Dr. Rohonczy János

Objektum-orientált programozás Java és C++ nyelven

ELTE, Általános és Szervetlen Kémiai Tanszék

Budapest, 2005.

Példaprogram

```
public class Proba {
  public static void main (String[] s) {          // belépési pont
    int i = Integer.parseint(s[0]);
    double f = factorial( i );
    System.out.println( "f=" + f );
  }
public static double factorial (int i) {
    if (i<0) return 0.;
    double f=1.0;
    while (i>1) {
      f *= i; i--;
    }
    return f;
  }
}
```


Példaprogram - 2

Java developer kit elérhetősége
http://www.javasoft.com

```
> javac Proba.java
> java Proba 4
f = 24.0
```


Azonosítók

Azonosítók

Változó (primitív, objektum), osztály
Metódus, paraméter
Címke

Unicode (0-ás lap ASCII)

1.Karakter: a... A... _ $ (Ne!) Szám nem lehet
Folytatás: fentiek + számjegyek + pénznem

Soha nem lehet írásjel vagy speciális karakter benne

Fenntartott szavak

- **Típusdeklaráció**

  boolean, byte, char, double, float, int, long, short, void

- **Típusmódosítók**

  abstract, final, native, static, synchronized, volatile, transient*

- **Érvényességi kört módosítók**

  public, protected, private

Fenntartott szavak - 2

- **Programstruktúrák**

  do, while, break, continue, if, else, for, switch, case, default, return, goto*

- **Kivételkezelés**

  try, catch, finally, throws, throw

- **Változó értéke**

  false, true, void

- **Operátor**

  instanceof

Fenntartott szavak - 3

- **Objektumok**

  class, interface, extends, implements, package, import, new, this, super

- **Fenntartott metódusnév**

  clone, equals, finalize, getclass, wait, hashcode, notify, notifyall, tostring

- **Egyéb fenntartott szavak***

  byvalue, const, future, generic, inner, operator, outer, rest, var, transient

Primitív adattípusok

• boolean	1 bit	értéke: true / false
• char	16 bit	unicode
• byte	8 bit	
• short	16 bit	
• int	32 bit	
• long	64 bit	
• float	32 bit	
• double	64 bit	

Primitív típusok - 2

- char

  'a' 'A' '\t' '\"' "
  \b \t \n \f \' \" \\ \xxx (x: octal) \073
  \uxxxx (x: haxadecimal) \u00f2

- int

  -5 342 0xff 0377 0xcac2

- long

  543L 0xffl

Primitív típusok - 3

- float

  0.55 -0.34 56. 1.E6 -6.02e-13 31.F

- double

  0532; -1.E5d

  double min = double.MIN_VALUE

  double inf = double.POSITIVE_INFINITY

Típuskonverzió

N: nem A: automatikus C: casting (int i = (int)5.343;)

mit \ mivé	bool	byte	char	int	long	float	double
bool	-	N	N	N	N	N	N
byte	N	-	C	A	A	A	A
char	N	C	-	A	A	A	A
int	N	C	C	-	A	A*	A
long	N	C	C	C	-	A*	A*
float	N	C	C	C	C	-	A
double	N	C	C	C	C	C	-

Paraméterátadás

- Érték szerint:

  primitív típusok

  Pl.: `metódus1 (int i, double d);`

- Cím (referencia) szerint:

  tömb, objektum

  Pl.: `metódus2 (double[] dd, Rectangle ra1);`

Operátorok precedenciája - 1

0.	. []	Objektum-, tömbelem	LEGMAGASABB
1.	++ -- + - ~ ! (típus)	pre- és post-inkrementálás i++ előjel bitenkénti negálás logikai negálás casting és zárójelezés	
2.	* / %	szorzás, osztás, maradék	
3.	+ -	összeadás, konkatenálás, kivonás	
4.	<< >> >>>	aritmetikai shift (előjel lép be) (zéró lép be)	

Operátorok precedenciája - 2

5.	< <= > >= instanceof	logikai műveletek típuskomparálás
6.	== !=	primitívekre és objektumokra
7.	&	bitenkénti AND
8.	^	bitenkénti XOR
9.	|	bitenkénti OR
10.	&&	aritmetikai AND

Operátorok precedenciája - 3

11.	||	aritmetikai OR	
12.	? :	feltételes értékadás a = (x>y) ? x : y;	

13.	= += -=
	*= /= %=
	<<= >>=
	>>>=
	&= ^= |=
		értékadások
		pl. a = a + 3;
		a+= 3;
			LEGALACSONYABB

Struktúrált programozás

Elágazás, ciklusszervezés, szinkronizáció, kivételkezelés

- **if - else if - else, switch - case - default,**
- **while, do - while, for**
- **synchronized**
- **try - catch - finally**

if - else if - else struktúra

```
int a=1, b=4, c=0;

if (a<b) c=a; else c=b;

if (a<b) {
   c=a;
   System.out.println(”igaz”);
}
else if (a==b) {
   c=b;
   System.out.println(”egyenlők”);
}
else {
   System.out.println(”a nagyobb”);
}
```


switch - case - default struktúra

```
switch (i) {
   case a:
       System.out.println(a);
       break;
   case 3:
   case 5:
      System.out.println(3);
      break;
   default:
      System.out.println(“más”);
}
```


while és do - while struktúrák

```
int a=3; b= 5;
while (a<b) {
  System.out.println(”a=” + a);
  if (a==3) break;
  a++;
}


do {
  System.out.println(”a=” + a);
  a++;
} while (a<b);
```


for struktúra

```
cimke:
for (int i=0;   i<5;   i++) {
    if (i==3) {
        break cimke;
    }
    if (i==2) {
        continue;
    }
    System.out.println(“i=” + i);
}


for (int i=0,  k=2;   i<5;   i++,  k += 3) {}
```

Kilépés struktúrából

break	kiléptet	*switch, while, do, for*	
continue	új ciklust indít.	*while*	elöl vizsgál
		do	végén vizsgál
		for	inkrementál és elöl vizsgál

Kilépés metódusból - *return*

```
public int oszt (int i, int j) {
  if (j==0) {
     return 0;
  }
  return i / j;
}

public void kiir (int i) {
  if (i==0) return;
  System.out.println("i=" + i);
  //(implicít return)
}
```


Szinkronizáló struktúra

```
public static void rendez (int[ ] a) {
   synchronized (a) {
       //védett blokk, kritikus rész
       // ide jön a sorbarendezés kódja
   }
}
```


Kivételkezelő struktúra

```
try {
    fileOlvaso("readme.txt");
}
catch (IOException ioHiba) {
    System.out.println("I/O hiba történt:" +
ioHiba);
}
catch (Exception masHiba) {
    System.out.println("Más hiba történt:" +
masHiba);
}
finally {
    System.out.println("Ezen túlvagyunk");
}
```

Konzol-applikáció írása

1. main() belépési pont és parancssori arg.

```
public class Test {
   public static void main(String[] s) {
   ...
   }
   ...
}
```

>java Test arg1 arg2
s[0] <-- arg1

2. Standard output használata

```
System.out.println("i="+i);
```

3. Egyszerű beolvasás

```
try {
   int i=System.in.read();              //  <-
'A'
   System.out.println(i+":"+(char)i);     //  ->
65
 }
catch (IOException ioe) {
 }
```


4. Fejlettebb beolvasás

```
import java.io.*;                                  // file legelején
.....
try {
  BufferedReader bin = new BufferedReader (
              new InputStreamReader (System.in));
  String line=bin.readLine().trim();
  System.out.println(line);
}
catch (IOException ioe) {
  ;
}
```


String literálok

```
String name = "Öcsi";

System.out.println("Hahó " + name);

String sor  = "  \" Itt vagyok\"  - mondta Öcsi.";
String sokSor     = "Így indul,
                  így végződik";          // TILOS

String sokSor     = "Így indul" +
                  "így végződik";         // OK

String = null;
String = "";
```

java.lang.String objektum

```
String s = new String ("abc") ;
s = "ABCABC" ;
s = "" ;
s = null ;
char c = s.charAt(0) ;
int ho = s.length();                    // METÓDUS !!!
int i= s.indexOf("BC");
i=s.lastIndexOf("BC");

boolean b = s.startsWith("AB");
b = s.equalsIgnoreCase("abcabc");

String t = s.substring(2, ho);
t = s.toLowerCase();
t = s.trim();
```


Stringből szám kiemelése

```
try {
  double d1 = Double.parseDouble(line);      // primitív
  Double d2 = Double.valueOf(line);          // Objektum
  d1 = d2.doubleValue();
}
catch (NumberFormatException nfe) {
  ;
}
```


