

Programozási nyelvek – Java

Típusok



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

1 Változók tárolása

2 Hatókör és élettartam

- Inicializáció
- Szemétgyűjtés
- Statikus tagok

3 Tömbök

- Többdimenziós eset

4 Felsorolási típus

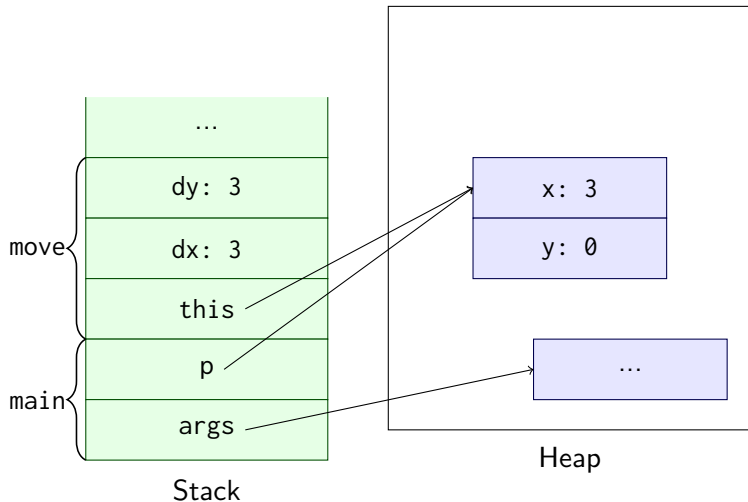
Referencia

- Osztály típusú változó
- Objektumra hivatkozik
- Heap
- Létrehozás: `new`
- Dereferálás: `.`

```
Point p;  
p = new Point();  
p.x = 3;
```



Különböző típusú változók a memóriában



Típusok

Primitív típusok

- byte: $[-128..127]$
- short: $[-2^{15}..2^{15} - 1]$
- int: $[-2^{31}..2^{31} - 1]$
- long: 8 bájt
- float: 4 bájt
- double: 8 bájt
- char: 2 bájt
- boolean: {false,true}

Referenciák

- Osztályok
- Tömb típusok
- ...



Ábrázolás a memóriában

Végrehajtási verem

Lokális változók és paraméterek
(Primitív típusú, referencia)

Heap

Objektumok, mezők
(Primitív típusú, referencia)



1 Változók tárolása

2 Hatókör és élettartam

- Inicializáció
- Szemétgyűjtés
- Statikus tagok

3 Tömbök

- Többdimenziós eset

4 Felsorolási típus

Lokális változók hatóköre és élettartama

- Más nyelvekhez (pl. C) hasonló szabályok
- Lokális változó élettartama: hatókör végéig
- Hatókör: deklarációtól a közvetlenül tartalmazó blokk végéig
- Elfedés: csak mezőt

```
class Point {  
    int x = 0, y = 0;  
    void foo( int x ){           // OK  
        int y = 3;             // OK  
        {  
            int z = y;  
            int y = x;          // Fordítási hiba  
            ...  
        }  
    }  
}
```



Objektumok élettartama

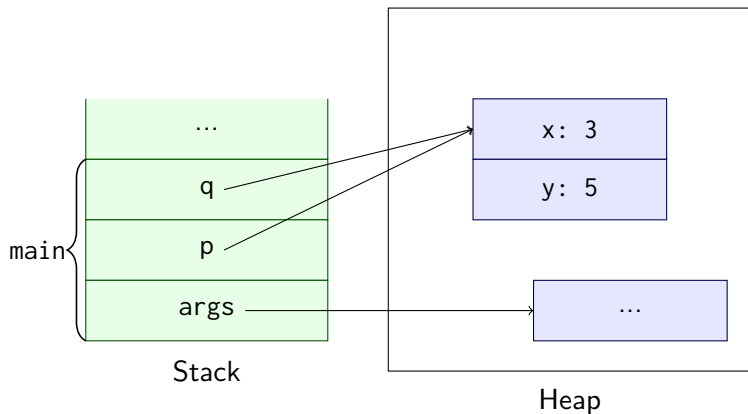
- Létrehozás + inicializálás
- Referenciák ráállítása
 - Aliasing
- Szemétgyűjtés

```
new Point(3,5)
Point p = new Point(3,5);
Point q = p;
p = q = null;
```



