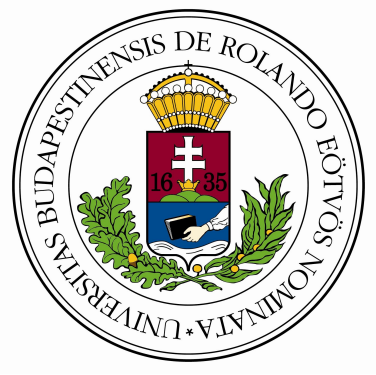
Programozási nyelvek – Java

Típusok



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

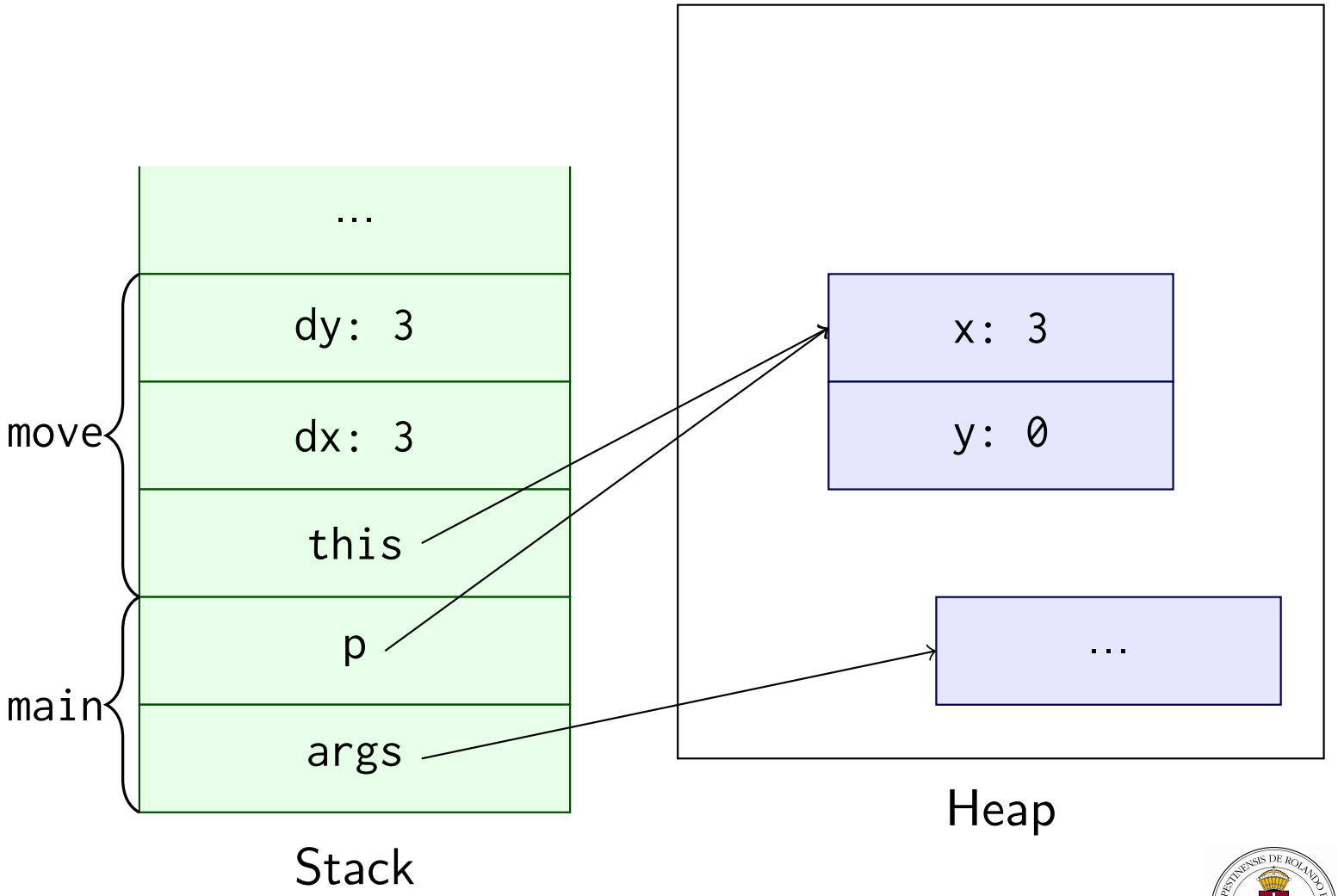
Outline

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, mezőik
(Primitív típusú, referencia)

