossz.main(Hossz.java:3)
$ java Hossz elmebeteg
9
$
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás





Stringből szám

```
class Szoroz {
  public static void main( String args[] ){
    int x = Integer.parseInt(args[0]);
    int y = Integer.parseInt(args[1]);
    System.out.println(x*y);
  }
}
```

```
$ javac Szoroz.java
$ java Szoroz 10 11
110
$
```

Feladat

- Írj olyan programot, aminek egy parancssori argumentumot kell adni, ami egy N szám. A program határozza meg, hogy milyen hosszú a parancssori argumentum, mint String. Ezután írja ki a képernyőre a hossz és N szorzatát.

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Vezérlési szerkezetek

- Értékadás**ok**
- Blokk
- Elágazás (egyszerű és összetett)
- Ciklus (elöl- és hátultesztelős, léptetős)
- Ciklusmegszakítás
- Kilépés alprogramból

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Értékadások

- `i = 33;`
- `i += 2;  i = i+2;`

  `i -= 2;  i *= 2;  i /= 2;`
- `i++;  i += 1;  i = i+1;`

```
terület[i] += terület[0]*i;
```

Léptetős ciklus: **for**

```
for (int i=0; i<10; i++)
  System.out.println(i);
```

- A ciklusváltozót deklarálhatom a cikluson belül.
- Ilyenkor a ciklusváltozó lokális a ciklusra.

Léptetős ciklus: **for**

```
for (int i=0; i<10; i++)
  System.out.println(i);
```

```
int i;
for (i=0; i<10; i++)
  System.out.println(i);
System.out.println(i);
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Faktoriális számítás

```
class Faktorialis {
  public static void main( String[] args){
    int faktoriális = 1;
    for( int i = 1; i<=10; i++ )
      faktoriális = faktoriális * i;
    System.out.println(faktoriális);
  }
}
```

Faktoriális számítás

```
class Faktorialis {
  public static void main( String[] args){
    int faktoriális = 1;
    for( int i = 1; i<=10; i++ )
      faktoriális *= i;
    System.out.println(faktoriális);
  }
}
```

Faktoriális számítás

```
class Faktorialis {
  public static void main( String[] args){
    int faktoriális = 1;
    for( int i = 1; i<=10; i++ ){
      faktoriális *= i;
      System.out.println(faktoriális);
    }
  }
}
```

Faktoriális számítás

```
class Faktorialis {
  public static void main( String[] args){
    int faktoriális = 1;
    for( int i = 1; i<=10; i++ ){
      faktoriális *= i;
      System.out.println(i + "!=" +
                                 faktoriális);
    }
  }
}
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Feladat

- A faktoriálist számoló programot írjuk át úgy, hogy a legelső parancssori argumentum faktoriálisát számolja ki!
- Írjunk olyan programot, ami a parancssori argumentumait összeszorozza!

  A parancssori argumentumok számára így hivatkozhatunk:

  `args.length`

Blokk utasítás

- `{` és `}` közé írt utasítássorozat

  `{ int i; i=12; if (i>0) i++; }`

- írható mindenhova, ahol utasítás kell
  - pl. `for` után a `Faktorialis` programban
  - egymásba ágyazható
- deklarációs utasítások akárhol

  `{int i; i=12; int j=i+1; i++; }`

  változó hatásköre: deklarációtól a blokk végéig

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Egyszerű elágazás: **if**

```
if  ( i>j )   i = 1/(i-j);

if  ( i>j )   {
      int k = i-j;
      i = 1/k;
}

if  ( i>j )   i = 1/(i-j);
else          {i = 0; j = 0;}
```

If-then-else probléma

```
if (a==1)
  if (b==2)
    c = 1;
else
  c = 2;
```

```
if (a==1)
  if (b==2)
    c = 1;
  else
    c = 2;
```

```
if (a==1){
  if (b==2)
    c = 1;
} else
  c = 2;
```

Többágú elágazás

```
if( <feltétel-1> ){
   <utasítások>
} else if( <feltétel-2> ){
   <utasítások>
} else {
   <utasítások>
}
```

Nincs **elsif** vagy **elif** szerkezet

Feladat

- Írjuk át az eddig elkészített programokat úgy, hogy ellenőrizzék, hogy megfelelő számú parancssori argumentummal hívták-e.