Aliasing

```
Point p = new Point(3,5), q = p;  
q.x = 6;
```



Üres referencia

```
Point p = null;  
p = new Point(4,6);  
if( p != null ){  
    p = null;  
}  
p.x = 3;    // NullPointerException
```



Üres referencia

```
Point p = null;  
p = new Point(4,6);  
if( p != null ){  
    p = null;  
}  
p.x = 3;    // NullPointerException
```



Mezők inicializálása

Automatikusan, nulla-szerű értékre

```
class Point {  
    int x = 0, y = 0;  
}
```

```
class Point {  
    int x, y;  
}
```

```
class Point {  
    int x, y = 0;  
}
```

```
class Point {  
    int x, y = x;  
}
```



Inicializálás üres referenciára

```
class Hero {  
    String name;           // == null  
    Hero bestFriend;       // == null  
}
```

```
Hero ironMan = new Hero();  
ironMan.name = "Iron Man";  
// ironMan.bestFriend == null
```



Lokális változók inicializálása

- Nincs automatikus inicializáció
- Explicit értékadás kell olvasás előtt
- Fordítási hiba (statikus szemantikai hiba)

```
public static void main( String[] args ){  
    int i;  
    Point p;  
    p.x = i;    // duplán fordítási hiba  
}
```

Lokális változóra garantáltan legyen értékadás, mielőtt az értékét használni próbálnánk!



Garantáltan értéket kapni

- „Minden” végrehajtási úton kapjon értéket
- Túlbiztosított szabály (ellenőrizhetőség)

Példa a JLS-ből (16. fejezet, Definite Assignment)

```
{  
    int k;  
    int n = 5;  
    if (n > 2)  
        k = 3;  
    System.out.println(k); /* k is not "definitely assigned"  
                           before this statement */  
}
```



Szemétgyűjtés

Feleslegessé vált objektumok felszabadítása

Helyes

Csak olyat szabadít fel, amit már nem lehet elérni a programból

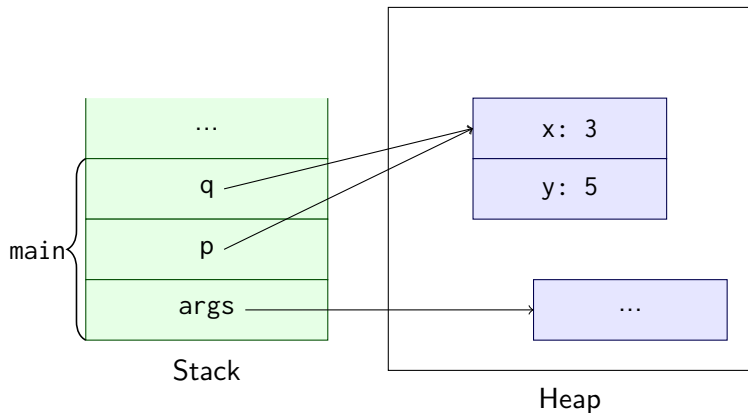
Teljes

Mindent felszabadít, amit nem lehet már elérni



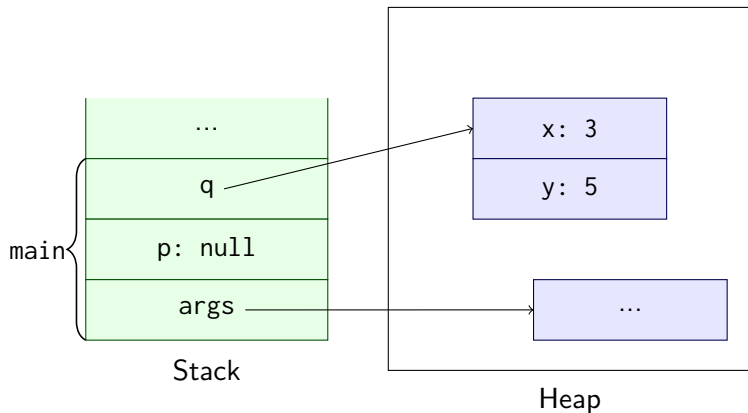
Még nem szabadítható fel

```
Point p = new Point(3,5), q = p;
```