Outline

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
            ...
        }
    }
}
```

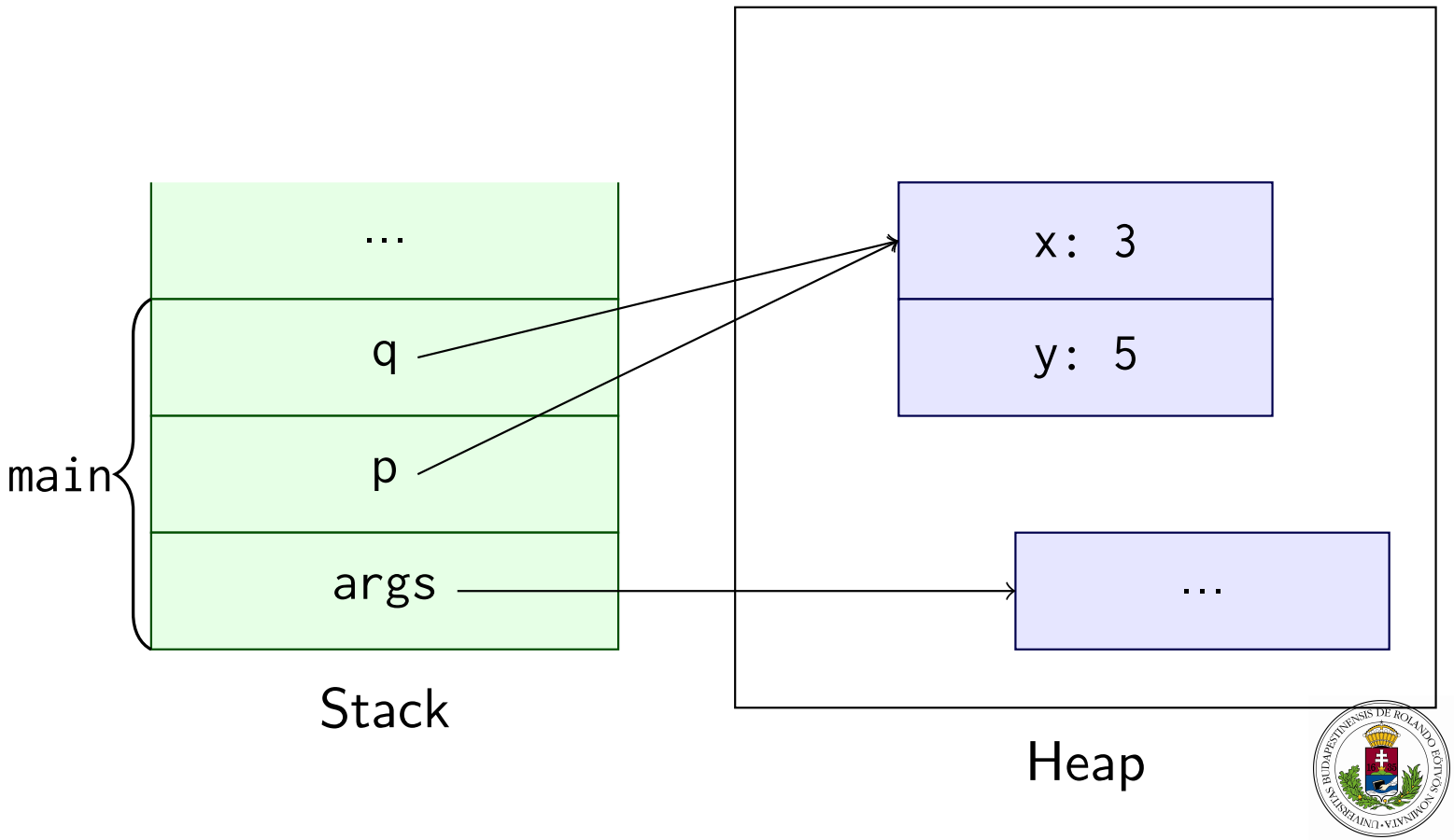
Objektumok élettartama

- 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
```

