Java II.
A Java programozási nyelv alapelemei

Ficsor Lajos

Miskolci Egyetem

Általános Informatikai Tanszék

Utolsó módosítás: 2008. 02. 19.

A Java formalizmusa

- A C, illetve az annak bővítéseként definiált C++ szintaktikájával nagyon rokon.
- A továbbiakban az alapelemeket a C nyelvvel összehasonlítva ismertetjük.
- A először csak a Java nyelv legegyszerűbb elemeivel foglalkozunk.
- A leírás a legtöbb esetben nem teljes, de a C nyelv ismeretében egyszerű programok írásához elegendő.
- A hiányzó információkat a későbbiekben pótoljuk.

A Java program

- A Java program osztályok halmaza. Végrehajtható kód csak osztály metódusában fordulhat elő.
- Egy alkalmazás belépési pontja egy olyan osztály, amelyben van egy
  `public static`
  minősítésű **`main`** nevű metódus.
- A fenti fogalmak (osztály, metódus, minősítők) pontos jelentésére a későbbiekben kitérünk.

Azonosítók

- A Java a Unicode karakterkészletet használja, tehát akár ékezetes azonosítókat is használhatunk.
- Az **azonosító** betűvel kezdődő és betűvel vagy számmal folytatódó karaktersorozat.
  - Az _ és a $ is a betűk közé sorolandó.
  - A betűk bármelyik karakterkészletből származhatnak.
  - Az azonosító hossza tetszőleges.
  - A JAVA is kis- és nagybetű érzékeny (*case sensitive*).
  - Kulcsszavak, valamint a **`const, goto, null, true, false`** szavak nem lehetnek azonosítók.

Megjegyzések

- Mint a C-ben:

```
/* Szöveg */
```

- Egysoros:

```
// Innentől a sor végéig komment
```

- Dokumentációs

```
/**  Tetszőleges szöveg*/
```

A javadoc segédprogram feldolgozza.

Egyszerű típusok

- Hasonlóak, mint C-ben, de pontosan definiált helyfoglalással és ábrázolási móddal
- Nincs **`unsigned`** minősítő
- Konstans változó **`final`** minősítővel deklarálható (a **`const`** helyett).

Egyszerű típusok táblázata

Típusnév	Jelentés
`boolean`	Logikai típus (**`true`** vagy **`false`** értékkel)
`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 (IEEE 754 szerint)
`double`	64 bites lebegőpontos szám (IEEE 754 szerint)

Literálok (konstansok)

- Logikai értékek: **`true`** vagy **`false`**
- A C-vel teljesen egyező módon használandók:
  - Egész konstans
  - Lebegőpontos konstans
  - Karakter konstans (az escape szekvenciák is!)
  - Szöveg konstans (csak Unicode szöveget is tartalmazhat)
- A szöveg konstanst a fordító automatikusan **`String`** típusúként kezeli (magyarázat majd később).

Lokális változó használata 1.

- **Definíciója csak metóduson belül**.
- Nincs globális változó!
- Formája mint a C-ben, beleértve a kezdőértékadást is.
- Egyszerű típusú lokális változó definíciója egyben helyfoglalást is jelent.

Lokális változó használata 2.

Különbségek:

- Változó definíció egy metóduson belül bárhol lehet (nem csak blokk elején).
- Természetesen a változó csak a definíció után használható.
- **Figyelem:** a későbbiekben osztály változókról (adattagokról) is lesz szó!

Egydimenzós tömb

Definíciója eltér a C szintaktikától.

Két lépés (amely egy utasításba összevonható):

1. Tömb típus deklarációja

   ```
   típusnév azonosító[];
   ```

   vagy

   ```
   típusnév [] azonosító;
   ```

2. Helyfoglalás a tömb elemeinek:

   ```
   azonosító = new típusnév[elemek száma]
   ```

Egydimenzós tömb (folyt.)

- A két lépés egy utasításba összevonva:

```
típusnév azonosító [] = new
  típusnév[elemek száma]
```

- Az elemekre való hivatkozás már ugyanaz, mint a C nyelvben.
- A Java-ban többdimenziós tömbök is definiálhatók, amelyek nem feltétlenül „négyzetesek”.
- Bár nincs előre definiált osztálya, de osztályhoz tartozóként kezeli a Java.

Egydimenzós tömb (folyt.)

- Minden tömbhöz tartozik egy **`length`** konstans, amely az elemek számát adja meg.
- Használata:
  `azonosító.lenght`

Alaptípusú tömb

Példa:

```
int jegyek[]; // Ez csak egy tömb
              // típus létrehozása
jegyek = new int[100]; // Helyfoglalás
                       // 100 elemnek
jegyek[16] = 1; // Elem értékének
                // beállítása
```

Az indexelés itt is 0-tól kezdődik.

Különbség: az érvénytelen index kivételt vált ki!

Megjegyzés:

A fentieket a későbbiekben még pontosítjuk!

Operátorok

- A C majdnem minden operátorát ismeri a Java (kivéve a pointerhez kapcsolódókat)
- Az operátorok jelentése az egyszerű típusokra ugyanaz.

Különbségek:

- négy új operátor (**`>>>`**, **`>>>=`**, **`instanceof`**, **`new`**)
- nincs **`,`** (vessző) operátor
- A kifejezések kiértékelési sorrendje meghatározott.

Kifejezések kiértékelési sorrendje

A kifejezések kiértékelési sorrendjét meghatározza:

1. Zárójelezés

2. Operandusok prioritása

3. Azonos prioritás esetén balról-jobbra szabály, kivétel az értékadás, amely jobbról-balra értékelődik ki.

A kifejezésekben a metódushívások ("függvény hívások") sorrendje is a kiértékelés sorrendjét követi.

Utasítás, blokk

Utasítás lehet:

- kifejezés utasítás
- deklarációs utasítás.

Az utasításokat pontosvessző zárja.

Kifejezés utasítás csak a következő lehet:

- értékadás,
- **++** és **--** operátorokkal képzett kifejezések,
- metódushívás,
- példányosítás.

Utasítás, blokk (folyt.)

- A deklarációs és kifejezés utasítások tetszőleges sorrendben követhetik egymást.
- Az utasítások sorozata **{ }** jelek közé zárva a blokk.
- Utasítás helyére mindig írható blokk.

Vezérlő utasítások

Lényegében megegyeznek a C utasításaival.

Különbségek:

- Az **`if, while, do`** utasításokban a feltétel csak logikai kifejezés lehet.
- A **`for`** utasításban a második kifejezés csak logikai kifejezés lehet
- A **`switch`** utasításban a szelektor csak egész kifejezés lehet.
- Nincs **`goto`** utasítás.