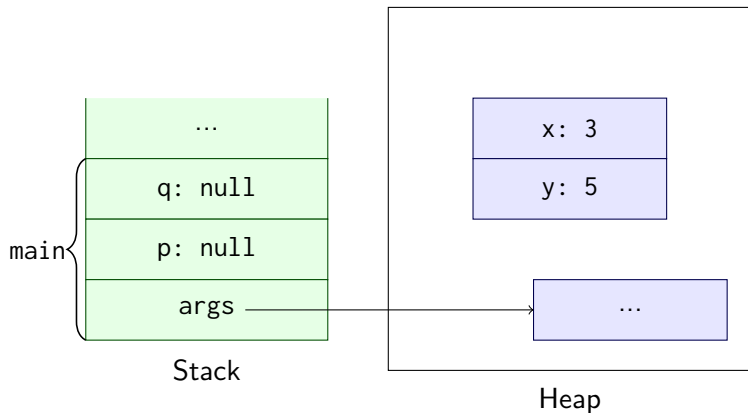
Még mindig nem szabadítható fel

```
p = null;
```



Már felszabadítható

```
q = null;
```



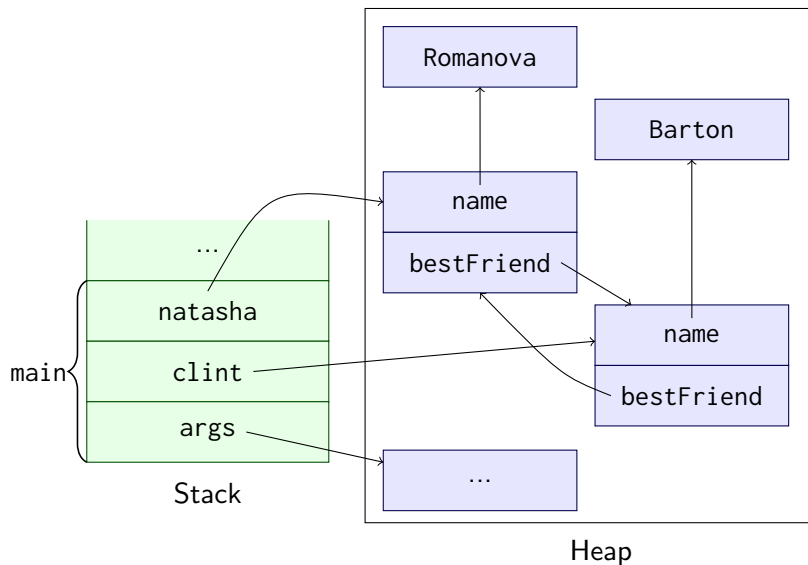
Bonyolultabb példa

```
class Hero {  
    String name;  
    Hero bestFriend;  
}
```

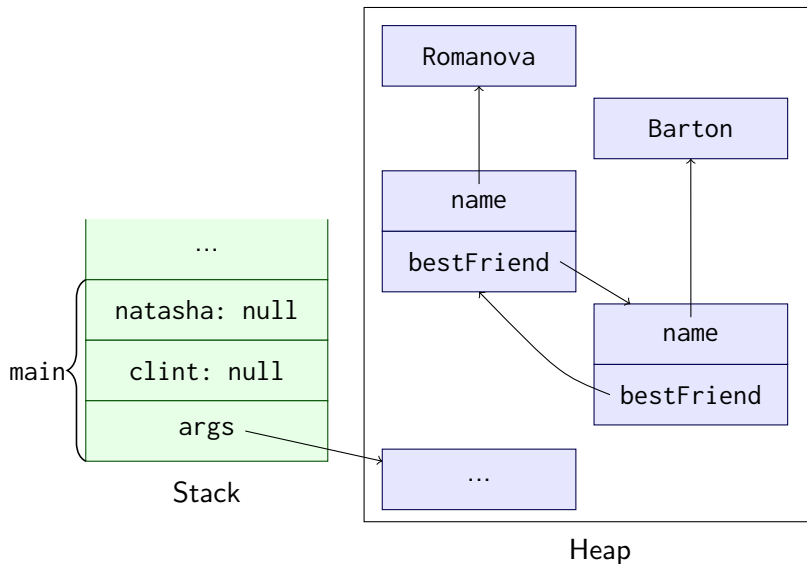
```
Hero clint = new Hero(),  
    natasha = new Hero();  
clint.name = "Barton";  
natasha.name = "Romanova";  
clint.bestFriend = natasha;  
natasha.bestFriend = clint;
```