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

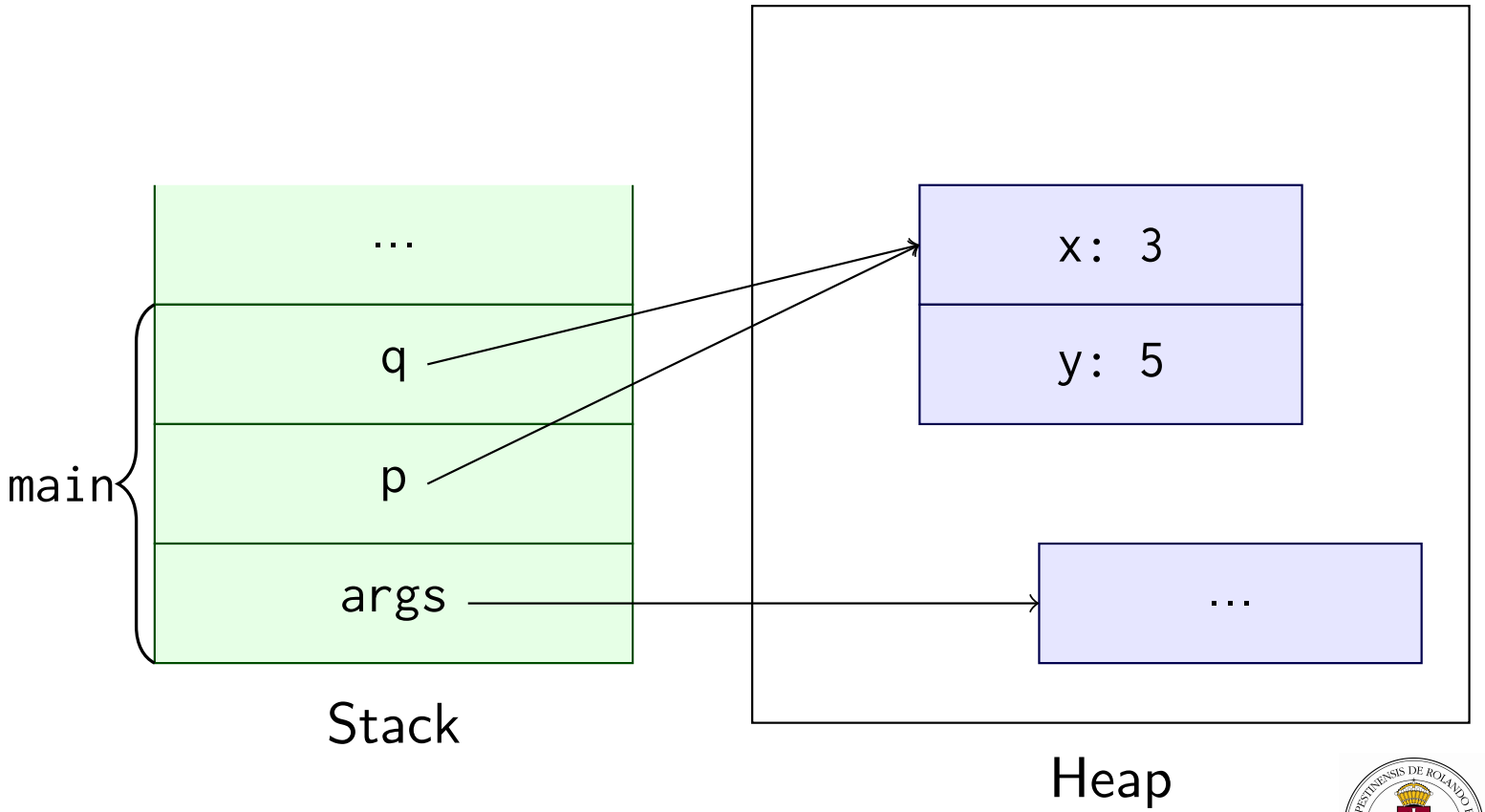
Csak olyat szabadít fel, amit már nem lehet elérni a programból