Java tutorial

Összetett elágazás: **switch**

```
switch (billentyű) {
  case 'w': y++; break;
  case 'z': y--; break;
  case 'a': x--; break;
  case 's': x++; break;
}
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Összetett elágazás: **switch**

```
switch (billentyű) {
  case 'w': y++; break;
  case 'z': y--; break;
  case 'a': x--; break;
  case 's': x++; break;
  default :
    System.out.println("Hiba!");
}
```

Összetett elágazás: **switch**

```
switch (billentyű) {
  case 'w': y++; break;
  case 'z': y--; break;
  case 'a': x--; break;
  case 's': x++; break;
  default :
    System.out.println("Hiba!");
}
```

Tesztelős ciklusok

N = 10

```
int i = N;
while (i<10) {
   System.out.print(i);
   i++;
}

int j = N;
do {
   System.out.print(j);
   j++;
} while (j<10);
```

Feladat

- A faktoriálist számoló programot írjuk át while ciklusra!
- Számítsuk ki két szám legnagyobb közös osztóját!

A **break** utasítás

```
int i=0, j=0;
külső: for (; i<10; i++)
  for (j=0; j<10; j++)
    if (tömb[i][j] == 0)
      break külső;
```

Címke állhat utasítások előtt, pl. `külső:`

A **break** utasítás

```
int i=0, j=0;
külső: for (; i<10; i++)
  for (j=0; j<10; j++)
    if (tömb[i][j] == 0)
      break külső;
```

Címke állhat utasítások előtt, pl. `külső`.

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

A **continue** utasítás

```
int i=0, j=0;
külső: for (; i<10; i++)
  for (j=0; j<10; j++)
    if (tömb[i][j] == 0){
      System.out.println(i + " " + j);
      continue külső;
    }
```

A **continue** utasítás

```
int i=0, j=0;
külső: for (; i<10; i++)
  for (j=0; j<10; j++)
    if (tömb[i][j] == 0){
      System.out.println(i + " " + j);
      continue külső;
    }
```

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Alprogramok

```
class Lnko {
  public static void main( String[] args){
    if( args.length != 2 ){
      System.err.println("Hibás paraméterezés!");
    } else {
      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      while( a != b )
        if( a > b )  a -= b;
        else         b -= a;
      System.out.println("lnko = " + a);
    }
  }
}
```

Alprogramok

```
class Lnko {
  static int lnko( int a, int b ){
    while( a != b )
      if( a > b )  a -= b;
      else         b -= a;
    return a;
  }
  public static void main( String[] args){
    if( args.length != 2 ){
      System.err.println("Hibás paraméterezés!");
    } else {
      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      System.out.println("lnko = " + lnko(a,b));
    }
  }
}
```

Kilépés alprogramból

- A return utasítással
  - `return 3;` pl. int visszatérési típus esetén
  - `return;` void visszatérési típus esetén (természetesen nem kötelező)
- Ha a visszatérési típus nem void, akkor kötelező minden lehetséges végrehajtási ágon szerepelnie!
  - A programszöveg statikus elemzése...

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Feladat

- Írjuk át a faktoriálist számító programot úgy, hogy a számolást kiemeljük egy függvénybe!

Megoldás (1. lépés)

```
class Faktorialis {
  public static void main( String[] args){
    if( args.length == 0 ){
      System.err.println("Hibás paraméterezés!");
    } else {
      int n = Integer.parseInt(args[0]);
      int fakt = 1;
      for( int i = 1; i<=n; i++ )
        fakt *= i;
      System.out.println(fakt);
    }
  }
}
```

Megoldás (2. lépés)

```
class Faktorialis {
  static long faktoriális( int n ){
    long fakt = 1;
    for( int i = 1; i<=n; i++ )
      fakt *= i;
    return fakt;
  }
  public static void main( String[] args){
    if( args.length == 0 ){
      System.err.println("Hibás paraméterezés!");
    } else {
      int n = Integer.parseInt(args[0]);
      System.out.println(faktoriális(n));
    }
  }
}
```

Kifejezések

- Literálokból, változónevekből, operátorokból, függvényhívásokból, zárójelekből építhetők fel.
- Operátorok kiértékelési sorrendje jól definiált. (prioritás, asszociativitás, fixitás)
- C-hez hasonló operátorok, néhány eltérés. Semmi különös...