Objektumok

Szöveg /String/

Tömb /array/

Egyéb objektumok

Tömbök

// Deklarálás

```
int [ ] ii  = new int[10];
String[ ] s = new String[4];
double [ ] d      = {1.,2.,3.};
String[ ] s2      = {”Hahó”,”Öcsi”}
```

// Értékadás - index 0-val kezdődik !!!

```
for (int i=0;i<ii.length;i++) {
  ii [ i ] = 2 * i ;
}
for (int i=0;i<4;i++) {
   s[i] = ”a” + ’b’;
 }
```

Többdimenziós tömbök

Deklarálás

```
int [ ][ ] i = new int[10][5];
int [ ][ ] j = {{1,0,0},{0,1,0},{0,0,1}};
 //újszerű
int [ ][ ] k = new int[10][ ];
for (int m=0; m<10; m++) {
  k [ m ] = new int [ m ];
//Deklaráció !
}
```

Értékadás

```
k [3][i]=123;
```

A tömbméret - tulajdonság és nem metódus!!!

```
int ho = k.length;
```


Tömb másolása

```
//a,
int [ ] k = {1,0,0};
int [ ] kcopy = k;              //ugyanoda hivatkozik,
nincs másolat

//b,
int [ ] kcopy=new int [ k.length ];
//másolat készül
for (int i=0;  i<k.length;  i++) {
  kcopy[ i ] = k[ i ];
}

//c,
int[] kcopy= (int [ ]) k.clone();  //Később részletesebben
```

- Cloneable interface implementálásával.
- a klónozás nem rekurzív.
- implicít casting kell.

Objektumok és osztályok

```
public class Circle {
  public static final double PI =  3.14159;
  public double r;
  public static double radiansToDegrees( double rads) {
    return rads * 180. / PI;
  }
  public double terulet () {
    return PI * r * r;
  }
  public double kerulet () {
    return 2 * PI * r ;
  }
}
```


Osztályszintű hozzáférés

```
//Osztályváltozóhoz  - static final
double pi = Circle.PI;

//Osztálymetódushoz - static
double d = Circle.radiansToDegrees(2.0);

//Névkonvenciók - erősen ajánlott
      konstansok (final)     - csupa nagybetű
      osztály és konstruktor - Nagybetűvel kezdődik
      változó és metódus     - kisbetűvel kezdődik

      konstruktor és metódusnév után () kötelező.
```

Példányszintű hozzáférés

```
Circle c = new Circle();          // példányosítás
c.r = 3.0;                  // értékadás

Circle d = new Circle();
d.r = c.r + 1.0;            // művelet változóval

double a = c.terulet();          // objektum-
orientált stílus
                            // művelet metódussal

// régi, procedurális stílus - ROSSZ
double a = terulet(c);
```


Sajátpéldány - this kulcsszó

```
public double terulet () {
  return Circle.PI * this.r * this.r ; //itt nem
kötelező a this
}

public void setRadius (double r) {
  this.r = r ;                        //itt KÖTELEZŐ
}
```

jobb megoldás

```
public void setRadius (double _r) {
  r = _r ;                          //ez így OK
}
```


Példányosítás, konstruktorok

```
public class Circle {
  public static final double PI =  3.14159;
  public double r;
  public Circle () {
  }
  public Circle (double _r) {
     this.r =_r;
  }
  //.....
}
```

Konstruktorok
- Kötelező névegyezés az osztálynévvel
- Nincs visszatérési típus
- Lehetőleg inícializáljon

Konstruktorok használata

```
Circle c1 = new Circle();
c1.r=0.25;
c1.setRadius(0.25);

Circle c2 = new Circle(0.25);
```

Iníciáló blokkok

```
public class Circle {
  public static final double PI;
  public double r;
  static {                    // Osztályszintű iníc.
     PI=3.14159;
  }

  {                           // Példányszintű iníc.
     r = 2.5;
  }
  public Circle () {                  //Rejtett
<clinit> metódus
     ......
  }
}
```


Metódusnevek túlterhelése

```
public class Idom {
   public double terulet (double a) {              //négyzet
      return a*a;
   }
   public double terulet (double a; double b) {    //téglala
      return a*b;
   }
}

Idom idom = new Idom();
double terulet1 = idom.terulet(5.);
double terulet2 = idom.terulet(5.,6.);
```


Csomagok - 1

- Egy file-ban egy publikus osztály (class)
  kötelező névegyezés
- Egy könyvtárban levő osztályok =
  egy package (csomag)
- Egymás alatti alkönyvtárakban levő osztályok =
  osztály-hierarchia (namespace)
- Gyökérkönyvtárak megtalálása
  CLASSPATH környezeti változó
  -cp illetve -classpath argumentumok

Csomagok - 2

```
package elte.chem;
public class Proba {
    public static void main(String[] s) {
       System.out.println("Haho");
    }
    public static int szoroz (int a,int b) {
       return a*b;
    }
}
```

> cd c:\javaspeci\elte\chem
> javac Proba.java
> cd c:\javaspeci
> java -cp c:\javaspeci elte.chem.Proba

Csomagok - 3

```
package elte.info;
import elte.chem.Proba;
public class Proba2 {
    public static void main(String[] s) {
       System.out.println(Proba.szoroz(2,3));
    }
}
```

```
> cd c:\javaspeci\elte\info
> javac -classpath c:\javaspeci
Proba2.java
> cd c:\javaspeci
> java -cp c:\javaspeci elte.info.Proba2
```


Objektumos programnyelv jellemzői

Öröklés
: Egy objektum megkapja egy másik tulajdonságait

Adatrejtés - hozzáférési kategóriák
: Változó- és metódusnevek rejtése, érvényességi körének szűkítése

Polimorfizmus
: Egy objektum más objektumként viselkedhet (Automatikus típuskonverzió)

Öröklés - *extends*

Új osztály egy meglevő osztály kiegészítésével készül

```
public class Anya extends Object {      // szülőosztály
   int a;                               // öröklődik
   private int b;                       // öröklődik, de nem
hozzáférhető
   public void setTerulet(int i) { a=i*i; }
}

public class Utod extends Anya {  // gyermekosztály
    int kerulet;
    public int getTerulet () {
      return this.a;
    }
    public void setTerulet (int i, int j) {
       a=i*j;
    }
}
```


Konstruktorok öröklődése - nincs

Utód konstruktora meghívja valamelyik szülőkonstruktorát

```
public class Utod extends anya {
   public Utod (int i) {
      super(5,i);        // super() mindig első
helyen
      ......
   }
}
```

Legalább implicít szülőkonstruktor-hívás vagy hiba!

```
public Utod (int i) {
   // super();        //ha nincs kiírva,akkor a
fordító
                      a super()-t behelyettesíti
   ......
}
```

Konstruktorok öröklődése - 2

Nem példányosítható objektum konstruktora

```
public class Utod {
  public static final double PI=3.14159;
  private Utod () {
    ;
  }
}
```

Mire jó?

```
double a = Utod.PI;            //OK
Utod utod= new Utod();         // nem megengedett
```


Objektum lezárása

Nincs destruktor metódus - nincs explicit megszüntetés
Garbage collector automatikusan felszabadítja a nem használt objektumok tárfoglalását.
Előtte meghívja a finalize() metódust.