Teljes

Mindent felszabadít, amit nem lehet már elérni

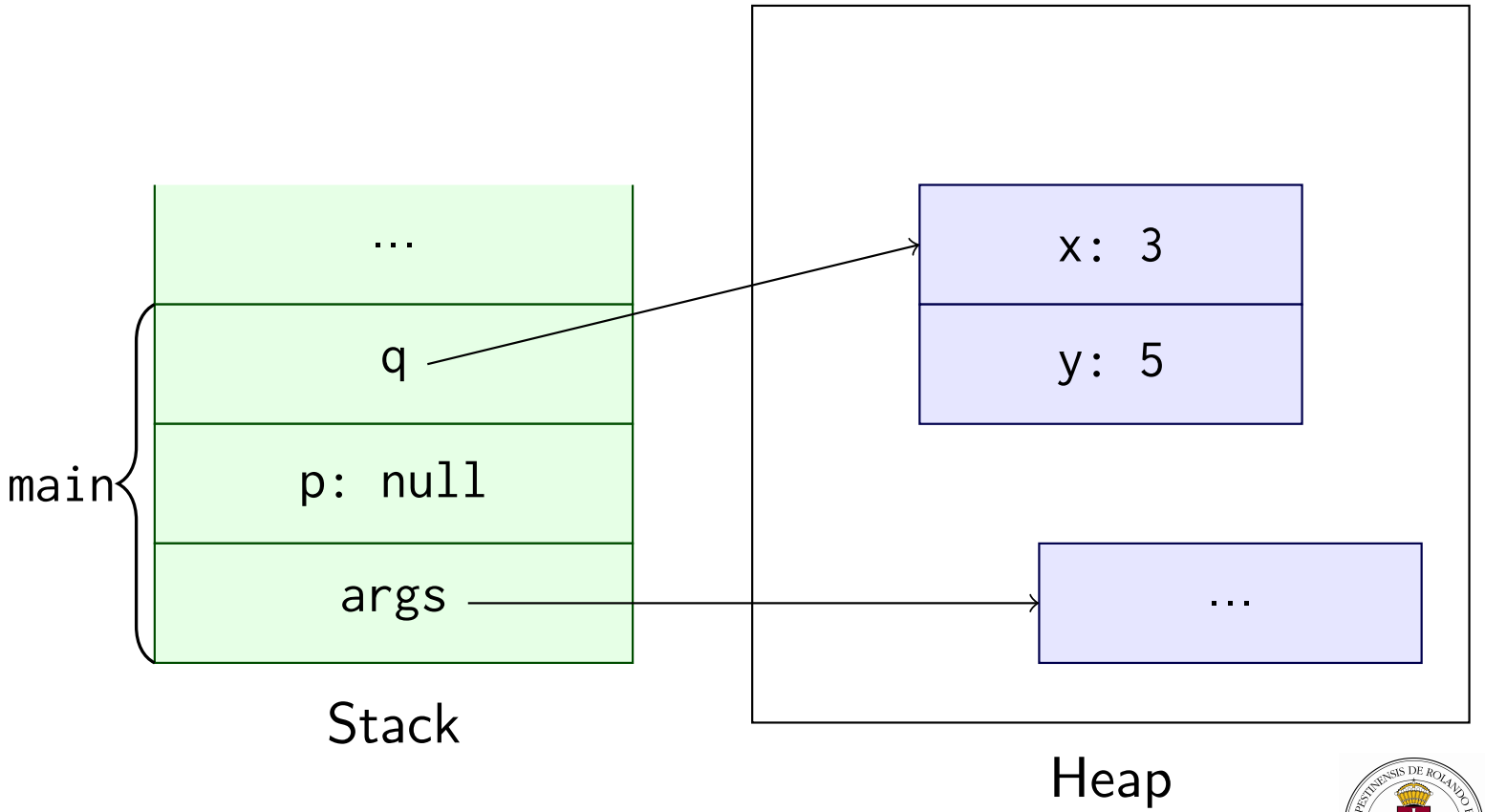
Még nem szabadítható fel

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



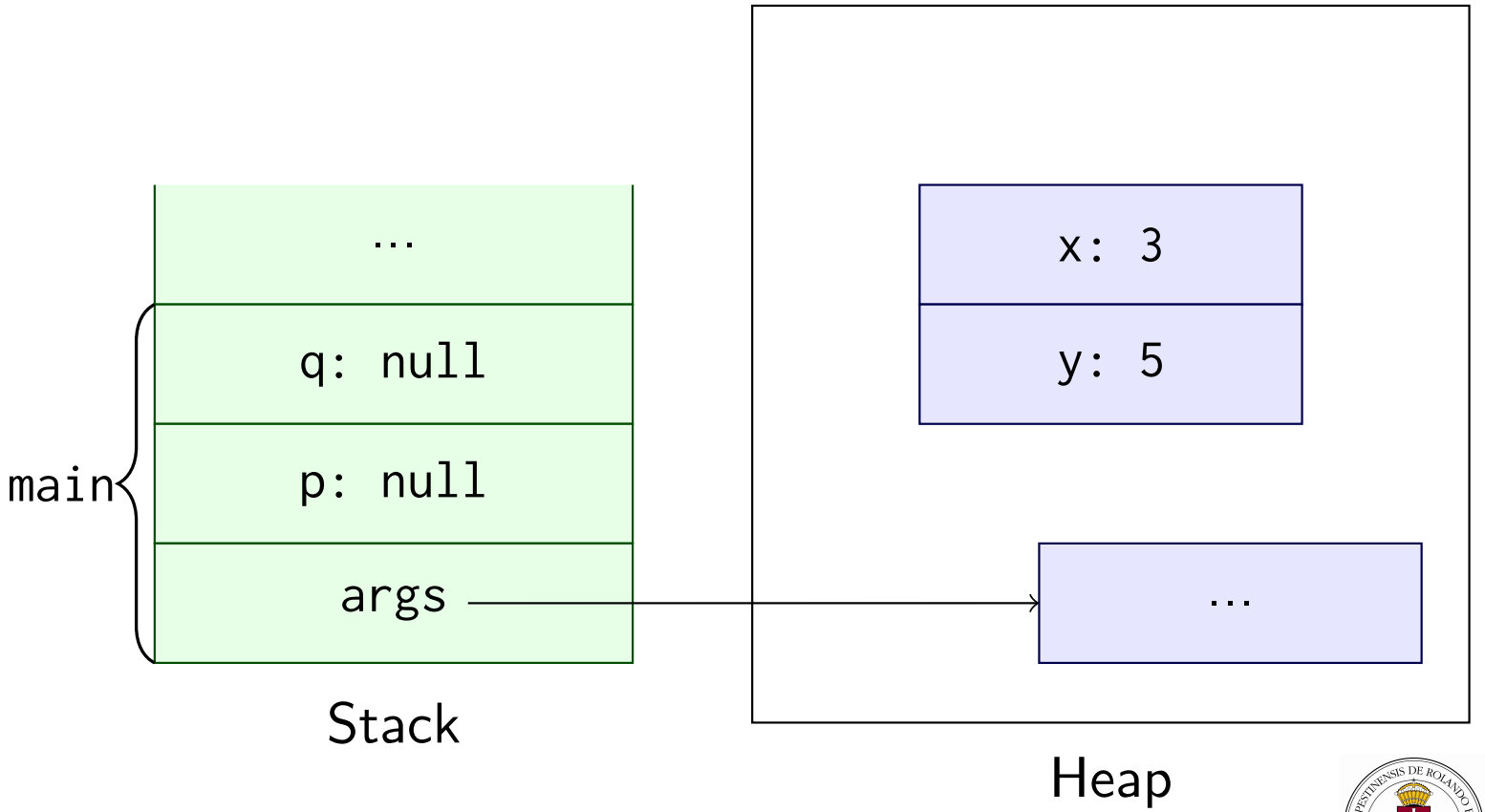
Még mindig nem szabadítható fel

```
p = null;
```



