

# **Java programozási nyelv**

## **6. rész – Java a gyakorlatban**

**Nyugat-Magyarországi Egyetem**  
**Faipari Mérnöki Kar**  
**Informatikai Intézet**

**Soós Sándor**  
**2004. október**

# Tartalomjegyzék

- FreeJava fejlesztőkörnyezet
- A FreeJava különleges funkciói
- Kódolási konvenciók a Java-ban
- Csomagok, package
- Csomagdeklaráció
- Osztályok elérése
- Adatbeolvasás billentyűzetről
- Adatkonverzió
- Gyakorló feladat

# FreeJava fejlesztőkörnyezet

- Ingyenes, magyar nyelvű és magyar fejlesztésű Java fejlesztőkörnyezet
- A program weblapja: [www.belovai.ini.hu](http://www.belovai.ini.hu)
- A feltelepített Java SDK-t használja. Be kell állítani az SDK könyvtárát.
- Menü: **Beállítások / JDK beállítások...**
- Magyar nyelvű felülettel bemutatja a teljes Java osztálykönyvtárát.

# A FreeJava különleges funkciói

- Teljes egészében Javában íródott, így teljesen platformfüggetlen
- Szintaxis kiemelés
- Zárójelpárok kiemelése, blokkok jelzése a bal margón. Ki-bekapcsolható a **Beállítások / Beállítások... / Java** menüpontban
- Java osztály böngésző és osztály elemző
  - **Beállítások / JDK Beállítások....**: Java src.jar (zip)
  - **Keresés** menü

# Kódolási konvenciók a Java-ban

- Emlékeztetőül: a Java megkülönbözteti a kis és nagybetűket!
- Ami nem rajtunk múlik:
  - A nyelv kulcsszavai: csupa kisbetű (class, for, this)
  - standard típusok: csupa kisbetű (int, long, boolean)
  - A Java osztálykönyvtár azonosítói:
    - packagek, csomagok: csupa kisbetű (java.lang)
    - osztályok: nagy kezdőbetű
    - metódusok és tagváltozók: csupa kisbetű, több szó esetén egybeírva és a második szótól nagy kezdőbetűvel
    - konstansok (final tagváltozók): csupa nagybetű

# Kódolási konvenciók, folytatás

- Megállapodás:
  - Saját programjainkban követjük a fenti konvenciókat
  - Minden osztályt külön file-ba teszünk
  - A fájl neve megegyezik az osztály nevével beleértve a kis-, nagybetűs írásmódot is.
- Mostantól az órán is be kell tartani ezeket a konvenciókat.

# Csomagok, package

- Megállapodtunk abban, hogy minden osztályt önálló fájlba teszünk.
- Egy komolyabb program több tucat osztályból épül fel, ami több tucat file-t jelent.
- Az osztályok, illetve forrásfile-ok kategóriákba sorolását teszik lehetővé a csomagok.
- Az osztályok hierarchikus struktúrát alkotnak, ahogyan láttuk a **java.lang** esetében.

# Csomagdeklaráció

- A forrásfájl elején adhatjuk meg, hogy az adott fájl melyik csomaghoz tartozzon:
  - **package csomagnév;**
  - A csomagnév egy tetszőleges azonosító önmagában, vagy több azonosító ponttal elválasztva. Például:
    - package geometria;**
    - package geometria.sikidom;**
    - package geometria.test;**
  - Ha két fájlban azonos csomagnevet deklarálunk, akkor a két fájl azonos csomagba fog tartozni.
  - Ha egy fájlban nem adunk meg csomagdeklarációt, ahogyan eddig tettük, akkor ez a fájl egy közös, névtelen csomagba fog tartozni.

# Osztályok elérése 1

- Hogyan érünk el egy osztályt egy több csomagból álló programban?
  - Szinte minden program ilyen, mert a nyelvi elemek egy része is különböző csomagokban van definiálva.
    - Például **Double**, **Integer**, stb.
  - Egy saját osztályban definiált osztályt közvetlenül elérhetünk a nevével.
  - A **java.lang** csomagot automatikusan importálja a rendszer, azaz a benne lévő osztályokat ugyanúgy elérhetjük, mintha a saját osztályunkban lenne. Ez történt az eddigi példaprogramok nagy részében.