Hősök a memóriában



```
natasha = clint = null;
```

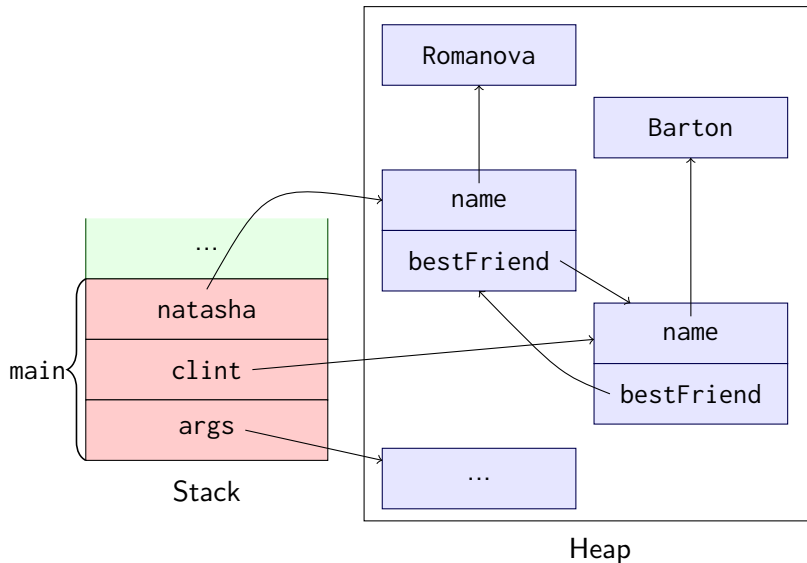


Mark-and-Sweep szemétgyűjtés

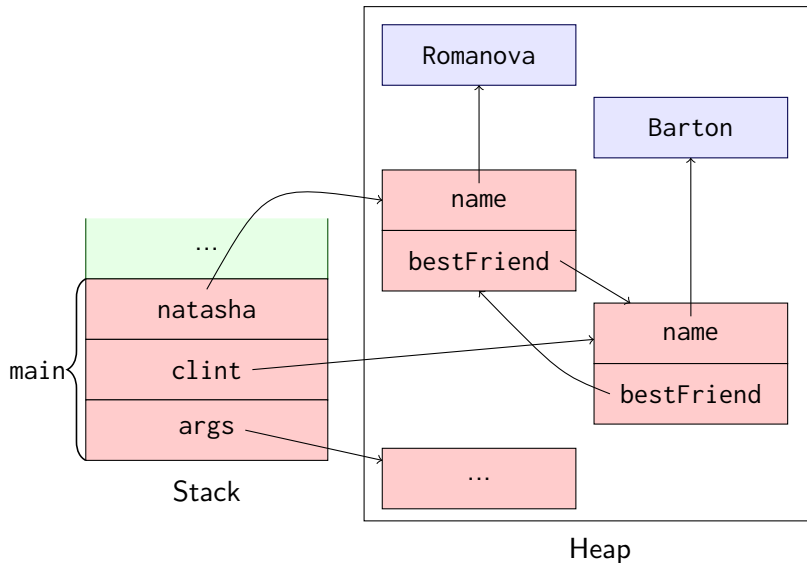
- Mark fázis
 - Kiindulunk a vermen lévő referenciákból
 - Megjelöljük a belőlük elérhető objektumokat
 - Megjelöljük a megjelöltekből elérhető objektumokat
 - ... amíg tudunk újabbat megjelölni (tranzitív lezárt)
- Sweep fázis
 - A jelöletlen objektumok felszabadíthatók