Már felszabadítható

```
q = null;
```

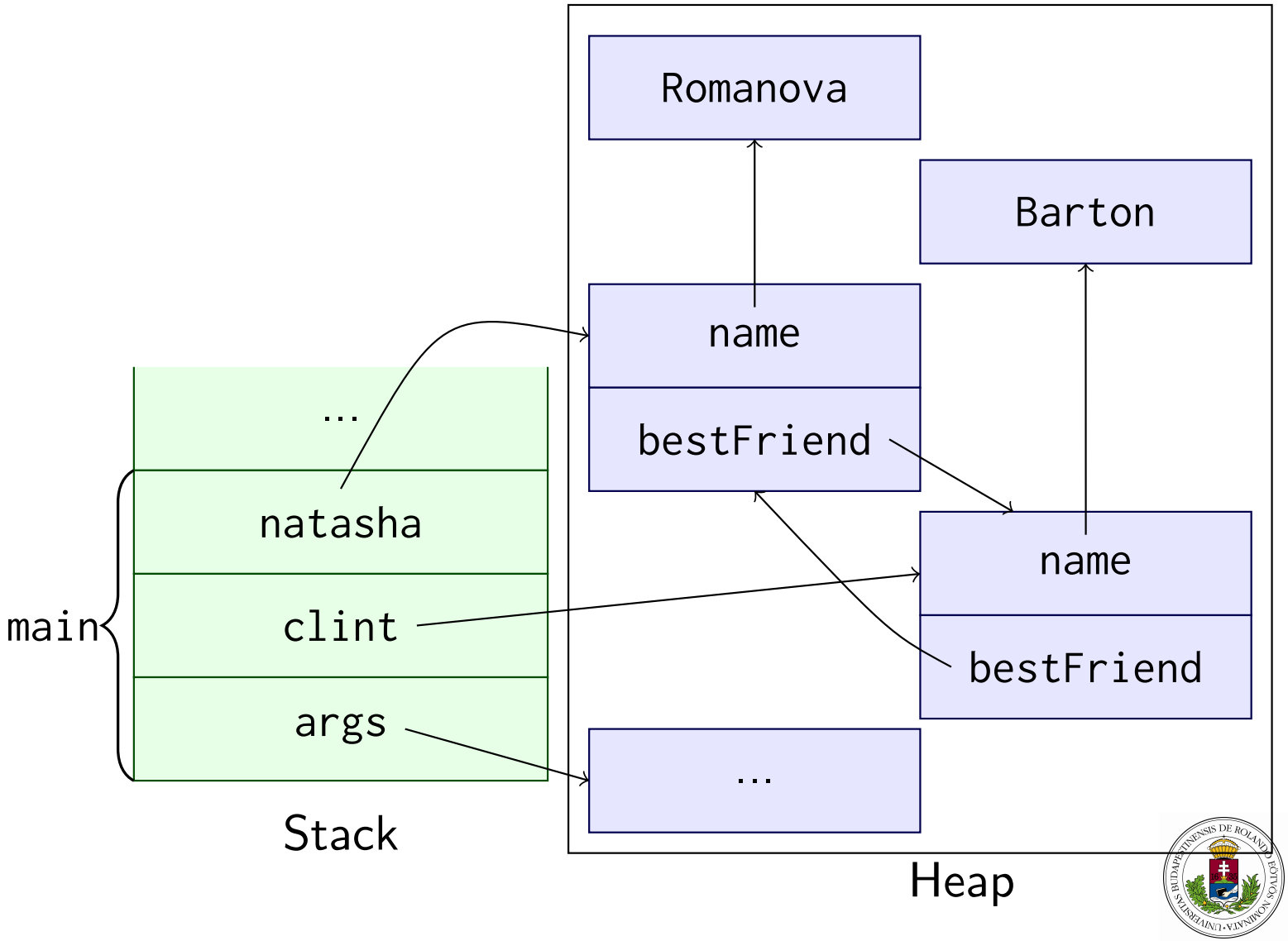


Bonyolultabb példa

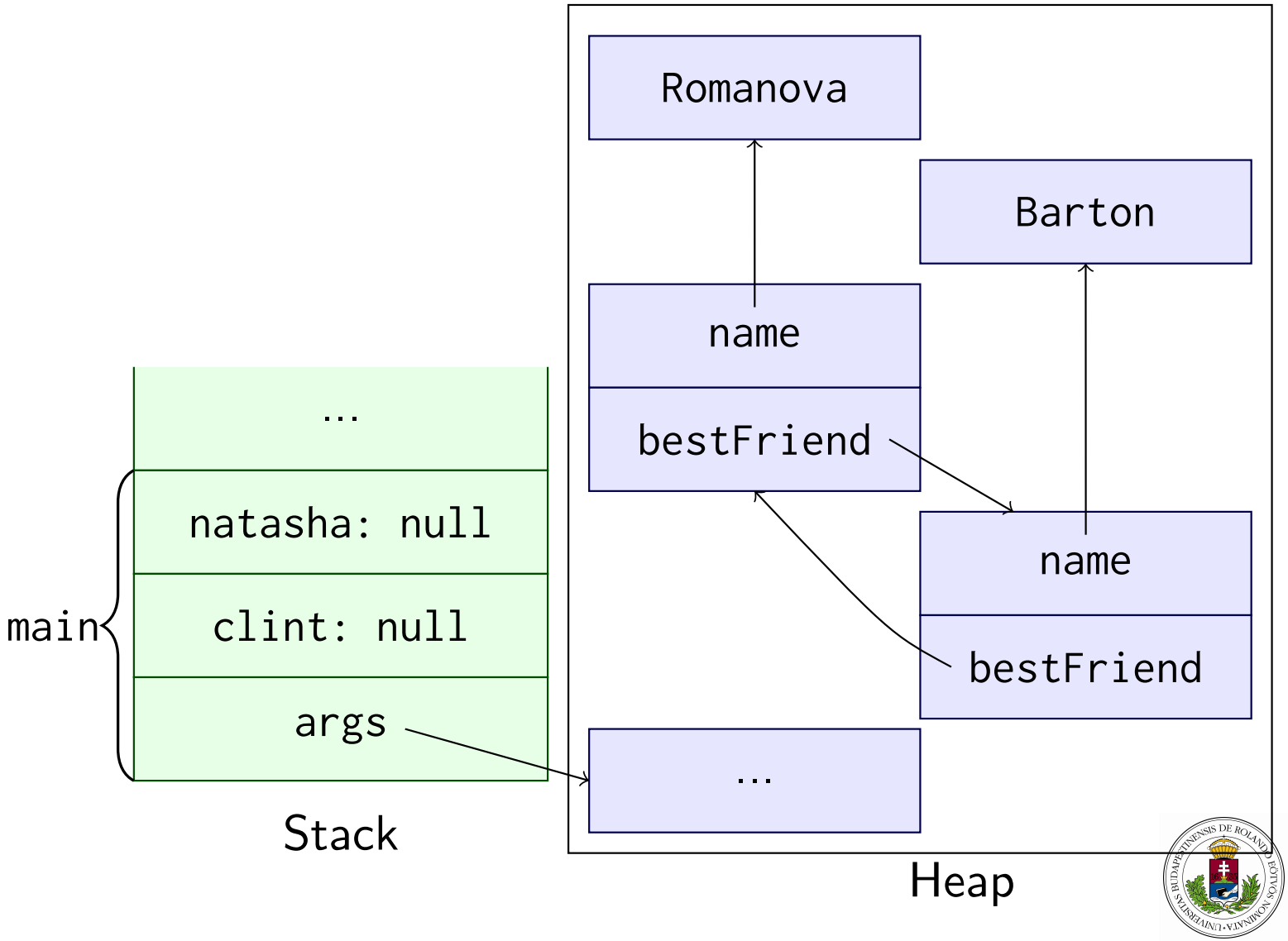
```
class Hero {
    String name;
    Hero bestFriend;
}
```

```
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 garbage collection

- Kiindulunk a vermen lévő referenciákból
- Megjelöljük a belőlük elérhető objektumokat
