II. A Java nyelv eszközei

1. Milyen eszközöket nyújt a Java a programozóknak

Korábban már említettük, hogy a Java a C nyelvből alakult ki, ezért a C, C++ nyelvben járatos programozóknak nem fog nehézséget okozni a nyelv szintaktikájának elsajátítása. Kis túlzással fogalmazhatnánk úgy is, hogy aki ismeri a C nyelv nyújtotta programozási eszközöket, az már ismeri a Java nyelv eszközeit is. Ennek ellenére ebben a fejezetben áttekintjük, átismételjük a fontosabb dolgokat, már csak azért is, mert a mondás szerint : „Az ismétlés a tudás anyja”.

2. Típusok

A Java erősen típusos nyelv. Ez azt jelenti, hogy előre el kell döntenünk, és deklarálnunk kell, hogy a változóinkban milyen adatokat szeretnénk tárolni a jövőben.
A nyelv a változókat két nagy csoportba sorolja:

1. ***Egyszerű (primitív) típusok***
   - int (egész)
   - float (valós)
   - long (egész)
   - char (karakter)
   - boolean (a c-vel ellentétben itt valódi logikai típusról van szó, melynek értékei: true,false)
2. ***Referencia típusok***
   - tömb
   - String (osztály típus)

A Java-ban a típuskonverzió a C-hez hasonló módon működik. Megkülönböztetünk automatikus és explicit, kikényszerített típuskonverziót. A szűkebb típusról szélesebb típusra konvertálás minden probléma nélkül végbemegy, azonban fordított esetben a kerekítés miatt információvesztés léphet fel. Meg kell jegyeznünk, hogy az automatikus típuskonverzió bizonyos esetekben nem működik és a fordító hibát jelez számunkra. Ilyenkor érdemes explicit módon megadni a típust. Összességében azt mondhatjuk, hogy a típuskonverzió korlátozott esetekben automatikusan végbemegy.

3. Változók

Változóinkat a használatuk előtt a C-ben megszokott módon deklarálni kell.

Újdonság a korábbi nyelvekhez képest, hogy a Java a változónevekhez az UNICODE karakterkészletet biztosítja, így lehetőségünk van magyar ékezetes betűk használatára a változónevekben. Megjegyzendő azonban, hogy osztálynevekben kerülendő az ASCII készleten kívüli karakterek használata, ugyanis az osztály nevének pontosan meg kell egyezni a fájl nevével. Ez pedig kerülendő, mert a különböző platformok eltérő módon támogatják vagy éppen nem támogatják az ékezetes karaktereket.

4. Kifejezések, műveletek

Minden kifejezést pontosvessző zár le!

Az alapműveletek pontosan úgy használhatók, mint a C nyelvben:

- +,-,/,*,%
- inkrementálás prefix (++a) és postfix (a++) alakban
- dekrementálás prefix (--a) és postfix (a--) alakban
- összehasonlító operátorok: = =, >, <, >=, <=, != (ezek csak egyszerű adattípusoknál használhatók)
- logikai operátorok: && (és) , || (vagy)

5. Szelekciók

```
if(kif){
   //törzs
}
else {
  //törzs
}
```

illetve

```
switch(kif) {
  case konst1 : utasítás; break;
  case konst2: utasítás; break;
  …
  default : utasítás;
}
```

6. Iterációk

Elöltesztelő ciklus

```
while (kif){
 // törzs
}
```

Hátultesztelő ciklus

```
do{
  //törzs
}while(kif);
```

Növekményes ciklus:

```
for(kezdő ; teszt ; növelés){
// törzs
}
```

7. Metódusok

A C-vel ellentétben a Java-ban nincsenek függvények.

Az első órán úgy tanultuk, hogy a Java program osztálydefiníciókból épül fel, és az osztályon belül lehetőségünk van metódusok definiálására.

Metódusok általános alakja:

visszatérési_érték ***metódusnév*** (*paraméterlista*)

```
{
  //törzs
}
```

A void típusú függvények nem adnak vissza értéket.

Fontos megjegyezni, hogy egy blokkban definiált változó csak a blokkon belül él!

8. Tömbök

A Java nyelvben a tömbök objektumok, a C nyelvtől eltérően igazi típusok.

Egy tömböt kétféleképpen lehet deklarálni:

- <típus> <név> []; (pl.. int tomb[] ;)
- <típus> [] <név>; (pl.: int [] tomb;)

A tömb létrehozásához a **new** operátort kell használnunk a következőképpen:

<név>= **new** <típus> [<méret>];

pl.: tomb = **new** int[20];

A tömbök használatakor már a C nyelvben megszokott módszert kell használni:
Pl.: adat = tomb[i]; ***Az indexelés 0-tól történik!!!***

9. Többdimenziós tömbök

A Java-ban nincsenek többdimenziós tömbök, de létre lehet hozni őket.
Ha egy tömb elemeinek tömböket definiálunk, akkor többdimenziós tömböt fogunk kapni.

Deklarálás:
<típus> [] [] <tömbnév>;
pl.: int [] [] matrix;

Definiálás:
new <típus> [méret1] [méret2];
pl.: int [] [] matrix = new int [3] [6];

Használat:
<név> [n] [m];
pl.: elem = [2] [1];

A tömbök méretének meghatározása azok length metódusa szolgál.
Pl.: hossz = vektor.length();

10. Stringek

A Java nyelvben a stringek nem karaktertömbök, mint a C nyelvben. Itt igazi típusról van szó, de nem primitív típusról, hanem osztály típusról. Ezért kell megjegyezni, hogy osztály révén a használatnál figyelni kell, hogy nagy kezdőbetűvel írjuk. Említettük, hogy az ismert összehasonlító operátorok csak primitív típusok esetén használhatók. Olyan típusok összehasonlítására, mint amilyen a String is, az equals() segítségére van szükségünk.

```
Pl.:  IF (nev.equals(„ember”)) {
        //törzs
      }
```

11. Beolvasás a billentyűzetről

Biztosan mindenki emlékszik a C nyelv scanf() függvényére, amivel igazán kényelmesen tudtunk adatokat fogadni a klaviatúráról. A Java nyelvben ez már nehezebb feladat, és ez nem is képezi e tantárgy anyagát, ezért egy előre elkészített osztályt biztosítunk a hallgatóságnak, amivel kényelmesen tudnak adatokat beolvasni. Ez az osztály az **Input** osztály lesz (Akinek van kedve, tanulmányozza az *Input.java* állományt).

Az osztály használatához először le kell fordítanunk az Input.java állományt, így létrejön az Input.class bájtkódú állomány. Ennek a bájtkódnak elérhetőnek kell lennie a fordítónak (tipikusan abban a könyvtárban kell lennie, ahol éppen dolgozunk).

A beolvasást az Input osztály következő metódusaival lehet elvégezni:

- **Input.**readln() – String beolvasásához
- **Input**.readInt() – Integer beolvasásához
- **Input**.readFloat() – lebegőpontos valós szám beolvasásához
- **Input**.readLong() – hosszú egész beolvasásához
- **Input**.readChar() – karakter beolvasásához

Példa:

```
String vezeteknev;
System.out.println(„Kérem adja meg a vezetéknevét:”);
vezeteknev = Input.readln();
System.out.println(„A vezetéknév: „+vezeteknev);
```