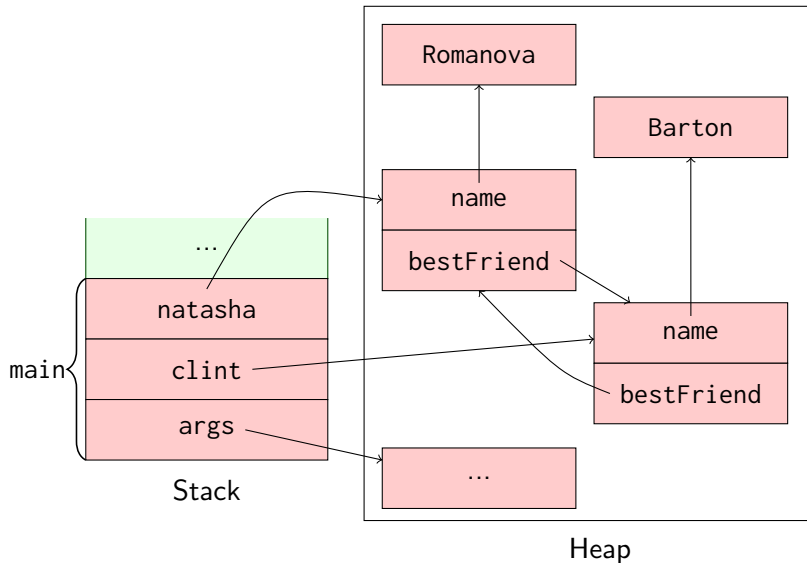
Mark-and-sweep: root set



Mark-and-sweep: propagálás

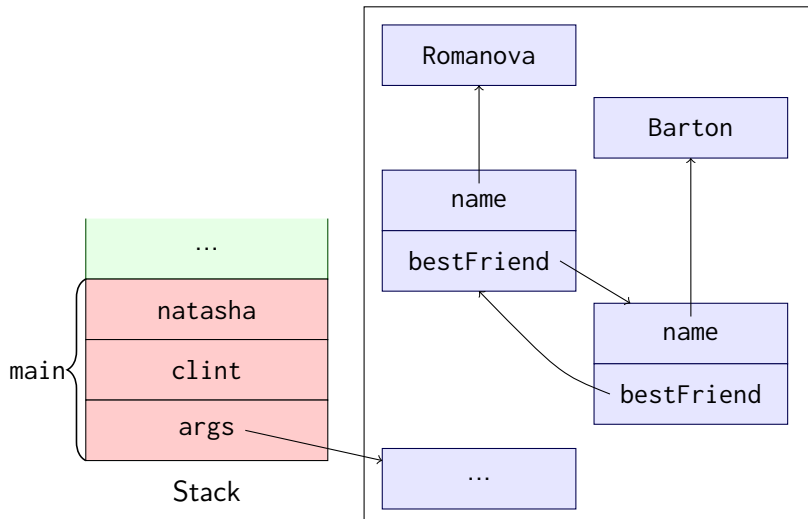


Mark-and-sweep: itt most mindegyik objektum elérhető



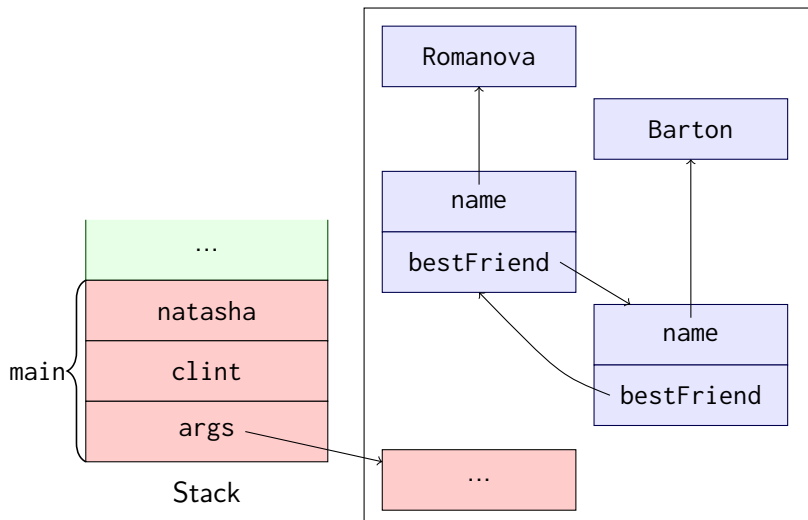
Mark-and-sweep: `natasha = clint = null`; ismét