- Megjelöljük a megjelöltekből elérhető objektumokat
- Amíg tudunk újabbat megjelölni (tranzitív lezárt)
- A jelöletlen objektumok felszabadíthatók

Statikus mezők

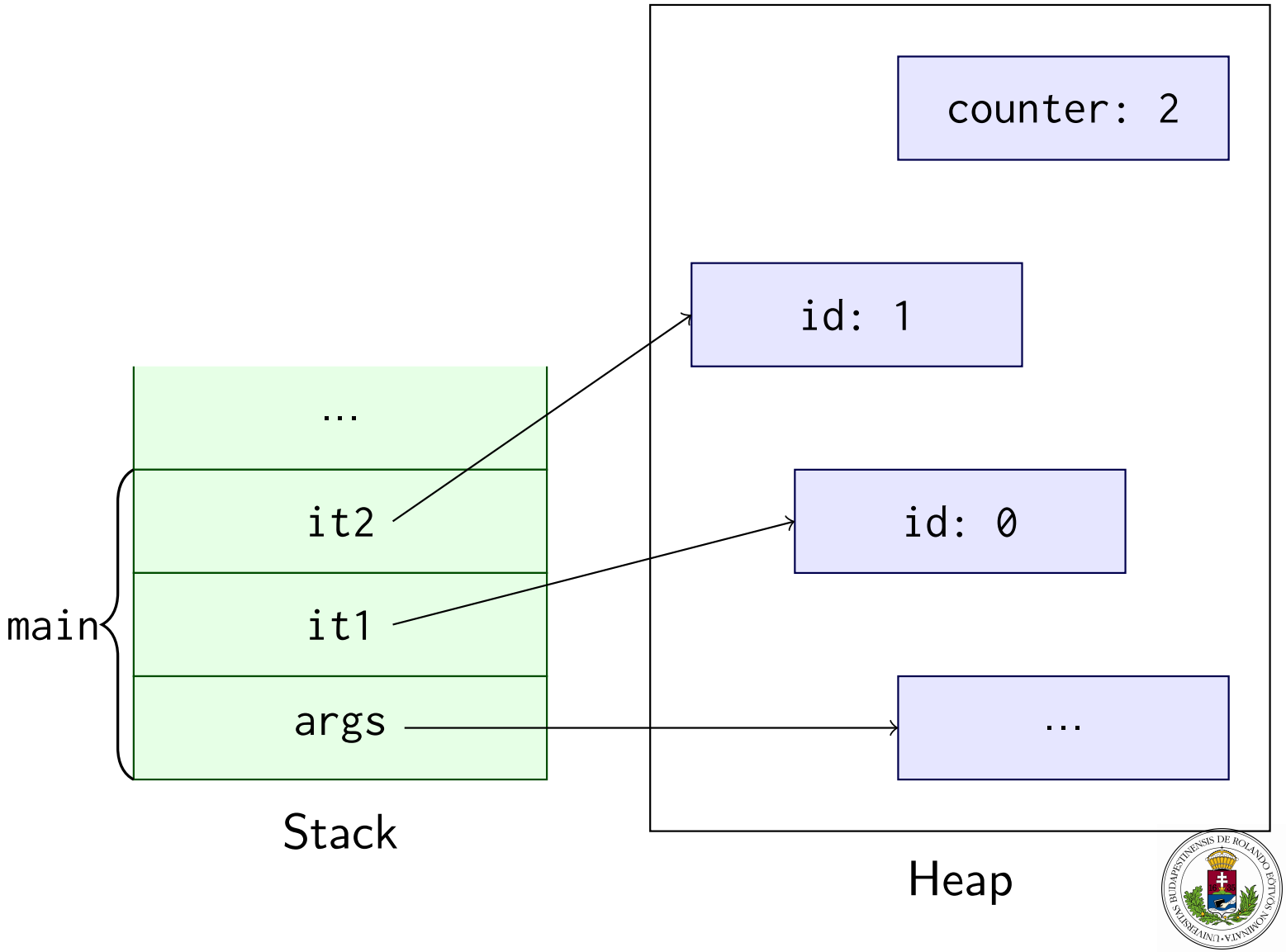
- Hasonló a C globális változóihoz
- Csak egy létezik belőle
- Az osztályon keresztül érhető el
- Mintha *statikus tárhelyen* lenne, nem az objektumokban