Törlés előtt szükség lehet

- nyitott file-ok lezárására
- network kapcsolat megszüntetésére
- temporális file-ok törlésére, stb.

```
protected void finalize() throws Throable {
      super.finalize();
      tempfile.delete();      //vagy hasonló
}
```


Hozzáférési kategóriák - 1

Nyilvános tag - bárhonnan hozzáférhető

```
package elte.chem;
public class Idom {
    public double terulet;          //
public
}
 public class Szamol {
   Idom idom = new Idom();
    idom.terulet = 25.;        // OK
}
```

```
package elte.info;
import elte.chem.Idom;
public class Szamol {
    Idom idom = new Idom();          // OK
     idom.terulet = 25.;             // OK
}
```


Hozzáférési kategóriák - 2

Félnyilvános tag - csak saját csomagból férhető el

```
package elte.chem;
public class Idom {
   double terulet;                  // nem public
}
public class Szamol {
   Idom idom = new Idom();
   idom.terulet = 25.;              // OK
}
```

```
package elte.info;
import elte.chem.Idom;
public class Szamol {
   Idom idom = new Idom();          // OK
   idom.terulet = 25.;              // HIBA
}
```

Hozzáférési kategóriák - 3

Privát tag - kizárólag csak saját metódusok láthatják

```
package elte.chem;
public class Idom {
   private double terulet;              // private
}
public class Szamol {
   Idom idom = new Idom();
   idom.terulet = 25.;                  // HIBA
}
```

```
package elte.info;
import elte.chem.Idom;
public class Szamol {
   Idom idom = new Idom();              // OK
   idom.terulet = 25.;                  // HIBA
}
```


Hozzáférési kategóriák - 4

Protected (védett) tag

```
package elte.chem;
public class Idom {
   protected double terulet;            // protected
}

//azonos csomagon belül
public class Szamol {
   Idom idom = new Idom();
   idom.terulet = 25.;                  // OK
}
```


Hozzáférési kategóriák - 5

Protected (védett) tag

```
package elte.chem;
public class Idom {
   protected double terulet;            // protected
}
```

```
package elte.info;
import elte.chem.Idom;
public class Szamol {
   Idom idom = new Idom();              // OK
   idom.terulet = 25.;                  // HIBA
}
public class UjraSzamol  extends Idom {  // Leszármazott
   terulet = 25.;                       // OK
}
```


Hozzáférési kategóriák

	public	protected	package	private
Definiáló osztályban	**Igen**	**Igen**	**Igen**	**Igen**
Másik osztály azonos csomagban	**Igen**	**Igen**	**Igen**	**Nem**
Leszármazott osztály más csomagban	**Igen**	**Igen**	**Nem**	**Nem**
Idegen osztály más csomagban	**Igen**	**Nem**	**Nem**	**Nem**

Objektum polimorfizmusa

Adattagok	- elfedés
Statikus adattagok	- elfedés
Metódusok	- felülírás
Statikus metódusok	- elfedés

Adattag elfedése - 1

```
public class Idom {
   public int x;                          //szülő
}

public class Négyzet extends Idom {
   public int x;                          //gyerek
   public setX (int a) {
      x=a;                                //gyerek
      this.x=a;                           //gyerek
      super.x=a;                          //szülő
      ((Idom) this).x=a;                  //szülő
   }
}
```

static változók teljesen hasonlóan viselkednek

Adattag elfedése - 2

```
public class Idom {
   public int x;                          //1. objektum
}
public class SikIdom extends Idom {
   public int x;                          //2. objektum
}
public class Négyzet extends SikIdom {
   public int x;                          //3. objektum
   public setX (int a) {
      x=a;                                //3. objektum
      super.x=a;                          //2. objektum
      super.super.x=a;                    //HIBA
      ((Idom)this).x=a;                   //1. objektum
   }
}
```


Metódus felülírása - 1

```
class AObj {
   int i=1;
   int f() { return i; }
   static char g() { return 'A';}
}

class BObj extends AObj{
   int i=2;                               // elfedés
   int f() { return -i; }                 // felülírás
   static char g() { return 'B';}         // nincs felülírás - helyette elfedés
}
```

Érvényes: azonos szignatúra
nem statikus - példány - metódusokra

Metódus felülírása - 2

```
public class Test {
  public static void main (String[] s) {
     BObj b=new BObj( );
     System.out.println(b.i);                  // 2      - elfedés
     System.out.println(b.f( ));               // -2     - felülírás
     System.out.println(b.g());                // B      - elfedés

     // b legyen olyan, mint AObj
     System.out.println(((AObj) b).i);         // 1      - elfedés feloldása
     System.out.println(((AObj) b).f( ));      // -2     - felülírás érvényes !!!!!
     System.out.println(((AObj) b).g());       // A      - elfedés feloldása
  }
}
```

AObj f()-jét el kell felejteni, mert felülírtuk BObj-ban!

Metódus felülírása - 3

Felülírt metódus elérése

```
System.out.println(super.f( ));          // 1      - AObj f()-je él!
```

Vigyázat!

super.f() eredménye nem azonos ((AObj)this).f()-el

super.super.f(); nem megengedett

Absztrakt osztályok

Jellemzői:

- az osztályt abstract módosító jelöli
- benne abstract - törzs nélküli metódus(ok)
- az abstract metódus nem lehet private, final, static v. native
- nem lehet példányosítani
- kiterjesztett osztály lehet abstract vagy nem abstract
  - abstract: van még abstract metódusa
  - nem abstract: minden metódusát felüldefiniáltuk

Absztrakt osztály - 2

```
public abstract class Idom {
   private boolean keruletIsmert = false;
   private double kerulet;

   public double getKerulet() {
      if (!keruletIsmert) {
         kerulet = keruletSzamol();
         keruletIsmert = true;
      }
      return kerulet;
   }
   protected abstract double keruletSzamol();
}
```

Absztrakt osztály - 3

```
public class Negyzet extends Idom {
   private double a;
   public Negyzet (double _a) { this.a = _a; }
   protected double keruletSzamol() { return 4 * a; }
}

public class Kor extends Idom {
   private double r;
   public Kor (double _r) { this.r = _r; }
   protected double keruletSzamol() {
      return r * r * Math.PI;
   }
}

Negyzet negyzet1 = new Negyzet (5.);
System.out.println("kerulet=" + negyzet1.getKerulet());
```


Végleges osztályok és metódusok

- Végleges osztályt final class módosító jelöli
- Nem lehet kiterjeszteni

- Végleges metódust final minősítő jelzi
- Nem lehet felüldefiniálni
- Mindkét esetben a final és abstract egymást kizárják

```
public final class Kor {
   ...
}
public class Teglalap {
   public final double getTerulet() {
      ...
   }
}
```


Interfészek

Java-ban bármely osztálynak csak egy super ősosztálya lehet.
Egy osztály többféle eredetű metódust is implementálhat.
Megoldás: speciális ős: interface

```
public class Kocka extend Idom {
   ...
}
public interface Anyag {
   ...
}
public interface Gyártmány extends Termék{
  ...
}
public class Dobókocka extends Kocka implements Anyag, Gyártmány {
  ...
}
```


Interfész jellemzői

Típus: interface
Belsejében csak konstansok és absztrakt metódusok lehetnek
- Konstans alap minősítői: public static final
- Metódusok alap minősítői: public abstract
  - Nem írjuk ki a minősítőket
  - A konstansok mind nagy betűvel írandók
  - A metódusoknak csak a fejét írjuk meg

```
interface  AlmaIfc {
   String[] SZIN = {"piros","sárga","zöld"};          //static
   String getColor();
   void setUnitPrice (double price);
   double getPrice   (double mennyiseg);
}
```

Interfész implementálása

Az összes metódus implementálni kell

```
public class Golden extends Object implements  AlmaIfc {
   int color;
   double unitPrice=200.;
   public Golden (int _color) { this.color=_color; }
   public String getColor()    { return Alma.SZIN[color]; }
   void setUnitPrice(double _price) { unitPrice=_price;}
   double getPrice(double mennyiseg) { return menniseg * unitPrice;}
   }
}
```


Mire jó az interface?

- Közös metódus-felület egészen különböző objektumoknak
- Több interfész metódusainak implementálása egyszerre
- Az objektum interface-ként tud viselkedni (casting)

```
Golden golden =new Golden(1);
double ara=((AlmaIfc)golden).getPrice(1.0);
```

- Szerializációs gondok kiküszöbölhetők vele

Objektum kölcsönös hivatkozása

```
public class AClass {
   public int getB() {
      return BClass.getB();      // Fordítási hiba! Nincs még BClass
   }
   public static int getA() {
      return 1;
   }
}
public class BClass {
   public int getA() {
      return AClass.getA();      // Fordítási hiba! Nincs még AClass
   }
   public static int getB() {
      return 1;
   }
}
```


Fordítási hiba kiküszöbölése

```
public interface BClassIfc{
   public static int getB();
}
public class AClass {
   public int getB() {
      return BClassIfc.getB();                    // Itt kerüljük el a kereszthivatkozást
   }
   public static int getA() { return 1; }
}
public class BClass implements BClassIfc {     // Itt zárjuk a kört
   public int getA() {
      return AClass.getA();
   }
   public static int getB() { return 1;  }
}
```

Kivétel (Exception) objektumok

```
public class BajVanException extends Exception () {
  public BajVanException() {
    super("Baj Van");
  }
  public BajVanException(String theMessage) {
    super(theMessage);
  }
}
```


