Budapesti Műszaki és Gazdaságtudományi Egyetem

Java technológia

Tömbök

Mik a tömbök?

- A tömbök a Javában **azonos**, vagy **egy ősből származó típusú** elemek **fix méretű**, **sorszámozott** halmazai.
- Egy tömbnek két fontos tulajdonsága van, az **elemtípus**a (element type), és az **elemszám**a (length).
- A tömbök **elemtípusukra és elemszámukra nézve immutábilisak**.
- Az elemtípus lehet **bármely primitív**, vagy **referencia típus**, beleértve a tömböket is, így **többdimenziós tömbök** is létrehozhatók.
- A tömbök **objektumként viselkednek**, melynek típusa, bár az **Objectből származik**, nem osztály, hanem egy speciális **tömb típus**, mely magában foglalja az elemtípust is.

Tömbök létrehozása

- Tömbök létrehozása a new kulcsszóval történik:

```
int[] aTomb;
aTomb = new int [30];
```

- A fenti példa egy 30 elemű tömböt hoz létre, az elemek 0-tól számozottak, és az elemtípus alapértékére inicializáltak.
- Tömböt az értékei alapján is létrehozhatunk, ilyenkor a méretét nem adhatjuk meg, hiszen az következik az elemek számából:

```
int[] bTomb = new int[] { 1,2,3,4,5,6,7 };
```

- Referencia típusok esetén hasonló a helyzet:

```
String[] szövegek = new String[] {
      "Kis kutya", "nagy kutya", "nem ugat hiába." }
```

Több dimenziós tömbök I.

- A több dimenziós tömbök lényegében tömbök tömbjei:

```
int[][] multiDim = new int [10][10];
int[] egySor = multiDim [0];                    // az első sor
int egyÉrték = egySor [0];                     // az első érték
```

- Több dimenziós tömböt úgy is létrehozhatunk, hogy a magasabb dimenziók mentén nem adjuk meg a méretét:

```
int[][] multiDim = new int [10][];
```

- Ekkor csak jelezzük, hogy az egyes "sorok" tömbök, de értékük null lesz, a "sorokat" később hozhatjuk létre:

```
multiDim [0] = new int [20];
multiDim [1] = new int [30];
```

Több dimenziós tömbök II.

- Nyilvánvalóan nem lehetséges viszont, hogy a magasabb dimenziók méretét megadjuk, az alacsonyabbakét nem:

```
int[][] multiDim = new int [][10];                    // fordítási hiba
```

- Az alábbi metódus egy tetszőleges méretű alsó háromszögmátrixot épít fel úgy, hogy a sorok hossza növekszik, az elemek értéke pedig a sor és oszlop koordináták összege.

```
int[][] mátrixKészítő (int sorok) {
   int[][] m = new int [sorok][];
   for (int s = 0;s < sorok;s++) {
      m [s] = new int [s + 1];
      for (int o = 0; o <= s; o++) {
         m [s][o] = s + o;
      }
   }
   return m;
}
```

Típuskonverziók I.

- A tömbtípusok egymás közti konverziója lehet implicit, illetve explicit konverzió, az elemtípusnak megfelelően:

```
class A {}
class B extends A {}
class C {}

A[] a1,a2 = new A [10],a3 = new B [10];
B[] b1,b2 = new B [10];
C[] c;

a1 = b2;                                          // implicit konverzió
b1 = (B[]) a3;                                    // explicit konverzió
b1 = (B[]) a2;              // explicit konverzió, ClassCastException
c = a2;                                            // fordítási hiba
```

Típuskonverziók II.

- Primitív elemtípusú tömbök nem konvertálhatók:

```
int[] ia = new int [10];
double[] da = new double [10];

da = ia;                                          // fordítási hiba
ia = (int[]) da;                                  // fordítási hiba
```

- A típuskonverziók veszélyeket is hordoznak magukban:

```
class A {}
class B extends A {}

B[] b = new B [10];                                          // OK
A[] a = b;                                                   // OK
a [0] = new A ();           // OK, de futásidőben ArrayStoreException
```

- **Tanulság:** a tömbök esetében is figyelni kell a fordításidejű és a futásidejű típusok közötti különbségekre!

Tömbök tulajdonságai

- A tömbök elemszáma a tömbtípusok length mezőjén keresztül kérdezhető le, de természetesen nem módosítható:

```
int[] a = new int [42];
int l = a.length                                                    // 42
a.length = 50;                        // fordítási hiba, a length final mező
```

- A 0 méretű tömbök effektíve immutábilisak.

```
int[] üres = new int [0];
```

Tömbök másolása

- Tömböket a java.lang.System osztály arraycopy metódusával másolhatunk hatékonyan:

```
int[] egyik = new int [100];
int[] másik = new int [200];

System.arraycopy (egyik,                                    // forrás
      0,                                          // kezdőindex a forrásban
      másik,                                                  // cél
      0,                                        // kezdőindex a célban
      100);                                                 // elemszám
```

- A tömböknek létezniük kell (a forrás és a cél sem lehet null), a másolandó tartomány nem "lóghat ki" a forrás, illetve a cél tömbből, az elemeknek pedig egyszerű értékadással (assignment conversion) kovertálhatóknak kell lenniük a céltömb elemtípusára.

Tömbök méretének változtatása

- A tömbök elemszáma nem módosítható, a feladat másolással oldható meg:

```
int[] átméretez (int[] t,int méret) {
   int[] t2 = new int [méret];
   System.arraycopy (t,0,t2,0,méret < t.length ? méret : t.length));
}
```

- Az alábbi példa egy dinamikusan táguló tömböt valósít meg:

```
int[] hozzáad (int[] t,int index,int elem) {
   if (t.length <= index) {
      int l = t.length;
      while (l <= index)
         l *= 2;
      int[] t2 = new int [l];
      System.arraycopy (t,0,t2,0,index);
      t = t2;
   }
   t [index] = elem;
   return t;
}
```

java.util.Arrays

- Tömbök hatékony kezelésére további (statikus) metódusok találhatóak a java.util.Arrays osztályban
  - sort() – tömb rendezése
  - fill() – tömb kitöltése egy adott elemmel
  - equals() és deepEquals() – tömbök tartalmának összehasonlítása (utóbbi Object típusú tömbökre)
  - toString() és deepToString() – tömbök tartalmának szöveges reprezentációja (utóbbi Object típusú tömbökre)
  - copyOf() – tömbről másolat készítése (gyakorlatban egyszerűbb mint a System.arraycopy() metódus)
- **Ezeknek a metódusoknak a többsége compiler intrinsic-ként van megvalósítva HotspotVM felett, vagyis a metódusok JVM bytekód helyett közvetlenül, specializált gépi kódra fordulnak – vagyis nagyon hatékonyak**