```
class Item {
    static int counter = 0;
}

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

Osztályszintű 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 it1 = new Item(), it2 = new Item();
        System.out.println( it1.id );
        System.out.println( it2.id );
        System.out.println( it1.counter );  // csúf, jelentése:
                                            // Item.counter
    }
}
```

`Item it1 = new Item(), it2 = 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();
    }
}
```

Outline

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 és feltöltése

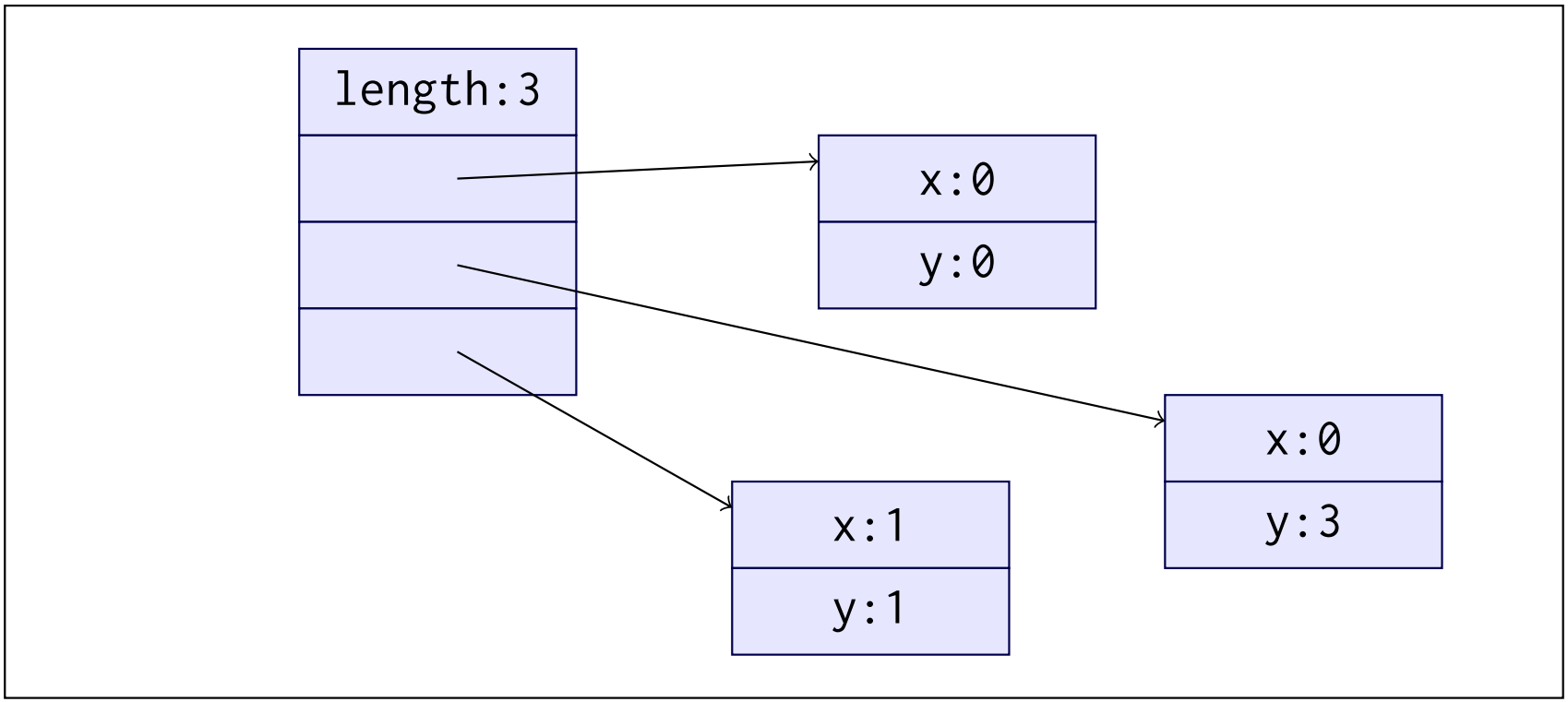
```
public static void main( String[] args ){
    int[] numbers = new int[args.length];   // 0-kkal feltöltve
    for( int i = 0; i < args.length; ++i ){
        numbers[i] = Integer.parseInt( args[i] );
    }
    java.util.Arrays.sort(numbers);
}
```