Kivétel eldobása

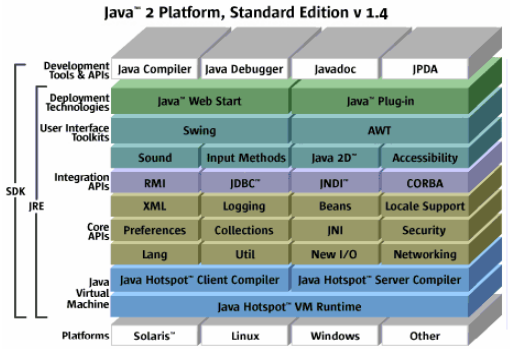
Metódushoz rendelt eszköz

```
public class Osztas {
  double a,b;
  public Osztas (double a, double b) {
    this.a=a; this.b=b;
  }
  public double oszt () throws BajVanException {
     if (b==0.) {
        throw new BajVanException("Nullaval oszt");
     }
     return a/b;
  }
}
```


Kivétel elkapása

```
...
public void osztunk() throws BajVanException,NullPointerException {
   Osztas osztas = new Osztas (5., 0.);
   try {
      double hanyados = osztas.oszt() {
   }
   catch (BajVanException bve) {              // itt elkapom
      System.out.println(bve);                // itt tovább dobom
      throw bve;
   }
}
...
```


Java2 platform szerkezete



Fontosabb Java2 csomagok - 1

java.applet	HTML appletekhez
java.awt	grafika
java.beans	grafikus objektum design
java.io	file műveletek
java.lang	adattípusok, op.rendszer
java.lang.reflect	tömb natív hozzáférés
java.math	banki matematika
java.net	network használata

Fontosabb Java2 csomagok - 2

- java.nio — új i/o műveletek - pufferkezelés
- java.sql — SQL kapcsolat ODBC/JDBC
- java.text — formattált szöveg/szám
- java.util — Vector, StringTokenizer, HashTable
- java.util.zip — ZIP tömörítés
- java.security — Java *'sandbox'* - biztonság

Különleges Java2 csomagok

- java.rmi — távoli metódushívás
- javax.crypto — titkosítás
- javax.rmi.CORBA — CORBA interface
- javax.swing — fejlett, swing grafika
- org.omg.CORBA — low level CORBA
- org.xml.sax — XML

Java 1.4.2 verzióban
135 szabványos objektumcsomag

java.lang.StringBuffer objektum

```
StringBuffer sb = new StringBuffer("ab");
sb.append("cd");
sb.insert(3,"bc");
sb.deleteCharAt(0);
```

java.io.* - *stream*-ek

stream (adatáram) csatorna(?)

Irány: bemeneti (szekvenciális input file)
kimeneti (szekvenciális output file)

Adattípus: byte
karakteres

Objektumok: alapfeladatok - irány szerint
kiegészítő funkciók

Absztrakt stream-ek

Abstract objektumok

Byte alapú
- **InputStream**
- **OutputStream**
- **RandomAccessFile**

Karakter alapú
- **Reader**
- **Writer**

Leszármaztatott osztályok →

Stream-ek és forrásaik

Forrás	*InputStream*	*OutputStream*
File	FileInputStream	FileOutputStream
Byte array	ByteArrayInputStream	ByteArrayOutputStream
Object	ObjectInputStream	ObjectOutputStream
Pipe (thread-ek)	PipedInputStream	PipedOutputStream
Filter	**FilterInputStream** BufferedInputStream DataInputStream PushbackInputStream	**FilterOutputStream** BuferedOutputStream DataOutputStream

Reader / Writer osztályok

Forrás	Reader	Writer
File	**InputStreamReader** FileReader	**OutputStreamWriter** FileWriter
Char array	CharArrayReader	CharArrayWriter
String	StringReader	StringWriter
Pipe (thread-ek)	PipedReader	PipedWriter
Filter	BufferedReader PushbackReader	BufferedWriter

Exception fajták

- IOException
  - EOFException
  - FileNotFoundException
  - InterruptedIOException
  - UnsupportedEncodingException
  - ObjectStreamException
    - InvalidClassException
    - NotSerializableException

java.io.File

```
File f =  new File("c:\java","proba.txt");
boolean test = f.isDirectory();
test = f.isFile();
test = f.isHidden();
test = f.canRead();
test = f.exists();
test = f.mkdir();
f.delete();
String path=f.getPath();
String[] list= f.listFiles();
```


java.io.Reader

```
try {
  FileReader fr = new FileReader("proba.ini");
  BufferedReader br = new BufferedReader ( fr );
  String line = "", param = "";
  do {
    line = br.readLine();
    if ( line.startsWith("A=")) {
       String param=line.substring(2,ho); break;
    }
  } while (line.length>0);
  br.close();
}
catch (IOException ioe) {;
}
```


java.io.Writer

```
try {
  FileWriter fw = new FileWriter("proba.ini");
  String line = "Hello\r\n";
    fw.write(line,0,line.length());
    char[] ch=line.toCharArray();
    fw.write(ch,0,line.length());

    fw.flush();
    fw.close();
}
catch (IOException ioe) {;
}
```

InputStream-ek

```
try {
   DataInputStream ds = new DataInputStream (
                              new BufferedInputStream(
                                 new FileInputStream("datafile") ) );
   int i     =  ds.readInt();
   byte b  =  ds.readByte();
   byte[] kbytes = new byte[1024];
   do {
       ds.read(kbytes);       //  ds.readFully(kbytes);
   } while (true);
   ds.close();
}
catch (IOException ioe) {;}
```


java.io.FilenameFilter

```
public class TextFilter implements FilenameFilter
   String end;
   public TextFilter (String txt) {                    // konstruktorban letesszük
      end=txt;
   }
   public boolean accept (File file, String name) {    // kötelező implementálni!
       boolean ok= file.isFile()   &&   name.endsWith(end);
       return ok;
   }
}
File f =  new File("c:\java","proba");                 // használata
public String[] listaz() {
    TextFilter tFilter = new TextFilter( ".txt" );
    String[] lista      = f.listFiles ( tFilter ) ;    // tFilter.accept()-et hívja minden file-ra
    return lista;
}
```


Könyvjelző-mechanizmus

```
int b; byte [] buf = {'H','e','l','l','o','!'};          // vagy buf = s.toCharArray();
try {
  ByteArrayInputStream bis = new ByteArrayInputStream (buf);
  int c = bis.available();                                //olvasható byte-ok alsó becslése
  for (int i  = 0;i<c/2;i++) {  b = bis.read(); System.out.print((char)b);  } //0-tól olvas
  bis.mark(c);                                                               //max. 6-ig
  for (int i = 0;i<c/2;i++) {   b = bis.read(); System.out.print((char)b);  } //3-tól olvas
  bis.reset();                                                               //vissza mark-ig
  for (int i = 0;i<c/2;i++) {   b = bis.read(); System.out.print((char)b);  } //3-tól olvas
  bis.close();
}
catch (IOException ioe) {;
}
```

Hello!lo!

Közvetlen elérésű file-ok

```
try {
  RandomAccessFile rfile = new RandomAccessFile("proba.txt","rw");
  int i=1;
  while ( i*4 < rfile.length() ) {
     rfile.seek(i*4);                  // pozíció
     int adat = rfile.readInt();
     rfile.seek(i*4);
     rfile.writeInt(adat+1);
     i*=3;
  }
  rfile.close();
}
catch (IOException ioe) {;}       // vagy EOFException
```

java.net csomag

- Hálózat kapcsolat alapja TCP/IP protokol
- IP cím 157.181.192.1
- IP név para.chem.elte.hu
- Protocol TCP összeköttetés-alapú
  UDP összeköttetés-mentes
- Portszám (és socket) 1-1024 védett, többi 65535-ig szabad
  TCP és UDP portok külön

UDP Szerver - csomagot fogad

```
import java.io.*;    import java.net.*;
public class Server {
     public static void main ( String[] s) {
       int port = 8888;
       try {
          byte[] uzenet =  new byte[1500];                    //puffer
          DatagramPacket packet = new DatagramPacket(uzenet, uzenet.length);
          DatagramSocket dSocket = new DatagramSocket(port);
          dSocket.receive(packet);                                    //itt várakozik
          String line = new String(uzenet,0,packet.getLength());   System.out.println(line);
          dSocket.close();
       }
       catch (UnknownHostException uh) {;}
       catch (IOException uh) {;}
     }
}
```