`natasha = clint = null`;



Mark-and-sweep: mark fázis vége

- A sweep fázis felszabadítja az elérhetetlen objektumokat



Statikus mezők

- Hasonló a C globális változóihoz
- Csak egy létezik belőle
- Az osztályon keresztül érhető el
- Mintha *statikus tárhelyen* lenne, nem az objektumokban

```
class Item {  
    static int counter = 0;  
}  
  
class Main {  
    public static void main( String[] args ){  
        System.out.println( Item.counter );  
    }  
}
```

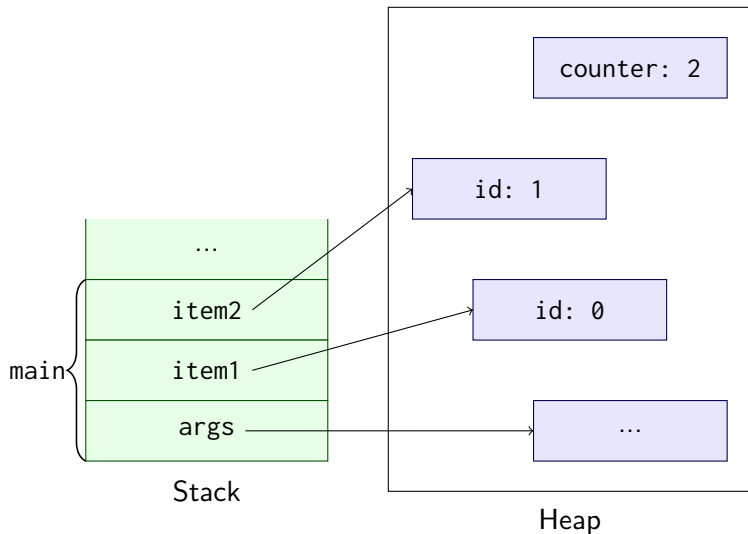


Osztálysztintű és példányszintű mezők

```
class Item {  
    static int counter = 0;  
    int id = counter++;      // jelentése: id = Item.counter++  
}  
  
class Main {  
    public static void main( String[] args ){  
        Item item1 = new Item(), item2 = new Item();  
        System.out.println( item1.id );  
        System.out.println( item2.id );  
        System.out.println( item1.counter ); // csúf, jelentése:  
                                              // Item.counter  
    }  
}
```



```
Item item1 = new Item(), item2 = new Item();
```



Statikus metódusok

- Hasonló a C globális függvényeihez
- Az osztályon keresztül hívható meg, objektum nélkül is lehet
- Nem kap implicit paramétert (this)
- A statikus mezők logikai párja

```
class Item {  
    static int counter = 0;  
    static void print(){  
        System.out.println( counter );  
    }  
}  
  
class Main {  
    public static void main( String[] args ){  
        Item.print();  
    }  
}
```



Statikus metódusban nincsen this

```
class Item {  
    static int counter = 0;  
    int id = counter++;  
    static void print(){  
        System.out.println( counter );  
        System.out.println( id );        // értelmetlen  
    }  
}  
  
class Main {  
    public static void main( String[] args ){  
        Item.print();  
    }  
}
```



- 1 Változók tárolása
- 2 Hatókör és élettartam
 - Inicializáció
 - Szemétgyűjtés
 - Statikus tagok
- 3 Tömbök
 - Többdimenziós eset
- 4 Felsorolási típus

Tömb

- Adatszerkezet
- Tömbelemek egymás után a memóriában
- Indexelés: hatékony
- Javában is 0-tól indexelünk, []-lel



Tömb típusok

`String[] args`

- Az args egy referencia
- A tömbök objektumok
 - A heapen tárolódnak
 - Létrehozás: `new`
- A tömbök tárolják a saját méretüket
 - `args.length`
 - Futás közbeni ellenőrzés
 - `ArrayIndexOutOfBoundsException`