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) };
```



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 ){
        id100[i][i] = 1;
    }
    return matrix;
}
```

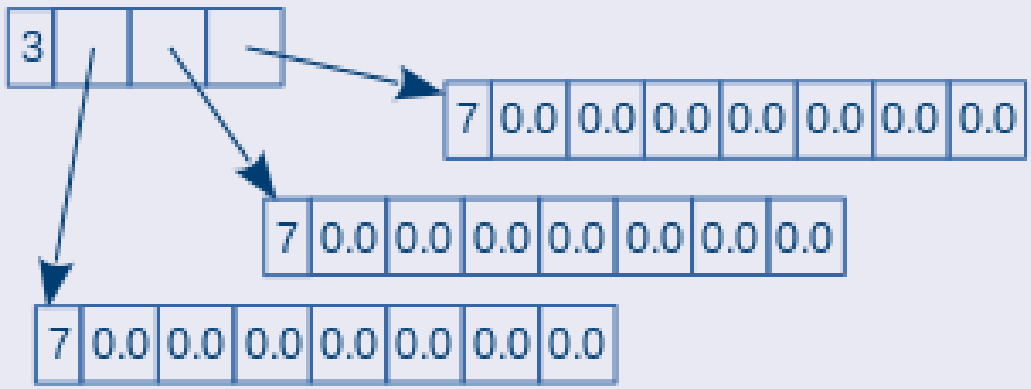
C versus Java

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];
```

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

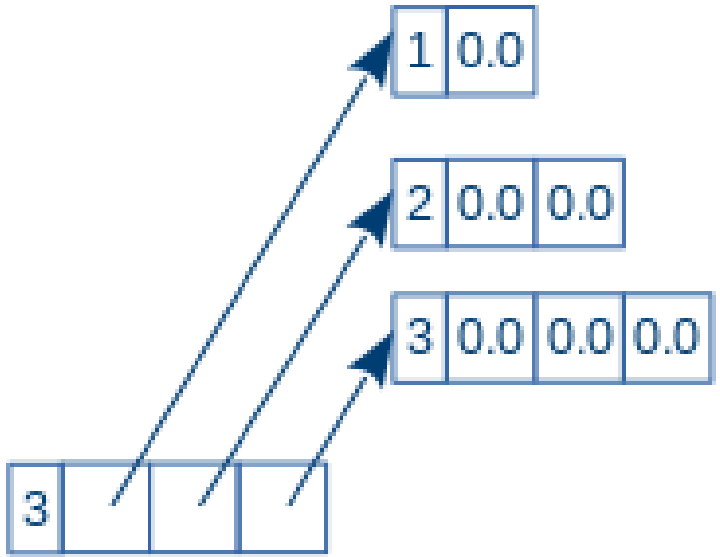
Tömbök tömbjének tömbje Javában

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

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

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`

Outline

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

enum versus int

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

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

Fix, zárt típusértékhalmaz

Csak a felsorolt típusértékek!

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

Fix, zárt típusértékhalmaz

Csak a felsorolt típusértékek!

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

- reflection segítségével sem példányosítható
- nem örökölhetünk 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;
    }
}
```

String is lehet switch-ben

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