UDP kliens - csomagot küld

```
import java.io.*;    import java.net.*;
public class Client {
     public static void main ( String[] s) {
       int serverPort = 8888;
       try {
         DatagramSocket dSocket = new DatagramSocket();   //nincs portszám
         try {
           InetAddress cim = InetAddress.getByName("localhost");
           String line = new String ("Haho");
           byte[] msg =  line.getBytes();                    //puffer
           DatagramPacket p = new DatagramPacket(msg,line.length(),cim,serverPort);
           dSocket.send(p);                                          //itt küldjük ki
         } catch (UnknownHostException uh) {;}       catch (IOException uh) {;}
         finally {dSocket.close();}
       } catch (SocketException se) {;}
     }
}
```


java.net.InetAddress

```
InetAddress cim = InetAddress.getByName("para.chem.elte.hu");
InetAddress cim = InetAddress.getLocalHost ();
String name      = cim.getHostName();
String ipszam    = cim.getHostAddress();
byte[] ipBytes   = cim.getAddress();
```

java.net.DatagramPacket

```
byte[] buffer = new byte[1500];
DatagramPacket dp = new DatagramPacket(buffer);
int port=8888;
DatagramPacket dp = new DatagramPacket(
                          buffer,buffer.length(),inetAddress, port);
dp.setData(buffer);
dp.setLength(buffer.length());
dp.setPort(5555);
dp.setAddress(inetAddress);

byte[] data = dp.getData();
int hossz   = dp.getLength();
int  port   = dp.getPort();
InetAddress iAddress = dp.getAddress();
```


java.net.DatagramSocket

```
int port = 6630;
DatagramSocket dSocket = new DatagramSocket();
DatagramSocket dSocket = new DatagramSocket(port);

dSocket.receive(datagramPacket) throws java.io.IOException;
dSocket.send (datagramPacket) throws java.io.IOException;
dSocket.close();
int port = dSocket.getLocalPort();

dSocket.connect(inetAddress, port);          //belső biztonsági ellenőrzésre:
dSocket.disconnect();                        //több csomag ugyanoda
```


TCP kliens - 1

```
import java.io.*;
import java.net.*;
public class TClient {
  public static void main(String[] s) {
    serverPort=8888;
    Socket socket;
      try {
        socket = new Socket ("127.0.0.1",serverPort);
        PrintWriter out = new PrintWriter(socket.getOutputStream());
        BufferedReader in = new BufferedReader(
                    new InputStreamReader(socket.getInputStream()));
        out.println("Haho");
        out.flush();
```


TCP kliens - 2

```
        String valasz = in.readLine();    //   VÁRAKOZÁS
        System.out.println(valasz);
      }
      catch (IOException ioe) {;}
      finally {
        try {
          socket.close();
        }
        catch (Exception ex) {;}
      }
  }  //end of main
}  // end of TClient
```

TCP szerver - 1

```
import java.io.*;
import java.net.*;
public class TServer {
  public static void main(String[] s) {
    serverPort=8888;
    boolean kilep=false;
    Socket socket;
    try {
      ServerSocket svSocket = new ServerSocket(serverPort);
      while (kilep==false) {
        socket = svSocket.accept();      //   VÁRAKOZÁS
        BufferedReader in = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream());
        String line = in.readLine();
```


TCP szerver - 2

```
          try {
            int ertek = Integer.parseInt(line);
            String valasz = Integer.toString(ertek * ertek);
          }
          catch (NumberFormatException nfe) {;}
          out.println(valasz);
          out.flush();
          if (ertek == 0) { kilep = true;}
          socket.close();
        } //while
      }
      catch (IOException ioe) { ; }
      finally { try { svSocket.close(); } catch (IOException ioe2) {;} }
    }  // main
}  // TServer
```


java.net.Socket

```
int port = 8888;
String host = "para.chem.elte.hu";
Socket socket = new Socket(host,port);
socket = new Socket (InetAddress.getByName(host) , port);
InputStream in  = socket.getInputStream();
InputStream out = socket.getOutputStream();
int localPort = socket.getLocalPort();
int port = socket.getPort();
socket.close();
socket.shutdownInput();
socket.shutdownOutput();
socket.setSoTimeOut(1000);        //  VÁRAKOZÁS maximális ideje ms-ban
```


java.net.ServerSocket

```
ServerSocket svSocket = new ServerSocket (8888);
svSocket = new ServerSocket (
                888,InetAddress.getByName("157.181.192.2"));
Socket socket = svSocket.accept ();              // VÁRAKOZIK
svSocket.close ();
```

Internet és WWW objektumok

URI Universal resource identifier
URL Universal resource locator

- protocol://gépnév:port/path/file.ext#ref
- protocol://gépnév:port/path/file.ext?keres=mit&arg1=hol&arg2=miért+és+mitől...
- jar:http://localhost/pub/test.jar!/pelda/Kor.class

URI részei

- protocol: http ftp file gopher mail jar
- gépnév: IP address
- port: TCP port szám (1-65535)
- path: /dir1/dir2/.../filename.ext
- anchor: dokumentumon belüli referencia
  nehézfém
- keresés: név=Kovács+Józsi

Adott URL olvasása - 1

```
import java.io.*;
import java.net.*;
public class UrlTest {
  public static void main(String[] s) {
    String cim = "http://vegyszer.chem.elte.hu/altkem/index.html";
    try {
      URL url = new URL(cim);
      Object obj = url.getContent();                    // VÁRAKOZIK
      kiir (obj);
    }
    catch (MalformedURLException me) {;}
    catch (IOException ioe) {;}
    catch (ClassCastException cce) {;}
  } // main
```


Adott URL olvasása - 2

```
void kiir (Object obj) {
  if (obj instanceof InputStream)  {
    try {
      BufferedReader br = new BufferedReader(new InputStreamReader(
                          (InputStream)obj ) ) );
      boolean vege = false;
      while (vege != true ) {
        String line =  br.readLine();
        if (line==null) { vege = true; continue;}
        System.out.println(line);
      }
      br.close();
    }
    catch (IOException ioe) {;}
  }
} // kiir  Test
```

java.net.URL

```
URL url = new URL("para.chemelte.hu");
url = new URL ("http","para.chem.elte.hu",8080,"proba.txt");
String protocol = url.getProtocol();
String host     = url.getHost();
int port        = url.getPort();
String file     = url.getFile();
String ref      = url.getRef();
String query    = url.getQuery();
// tartalom olvasása
InputStream in = url.openStream();
Object obj     = url.getContent() throws java.io.IOException;
URLConnection urlConnection = url.openConnection();
```


java.net.URLConnection

```
URLConnection urlConnection = url.openConnection();
urlConnection   = new URLConnection(url);
String type     = urlConnection.getContentType();
int length      = urlConnection.getContentLength();
String encode   = urlConnection.getContentEncoding();
InputStream in  = urlConnection.getInputStream();
```


java.net.URLEncoder

```
String text = "Haho Öcsi";
String uText = URLEncoder.encode(text);
        ⇒ t = "Haho+Öcsi"
```

betűk, számok, -, _ ⇒ változatlanul
szóköz ⇒ +
~ ! @ # $ % ^ & * ⇒ *FF*%

```
String text = URLDecoder.decode(uText);
```

MIME = x-www-form-urlencoded

MIME kódok

Dokumentum fejléc mezői

- MIME-Version: 1.0
- Content-transfer-encoding: 7bit 8bit binary quoted-printable x-valami
- Content-type: *Tipus*/*Pontosítás*; paraméternév=paraméterérték
  - text/plain /richtext /html
  - image/jpeg /gif
  - audio/basic
  - video/mpeg
  - application/postscript /octet-stream
  - message/rfc822
  - multipart/mixed /parallel /alternative

Objektumok szerializációja

```
try {
    Socket s = new Socket(host,port);
    so = new ObjectOutputStream(s.getInputStream());
    so.writeInt(1);
    so.writeObject(szoveg);
    so.writeObject(adat);
    so.flush();
} catch (IOException ee) {;}

try {
    int szam = si.readInt();
    String szoveg = (String) si.readObject();
    double[] adat = (double[]) si.readObject();
}
catch (Exception e) {;}
```


Grafikai csomagok - 1

- **java.awt**
  - grafikus operációs rendszerek elemeire épül
  - alap window funkciók
    - grafikus elemek (Point, Rectangle, ...)
    - grafikus komponensek (Component : Label, TextField, Button, Choice, Checkbox, ...)
    - konténerek (Window, Frame, Panel, ScrollPane...)
    - elrendezés - LayoutManager-ek (GridLayout...)
    - Menu
    - Color, Cursor, Dialog, Font, Graphics, Graphics2D