Tömbök bejárása

```
public static void main( String[] args ){  
    for( int i = 0; i < args.length; ++i ){  
        System.out.println( args[i] );  
    }  
}
```



ArrayIndexOutOfBoundsException

```
public static void main( String[] args ){  
    for( int i = 0; i <= args.length; ++i ){  
        System.out.println( args[i] );  
    }  
}
```



Iteráló ciklus (enhanced for-loop)

```
public static void main( String[] args ){  
    for( int i = 0; i < args.length; ++i ){  
        System.out.println( args[i] );  
    }  
}
```

```
public static void main( String[] args ){  
    for( String s: args ){  
        System.out.println( s );  
    }  
}
```



Tömbök létrehozása, feltöltése, rendezése

```
class Sort {  
    public static void main( String[] args ){  
  
        int[] numbers = new int[args.length]; // 0-kkal feltöltve  
  
        for( int i = 0; i < args.length; ++i ){  
            numbers[i] = Integer.parseInt( args[i] );  
        }  
  
        java.util.Arrays.sort(numbers);  
  
        for( int n: numbers ){ System.out.println(n); }  
    }  
}
```



Statikus tagok importja

```
import static java.util.Arrays.sort;
class Sort {
    public static void main( String[] args ){

        int[] numbers = new int[args.length]; // 0-kkal feltöltve

        for( int i = 0; i < args.length; ++i ){
            numbers[i] = Integer.parseInt( args[i] );
        }

        sort(numbers);

        for( int n: numbers ){ System.out.println(i); }
    }
}
```



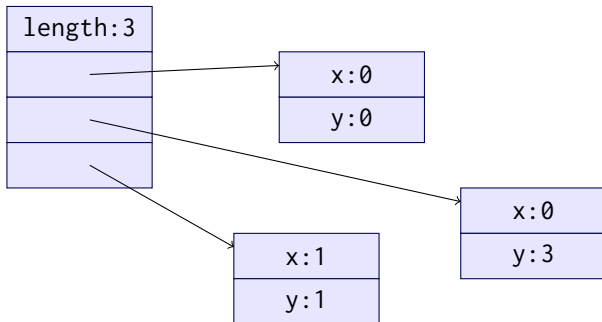
Referenciák tömbje

```
Point[] triangle = { new Point(0,0),  
                      new Point(0,3),  
                      new Point(1,1) };
```



Referenciák tömbje

```
Point[] triangle = { new Point(0,0),  
                     new Point(0,3),  
                     new Point(1,1) };
```



Heap



Lépésről lépésre

```
static void séta(){  
    Láb[] százlábú;  
    System.out.println( százlábú.length );  
}
```



Lépésről lépésre

```
static void séta(){  
    Láb[] százlábú;  
    System.out.println( százlábú.length );  
  
    százlábú = null;  
    System.out.println( százlábú.length );  
}
```



Lépésről lépésre

```
static void séta(){  
    Láb[] százlábú;  
    System.out.println( százlábú.length );  
  
    százlábú = null;  
    System.out.println( százlábú.length );  
  
    százlábú = new Láb[100];  
    System.out.println( százlábú.length );  
}
```



Lépésről lépésre

```
static void séta(){  
    Láb[] százlábú;  
    System.out.println( százlábú.length );  
  
    százlábú = null;  
    System.out.println( százlábú.length );  
  
    százlábú = new Láb[100];  
    System.out.println( százlábú.length );  
  
    for( int i = 0; i<100; i+=2 ){  
        százlábú[i]    = new Láb("bal");  
        százlábú[i+1] = new Láb("jobb");  
    }  
}
```



Mátrix

```
double[][] id3 = { {1,0,0}, {0,1,0}, {0,0,1} };
```



Mátrix

```
double[][] id3 = { {1,0,0}, {0,1,0}, {0,0,1} };
```

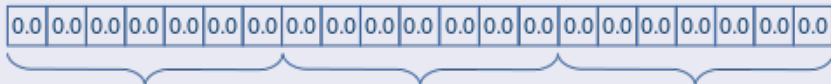
```
static double[][] id( int n ){  
    double[][] matrix = new double[n][n];  
    for( int i=0; i<n; ++i ){  
        matrix[i][i] = 1;  
    }  
    return matrix;  
}
```