```
int y = 3*(4+3)/21 + 41;
int x = (y < 0) ? (y ++) : (y >> 2);
```

- A boolean típus különbözik az int típustól!

`false && true`	"hagyományos", lusta kiértékelésű "és"
`false & true`	mohó/szigorú kiértékelésű "és"
`3 & 5`	bitenkénti "és"

Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Megjegyzések

- `//` jeltől sor végéig
- `/*` és `*/` jelek közé írva
- dokumentációs megjegyzés, `/**` és `*/` jelek közé írva

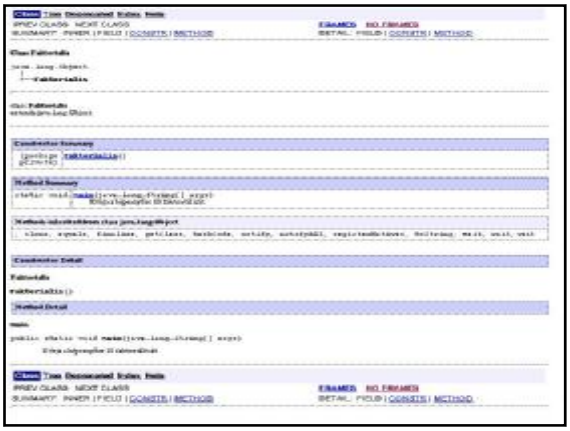
```
class Faktorialis {
  /** Kiírja a képernyőre 10 faktoriálisát */
  public static void main( String[] args){
    /* A paramétereket nem
       használom semmire */
    int faktoriális = 1;          // ebbe gyűjtöm
    for( int i = 1; i<=10; i++ )
      faktoriális = faktoriális * i;
    System.out.println(faktoriális);
  }
}
```

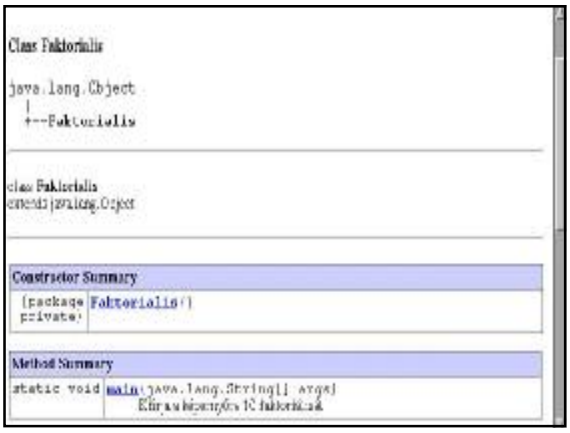
Java tutorial

- Copyright © 2000-2001, Kozsik Tamás

Dokumentációs megjegyzések

- A **javadoc** program segítségével HTML formátumú dokumentáció gyártható a Java forrásfájlokból.
- Ebbe a dokumentációba bekerülnek a dokumentációs megjegyzések.
- Különböző, ún. tag-ek segítségével paraméterezhetjük fel a dokumentációt.
  - `@param` paraméter dokumentálása
  - `@result` visszatérési érték dokumentálása
  - `@see` hivatkozás más osztályra
  - stb.
- például: `javadoc -private Faktorialis.java`





Class Faktorialis

java.lang.Object
|
+---Faktorialis

class Faktorialis
extends java.lang.Object

Constructor Summary	
(package private)	Faktorialis()

Method Summary	
static void	main(java.lang.String[] args) Ebben a példányban 10 faktoriálisát

Feladat

- Dokumentáljuk a faktoriálist és a legnagyobb közös osztót számoló programjainkat!

Java tutorial

-