Grafikai csomagok - 2

- **java.awt.event**
  - eseményfigyelők
    - ActionListener, KeyListener, MouseListener...
- **javax.swing**
  - grafikus komponensek kiterjesztése Java-ban írt grafikai megvalósítással "Swing"
  - (implementálják a javax.accessibility.Accessible interface-t)
  - J-vel kezdődik a nevük
  - Java 1.2-től standard könyvtárban
  - java.awt.Container kiterjesztései
  - *minden* awt komponensnek van megfelelője

Grafikai csomagok - 3

- java.awt.color - színterek között konvertál
- java.awt.datatransfer - clipboard
- java.awt.dnd - drag and drop
- java.awt.color - színterek között konvertál
- java.awt.font - low level fontkezelés
- java.awt.geom - 2D geometriai elemek transzformációi
- java.awt.image - pufferelt képkezelés
- java.awt.print - nyomtatás

- egyebek - felhasználót közvetlenül nem érintő csomagok

Grafikai csomagok - 4

- javax.swing.border - komponensek bekeretezése
- javax.swing.colorchooser - színválasztó komponensek
- javax.swing.event - Swing komponens eseménykezelők (táblázat, szövegkurzor, hiperlink)
- javax.swing.filechooser - file-t kiválasztó objektumok
- javax.swing.plaf - *pluggable look-and-feel*
- javax.swing.table - táblázat objektum
- javax.swing.text - text objektumok (komplett texteditor)
- javax.swing.text.html és rtf - speciális szövegobjektumok
- javax.swing.tree - fa-struktúra megjelenítése
- javax.swing.undo - *undo* eseménykezelés

GUI elemek kapcsolata

- awt: Component
  - Container (Frame)
  - Component (Button)
  - Layout
  - Event
  - EventListener
    - » Listener interface implementációja vagy
    - » Adapter osztály
- swing: Component -> Container
  - JComponent (JButton, ...)
  - .....

Grafikus program indítása

```
public class grafika {
    public static void main(String[] args) {
        GrafikaFrame frame = new  GrafikaFrame();
        frame.pack();
        frame.show();
    }
}
```


Grafikus Példa - 1

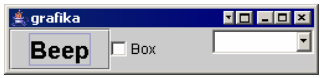
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class GrafikaFrame extends JFrame {
    public GrafikaFrame() {
        super("grafika");                          //JFrame konstruktor
        Container pane = this.getContentPane();
        GridLayout grid= new GridLayout(1,3);      // 1sor, 3 oszlop
        pane.setLayout(grid);
        JButton button = new JButton("Beep");
        button.setFont(new Font("SansSerif", Font.BOLD, 20));
        pane.add(button);
        pane.add(new Checkbox("box"));
        pane.add(new Choice());
```

Grafikus Példa - 2

```
WindowAdapter adapter = new WindowAdapter() {
   public void windowClosing(WindowEvent e) {
      GrafikaFrame.this.windowClosed();      //grafikaFrame példánya
   }
};
this.addWindowListener (adapter);

ActionListener listener = new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      GrafikaFrame.this.buttonPressed();
   }
};
button.addActionListener (listener);
} //konstruktor vége
```


Grafikus Példa - 3

```
// metódusok
protected void buttonPressed() {
   Toolkit.getDefaultToolkit().beep();
}
protected void windowClosed() {
   System.exit(0);
}
}
```



Komponensek és J-komponensek

Button	JButton
Checkbox és CheckboxGroup	JCheckbox
Choice	JChoice
List	JList
Label	JLabel
Scrollbar	JScrollbar
TextField	JTextField
TextArea	JTextArea
Canvas	Új: JSpinner (Java 1.4)
Container	---

Elrendezés Layout-okkal

- GridLayout
- FlowLayout
- BorderLayout
- CardLayout
- BridBagLayout
- *null* layout

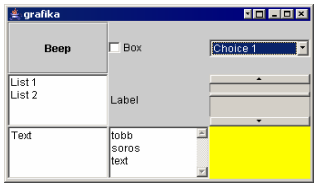
GridLayout - 1

Ekvidisztáns rácselrendezés egy JFame-ben

```
Container pane = this.getContentPane();
GridLayout gridLayout = new GridLayout(3,3);
pane.setLayout(gridLayout)
pane.add(new JButton("Beep"));
pane.add(new Checkbox("Box"));
Choice choice = new Choice();
choice.add("Choice 1");
choice.add("Choice 2");
pane.add(choice);
// folyt...
```

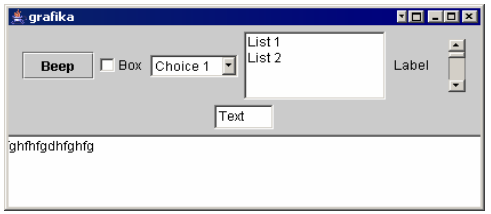

GridLayout - 2

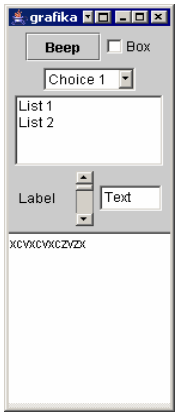
```
List list = new List();
list.add("List 1");
list.add("List 2");
pane.add(list);
pane.add(new Label("Label"));
pane.add(new Scrollbar());
pane.add(new TextField("Text"));
pane.add(new TextArea());
Canvas canvas=new Canvas();
canvas.setBackground(Color.yellow);
pane.add(canvas);
```



FlowLayout - 1

Folytonos elrendezés





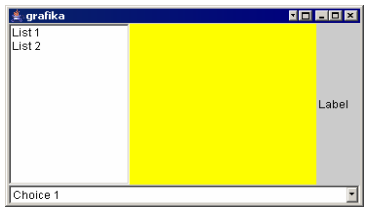
FlowLayout - 2

```
FlowLayout flow = new FlowLayout();
pane.setLayout(flow);
pane.add(new JButton("Beep"));
pane.add(new Checkbox("Box"));
Choice choice = new Choice();
choice.add("Choice 1");
choice.add("Choice 2");
pane.add(choice);
List list = new List(); list.add("List 1"); list.add("List 2");
pane.add(list);
pane.add(new Label("Label"),"East");
pane.add(new Scrollbar());
pane.add(new TextField("Text"));
pane.add(new TextArea());
```

BorderLayout - 1

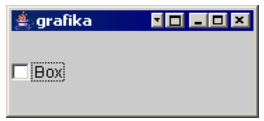
Peremhez igazítás

```
BorderLayout   border  =  new BorderLayout();
List list = new List(); list.add("List 1"); list.add("List 2");
pane.add(list,"West");
pane.add(new Label("Label"),"East");
Choice choice = new Choice();
choice.add("Choice 1");
choice.add("Choice 2");
pane.add(choice, "South");
Canvas canvas=new Canvas();
canvas.setBackground(Color.yellow);
pane.add(canvas,"Center");
```



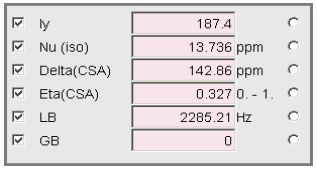
CardLayout

```
Container pane = this.getContentPane();
CardLayout card = new CardLayout();
pane.setLayout(card);
pane.add("button",new Button("Button"));
pane.add("box",new Checkbox("Box"));

card.first(pane);          // első lapot mutatja
card.next(pane);           // következőt mutatja
card.show(pane,"box"); // box nevűt mutatja
```



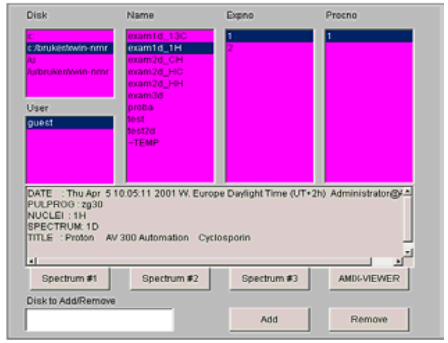
GridBagLayout és korlátok

```
GridBagLayout paramLayout=new GridBagLayout();
JPanel panel = new JPanel(paramLayout);
Component[] component = new Component[ 7 ];          //.... fel kell tölteni
GridBagConstraints gbc=new GridBagConstraints();   // Constraint = korlát
gbc.gridwidth = 1;
gbc.gridheight = 1;
gbc.anchor=GridBagConstraints.WEST;
gbc.fill=GridBagConstraints.HORIZONTAL;
for (j=1; j<=6; j++) {
      gbc.gridy = j;
      for (int i=0; i<7; i++) {
        gbc.gridx = i+1;
        paramLayout.setConstraints(component [ i ],gbc);
        panel.add(item);
      }
}
```



null Layout