C versus Java

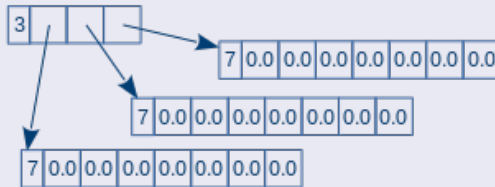
Többdimenziós tömb C-ben

```
double matrix[3][7];  
for(int i=0; i<3; ++i) for(int j=0; j<7; ++j) matrix[i][j] = 0.0;
```



Tömbök tömbje Javában

```
double[][] matrix = new double[3][7];
```



Indexelés

Háromdimenziós tömb C-ben

```
T t[L][M][N];
```

$$\text{addr}(t_{i,j,k}) = \text{addr}(t) + ((i \cdot M + j) \cdot N + k) \cdot \text{sizeof}(T)$$

Tömbök tömbjének tömbje Javában

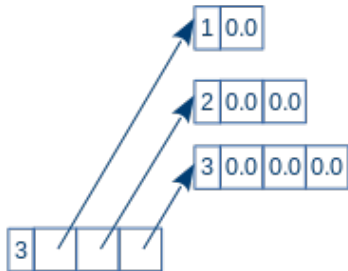
```
T[][][] t = new T[L][M][N];
```

$$\text{addr}(t_{i,j,k}) = \text{val}_8\left(\text{val}_8(\text{addr}(t) + 4 + i \cdot 8) + 4 + j \cdot 8\right) + 4 + k \cdot \text{sizeof}(T)$$



Alsóháromszög-mátrix

```
static double[][] zeroLowerTriangular( int n ){  
    double[][] result = new double[n][];  
    for( int i = 0; i<n; ++i ){  
        result[i] = new double[i+1];  
    }  
    return result;  
}
```



Parancssori argumentumok

- C-ben: `char *argv[]`
- Java megfelelője: `char[][] argv`
- Javában: `String[] args`



- 1 Változók tárolása
- 2 Hatókör és élettartam
 - Inicializáció
 - Szemétgyűjtés
 - Statikus tagok
- 3 Tömbök
 - Többdimenziós eset
- 4 Felsorolási típus

Referencia típusok Javában

- Osztályok (class)
- Interfészek (interface)
- Felsorolási típusok (enum)
- Annotáció típusok (@interface)



Felsorolási típus

```
enum Day { SUN, MON, TUE, WED, THU, FRI, SAT }
```

- Referencia típus
- Értékek: objektumok, nem intek



Felsorolási típus

```
enum Day { SUN, MON, TUE, WED, THU, FRI, SAT }
```

- Referencia típus
- Értékek: objektumok, nem intek

```
Day best = Day.SAT;           // import static?  
best = 3;                     // fordítási hiba  
int n = best;                 // fordítási hiba  
int m = best.ordinal();      // 6
```



Fix, zárt típusértékhalmoz

Csak a felsorolt típusértékek!

- **nem hívható meg a konstruktor, pl.:** `new Day()`



Fix, zárt típusérték-halmaz

Csak a felsorolt típusértékek!

- **nem hívható meg a konstruktor, pl.:** `new Day()`
- reflection segítségével sem példányosítható
- nem örökölhettünk belőle
- klónozással sem jön létre új objektum
- objektumszerializációval sem hozható létre új objektum



Konstruktorok, tagok

```
enum Coin {  
    PENNY(1), NICKEL(5), DIME(10), QUARTER(25);  
  
    private final int centValue;  
  
    Coin(int centValue) { this.centValue = centValue; }  
  
    public int centValue() { return centValue; }  
  
    public int percentageOf( Coin that ) {  
        return 100 * centValue / that.centValue();  
    }  
} // Forrás: Java Community Process (módosítva)
```



switch-utasításban

```
static int workingHours( Day day ){  
    switch( day ){  
        case SUN:  
        case SAT: return 0;  
        case FRI: return 6;  
        default:  return 8;  
    }  
}
```