# Osztályok elérése 2

- Egy idegen csomagbeli osztály elérése.
  - Bármely osztályt elérhetünk a teljes nevével:
    - **csomag.osztálynév;**
  - Importdeklarációval közvetlenül elérhetjük az importált osztályokat:
    - **import csomag.osztálynév;**
    - **import csomag.\*;**
    - az utóbbi esetben a csomag minden publikus osztályát importáltuk
    - az **import java.lang.\*;** importdeklaráció minden fordítási egységben automatikusan megtörténik

# Adatbeolvasás billentyűzetről

- `javax.swing.JOptionPane` osztály
- Az osztály különböző metódusaival különböző típusú, modális párbeszéd ablakokat hozhatunk létre
- Modális párbeszéd ablak:
  - az ablak bezárásáig megáll az adott szál futása
- A számunkra fontos metódusok:
  - **`public static String showInputDialog(Object message)`**
    - Egy ablakban kiírja a **message** szöveget és egy szövegbeviteli mezőt. Visszaadja a mezőbe írt szöveget **String** típusként.
  - **`public static void showMessageDialog(Component parentComponent, Object message)`**
    - Egy ablakban kiírja a **message** szöveget. **parentComponent** lesz az ablak szülőkomponense, **null** esetén a képernyő.

# Adatbeolvasás példa

```
public static void main(String[] args) {  
    String s, t;  
  
    s = JOptionPane.showInputDialog("Írj be valamit!");  
  
    //lekezeljük azt is, ha a felhasználó nem írt be semmit  
    if (s!=null)  
        t="Ezt írtad be: " + s;  
    else  
        t="Nem írtál be semmit";  
  
    JOptionPane.showMessageDialog(null, t);  
}
```

# Adatkonverzió

- A **showInputDialog** metódus String típusban adja vissza a begépelte adatot. Így a metódus általánosabb, viszont át kell konvertálni a kapott numerikus adatokat.
- Ehhez felhasználhatjuk a standard típusokat beágyazó osztályokat, amelyek az absztrakt **java.lang.Number** leszármazottai, például:
  - int → java.lang.Integer
  - double → java.lang.Double
  - ...

# Adatkonverzió, folytatás

1. lépés: **String** → **Number** leszámazott
    - konstruktor által, vagy a
    - **valueOf()** static metódus által
    - ebben a lépésben történik meg a tényleges konverzió
  2. lépés: **Number** → standard típus (**int**, **double**, ...)
    - **xyValue()** metódus által
      - xy helyén a standard típus áll: **intValue()**, **doubleValue()**
- A fenti két lépést egyetlen kifejezéssel is megvalósíthatjuk.

# Adatkonverzió példa

```
String s1 = "3.14", s2 = "2.71";  
double d1, d2;  
Double D1;
```

```
// konverzió lépésekben
```

```
D1 = new Double( s1 ); // vagy D1 = Double.valueOf(s1);  
d1 = D1.doubleValue();
```

```
System.out.println( "s1: " + s1 + " = " + d1 );
```

```
// konverzió egy lépésben
```

```
d2 = new Double(s2).doubleValue();  
// vagy d2 = Double.valueOf(s2).doubleValue();
```

```
System.out.println( "s2: " + s2 + " = " + d2 );
```

# Gyakorló feladat

- Alakítsuk át az előző, síkidomos feladatot a következőképpen:
  - A **Sikidom**, **Teglalap** és **Kor** osztályok legyenek egy **geometria.sikidom** nevű csomagban!
  - A további definiált osztályok legyenek szintén ebben a csomagban!
  - A főprogram legyen egy **geometria.foprogram** nevű csomagban!
  - A főprogramot alakítsuk át úgy, hogy a szükséges adatokat interaktívan olvassuk be!