```
Panel panel=new Panel();
panel.setLayout(null);
add("North",panel);
panel.resize(540,450);

Label diskT=new Label("Disk");
int c1=21;  int r1=0;
int w1=112; int h1=20;
diskT.setBounds(c1,r1,w1,h1);
panel.add(diskT);
```



Esemény-orientált programozás

- Adott *eseményt* egy objektum ír le.
  - az objektum neve *Event*-re végződik
- Az *XXXEvent*-et figyelő objektumnak az *XXXListener*-t kell implementálnia.
- Az *XXXListener*-t megvalósító objektumot egy grafikus objektumhoz regisztráltatunk:
  addXXXListener() illetve removeXXXListener()
- *XXXAdapter* osztály az *XXXListener*-t implementálja üres feldolgozó metódusokkal.

Eseményfigyelés

- Egy fajta esemény bekövetkeztekor a grafikus objektum a nála regisztrált eseményfigyelő egy bizonyos metódusát adott paraméterekkel meghívja.
- Egy grafikus objektumhoz több figyelőt is regisztrálhatunk - az értesítés sorrendje nem függ a regisztráció sorrendjétől.

Eseményfigyelő objektumok

1. Közös Listener

   Minden eseményforráshoz egyetlen - közös - Listener példányt rendelünk.
   A forrást getID(), getSource()-al tudjuk meg.

2. Egyedi Listenerek

   Minden eseményforráshoz egy saját Listener példányt rendelünk.

3. Névtelen egyedi Listener

   Az eseményforráshoz egy névtelen Listener osztályt példányosítunk.

Közös Listener - 1

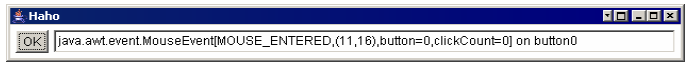
```
import java.io.*;
import java.awt.*;
import java.awt.event.*;

public class EventTest extends Frame implements Runnable, MouseListener {
    EventTest tester;
    TextField text;
    Button button;
    public static void main(String[] args) {
        EventTest test = new EventTest("Haho");
        test.run();
    }
    public EventTest (String s) {
        super(s);
    }
```

Közös Listener - 2

```
public void run () {
        setLayout (new  FlowLayout());
        button = new Button("OK");
        button.addMouseListener(this);
        text = new TextField(80);
        add(button);
        add(text);
        pack();
        show();
}
public void mouseClicked ( MouseEvent me )  {
        System.exit(0);
}
```


Közös Listener - 3

```
public void mouseEntered ( MouseEvent me ) {
        text.setText ( me.toString() );
}
public void mouseExited(MouseEvent me) {
}
public void mousePressed(MouseEvent me) {
}
public void mouseReleased(MouseEvent me) {
}
}
```



Leggyakoribb Event fajták

- ActionEvent — Button, MenuItem
- AdjustmentEvent — Scrollbar
- ComponentEvent — Component
  - FocusEvent
  - WindowEvent
  - InputEvent
    - KeyEvent
    - MouseEvent

Gyakori Event metódusok

- String s = actionEvent.**toString**();
- String t = actionEvent.**getActionCommand**();
  //(Button, MenuItem)
- int flag = actionEvent.**getModifiers**();
  //CTRL_MASK, ALT_MASK,..

- int i = adjustmentEvent.**getValue**(); //Scrollbar
- int type = adjustmentEvent.getAdjustmentType();
  //BLOCK_INCREMENT, TRACK, UNIT_INCREMENT

ComponentEvent metódusok

- Component c = componentEvent.**getComponent**();
- int id = focusEvent.**getID**(); // (FOCUS_GAINED, FOCUS_LOST)
- Window w = windowEvent.**getWindow**();
- long time = inputEvent.**getWhen**();

MouseEvent metódusok

- int x = mouseEvent.**getX**();
- int y = mouseEvent.**getY**();
- int m = mouseEvent.**getModifiers**();
  - ALT_MASK
  - ALT_GRAPH_MASK
  - BUTTON1_MASK ...
  - CTRL_MASK
  - SHIFT_MASK
  - META_MASK
- boolean sh = mouseEvent.**isShiftDown**(); ....
- long t = mouseEvent.**getWhen**();

KeyEvent metódusok

- char ch = keyEvent.**getKeyChar**(); // Unicode
- int code = keyEvent.**getKeyCode**();
- int id = keyEvent.getID();
  // KEY_PRESSED
  // KEY_RELEASED
- String t = KeyEvent.**getKeyText**(code); // Nyelvfüggő

Listenerek - interfészek

MouseListener

```
public abstract void mouseClicked ( MouseEvent me );
public abstract void mouseEntered ( MouseEvent me );
public abstract void mouseExited(MouseEvent me) ;
public abstract void mousePressed(MouseEvent me);
public abstract void mouseReleased(MouseEvent me);
```

Az implementáló osztálynak kell kitölteni a metódus törzsét.

```
public void mousePressed(MouseEvent me) {
    System.out.println(me.getX());
}
```

Adapterek - objektumok

MouseAdapter

```
public void mouseClicked ( MouseEvent me )  { }
public void mouseEntered ( MouseEvent me ) { }
public void mouseExited(MouseEvent me) { }
public void mousePressed(MouseEvent me) { }
public void mouseReleased(MouseEvent me) { }
```

Elég csak a szükségeset felüldefiniálni.

Egyedi figyelés Listener-rel - 1

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;

public class EventTest extends Frame implements Runnable  {
    EventTest tester;
    Button button;
    public static void main(String[] args) {
        EventTest test = new EventTest("Haho");
        test.run();
    }
    public EventTest (String s) {
        super(s);
    }
```


Egyedi figyelés Listener-rel - 2

```
    public void run () {
        setLayout (new  FlowLayout());
        // Komponensenként sorban
        MyListener myListener = new MyListener(this);
        button = new Button("OK");
        button.addMouseListener(myListener);
        add(button);
        //----------
        pack();
        show();
    }
}
```


Egyedi figyelés Listener-rel - 3

```
class MyListener extends Object implements MouseListener {
    Object object ;
    public MyListener (EventTest _object) {
        object = _object;
    }
    public void mouseClicked ( MouseEvent me )  {
        object = me.getSource();
    }
    public void mouseEntered   ( MouseEvent me ) {}
    public void mouseExited    ( MouseEvent me ) {}
    public void mousePressed   ( MouseEvent me ) {}
    public void mouseReleased( MouseEvent me ) {}
}
```

Egyedi figyelés Adapterrel

```
class MyListener extends MouseAdapter {
    Object object ;
    public MyListener (EventTest _object) {
        object = _object;
    }
    public void mouseClicked ( MouseEvent me )  {
        object = me.getSource();
    }
}
```


Névtelen Listener

```
public void run () {
        setLayout (new  FlowLayout());
        // Komponensenként sorban
        // kimarad: MyListener myListener = new MyListener(this);
        button = new Button("OK");
        button.addMouseListener(
           new MouseAdapter() {
              public void mouseClicked (MouseEvent me) {System.exit(0);}
           }
        );
        add(button);
        //----------
        pack();
        show();
    }
```



```
package grafika;    import java.awt.*;    import java.awt.event.*;    import javax.swing.*;

public class Grafika extends JFrame implements Runnable, ActionListener {
   public Grafika() { }
   public static void main(String[] args) {
      Grafika grafika = new Grafika();
      grafika.run();
   }
   public void run() {
      Container pane = this.getContentPane();
      pane.setLayout ( new FlowLayout ( ) );
      JButton button = new JButton("gomb");
      button.addActionListener ( this);
      pane.add(button);
      this.pack();   this.show();
   }
   public void actionPerformed (ActionEvent ae) {  System.exit(0);    }
}
```


Rajzolás - 1

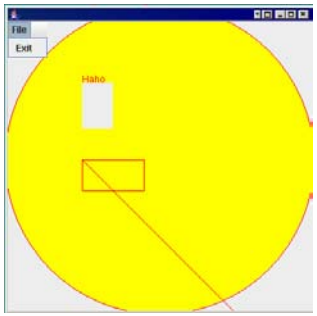
```
package firka;
import javax.swing.*; import java.awt.*; import java.awt.event.*;

public class Main extends JFrame implements ActionListener{
   public Main() {
     //menü
     JMenuBar mb=new JMenuBar();
     this.setJMenuBar(mb);
     JMenu file=new JMenu("File");
     JMenuItem exit=new JMenuItem("Exit");
     file.add(exit);
     mb.add(file);
     exit.setActionCommand("exit");
     exit.addActionListener(this);
     //repaint();
   }
```

Rajzolás - 2

```
public static void main(String[] args) {
    Main main=new Main();
    main.pack();
    main.show();
}
public void actionPerformed(ActionEvent ae) {
   String command=ae.getActionCommand();
   if (command.equals("exit")) {
      System.exit(0);
   }
}
public Dimension getPreferredSize() {
    return new Dimension(400,400);
}
```


Rajzolás - 3

```
public void paint(Graphics g) {
    Dimension size = getSize();
    // átmérő
    int d = Math.min(size.width, size.height);
    int x = (size.width - d)/2;
    int y = (size.height - d)/2;
    // rajzolás
    g.setColor(Color.yellow);
    g.fillOval(x, y, d, d);
    g.setColor(Color.red);
    g.drawOval(x, y, d, d);
    g.drawLine(100,200, 300,400);
    g.drawRect(100, 200, 80, 40);
    g.drawString("Haho",100,100);
    g.clearRect(100,100,40,60);
  }
}
```



Appletek és beágyazásuk HTML-be

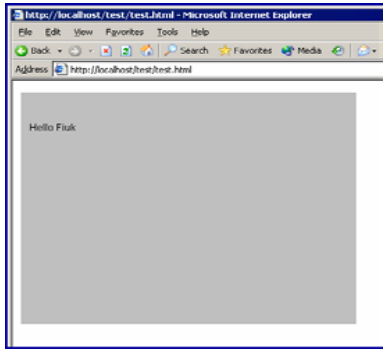
```
<html>
   <head></head>
   <body>
        < applet  codebase = “http://localhost:80/test”
                  archive    = “http://localhost:80/test/test.jar”
                  code       = “proba.class”
                  width       = 400   height = 300
                  name       = “ap1”     myscript
        >
        < param  name = “text”  value = “Hello Fiuk” >
        </applet>
   </body>
<html>
```


Applet

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

public class proba extends Applet {
    String text;
    public void init() {
       text = getParameter(“text”));
    }
    public void paint(Graphics g) {
       g.drawString(text,10,50);
    }
```

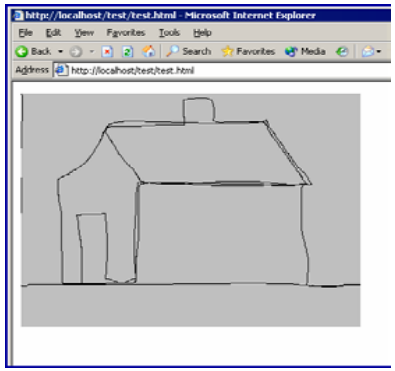
Applet - 2

```
    public void start() {
    }
    public void stop() {
    }
    public void destroy() {
    }
}
```



Firka - Java 1.0 eseményfigyelés

```
import java.applet.*;
import java.awt.*;
public class firka extends Applet {
   Graphics g;
   int regi_x = 0;
   int regi_y = 0;
   public void init() {
     g = this.getGraphics();
   }
   public boolean mouseDrag(
        Event e, int x, int y) {
     g.drawLine(regi_x, regi_y, x, y);
     regi_x = x; regi_y = y;
     return true;
   }
}
```



Soros port kezelése

Szükséges javacomm20-win32.zip

Benne

comm.jar	--> <jdk>\lib	
javax.comm.properties	--> <jdk>\lib	//config file
win32com.dll	--> <jdk>\bin	

set CLASSPATH = <jdk>\lib\comm.jar;%classpath%

Példa soros port kezelésre

```
import java.io.*;
import java.util.*;
import javax.comm.*;

public class SimpleRead implements Runnable, SerialPortEventListener {
   static CommPortIdentifier portId;
   static Enumeration portList;

   InputStream inputStream;
   SerialPort serialPort;
   Thread readThread;
   static String port="com1";
   static int baud=9600;
```

Soros port kezelője - 2

```
public static void main(String[] args) {
  portList = CommPortIdentifier.getPortIdentifiers();
  while (portList.hasMoreElements()) {
    portId = (CommPortIdentifier) portList.nextElement();
    if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
      if (portId.getName().equals(port)) {          // com1 megtalálása
        SimpleRead reader = new SimpleRead();       // példányosítás
      }
    }
  }
}
public void run() {
  try {   Thread.sleep(20000);                      // szál elaltatása 20 sec-ig
  } catch (InterruptedException e) {}
}
```


Soros port kezelője - 3

```
public SimpleRead() {                               // a példány konstruktora
  try {
    serialPort = (SerialPort) portId.open("SimpleReadApp", 5000);
  } catch (PortInUseException e) {}

  try {
    inputStream = serialPort.getInputStream();
  } catch (IOException e) {}

  try {
    serialPort.addEventListener(this);
  } catch (TooManyListenersException e) {}
  serialPort.notifyOnDataAvailable(true);
```


Soros port kezelője - 4

```
    try {
      serialPort.setSerialPortParams (baud,
      SerialPort.DATABITS_8,
      SerialPort.STOPBITS_1,
      SerialPort.PARITY_NONE);
    } catch (UnsupportedCommOperationException e) {}
    readThread = new Thread(this);                  // Runnable -> Thread
    readThread.start();
}
```


Soros port kezelője - 5

```
public void serialEvent (SerialPortEvent event) {
  switch(event.getEventType()) {            //.....
    case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
      break;
    case SerialPortEvent.DATA_AVAILABLE:
      byte[] readBuffer = new byte[20];
      try {
        while (inputStream.available() > 0) {
          int numBytes = inputStream.read(readBuffer);
        }
        System.out.println(new String(readBuffer));
      }
      catch (IOException e) {}
      break;
  }
 }
}
```

Távoli metódus-hívás (RMI)

- RMI - Remote Method Invocation
- Részei
  - Szerver *objektum*
  - Registry
  - Security Manager
  - Az *objektum* szerver- és kliens-csonkjai
  - Kliens

RMI szerver

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class Naplo extends UnicastRemoteObject implements NaploIfc {
  public static void main(String args[]) {
    //System.setSecurityManager(new RMISecurityManager());
    try {
      Naplo n = new Naplo();
      Naming.rebind("rmi://localhost:9999/Naplo",n);     // Bejegyzi a registry-be
      System.out.println("Naplozas fut ...");
    } catch (Exception e) {
      System.out.println("Naplozas nem indithato: "+e);
    }
  }
```


RMI szerver - 2

```
  public Naplo() throws java.rmi.RemoteException {
    super(); // Ez exportálja a távoli objektumot
  }

  // Szolgáltatás
  public void naploz(String szoveg) throws java.rmi.RemoteException {
    System.out.println(szoveg);
  }
}
//-------------------------------------------------------------------------------------------
public interface NaploIfc extends java.rmi.Remote {
  public void naploz(String szoveg) throws java.rmi.RemoteException;
}
```


RMI kliens

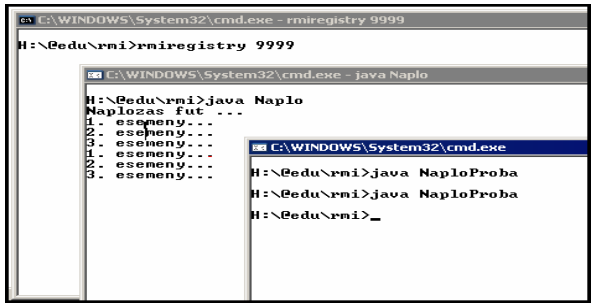
```
import java.rmi.*;

public class NaploProba {
    public static void main(String args[]) {
      try {
        NaploIfc ni=(NaploIfc) Naming.lookup("rmi://localhost:9999/Naplo");
            ni.naploz("1. esemeny...");
            ni.naploz("2. esemeny...");
            ni.naploz("3. esemeny...");
      } catch (Exception e) {
        System.out.println("Error "+e);
      }
    }
}
```

RMI fordítási lépések

- javac NaploIfc.java
  - **NaploIfc.class**
- javac Naplo.java
  - **Naplo.class**
- javac NaploProba.java
  - **NaploProba.class**
- rmic Naplo
  - **Naplo_skel.class**
  - **Naplo_stub.class**

RMI Registry és alkalmazás

Futtatás egyszerre három konzol-ablakban:



Tartalom

Tartalom

Tartalom

Tartalom

Tartalom

Tartalom

Tartalom

Vége