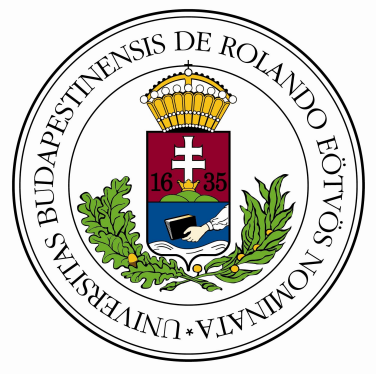
Programozási nyelvek – Java

Objektumelvű programozás



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

Absztrakció - típusmegvalósítás

- Egységbe zárás (encapsulation)
- Információelrejtés

Outline

Osztály, objektum, példányosítás

```
class Point {       // osztálydefiníció
    int x, y;       // mezők
}
```

Osztály, objektum, példányosítás

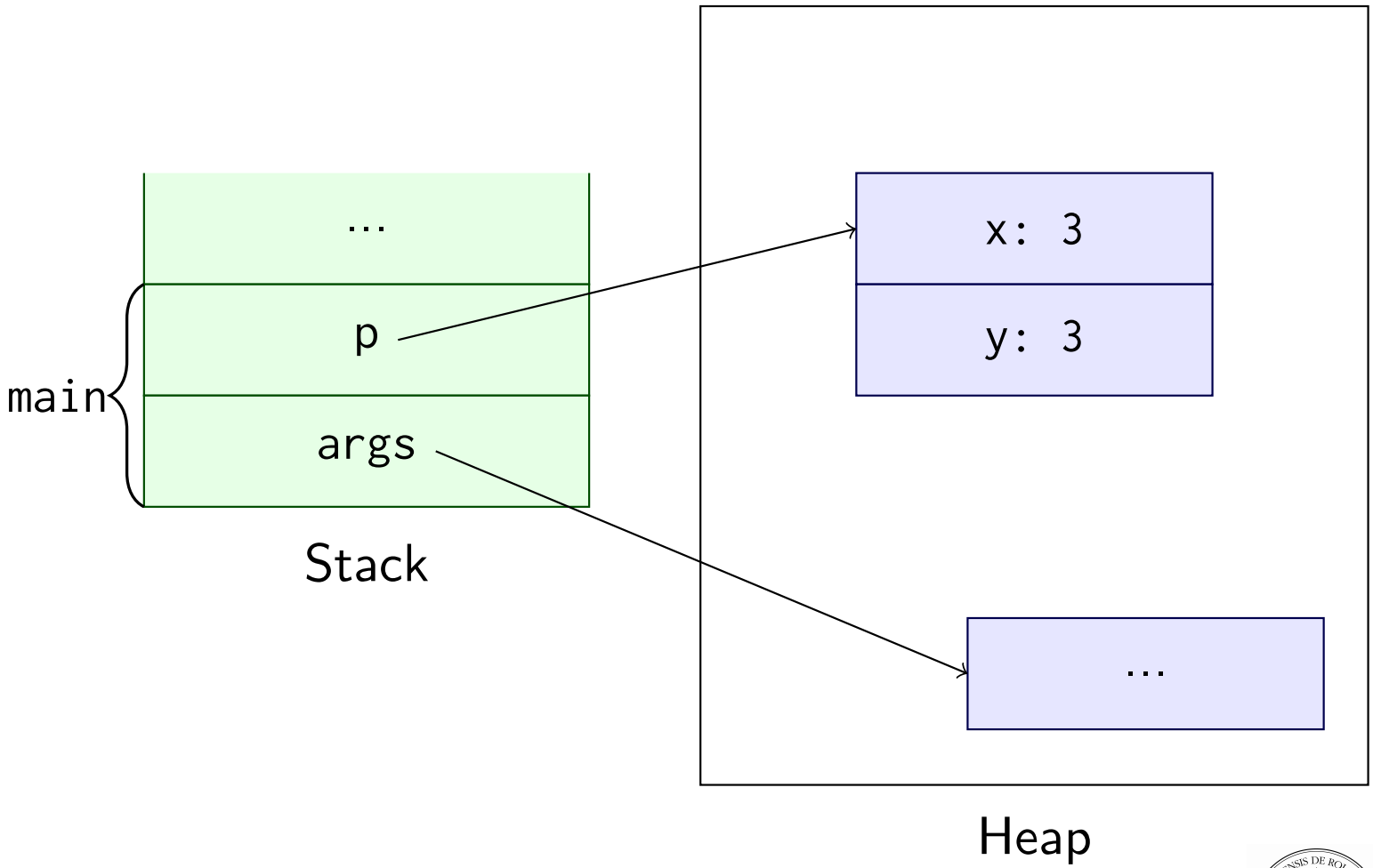
```
class Point {       // osztálydefiníció
    int x, y;       // mezők
}

class Main {
    public static void main( String[] args ){   // főprogram
        Point p = new Point();    // példányosítás (heap)
        p.x = 3;                  // objektum állapotának
        p.y = 3;                     // módosítása
    }
}
```

Fordítás, futtatás

```
$ ls
Main.java  Point.java
$ javac *.java
$ ls
Main.class  Main.java  Point.class  Point.java
$ java Point
Error: Main method not found in class Point, please define
 the main method as:
   public static void main(String[] args)
$ java Main
$
```

Stack és heap



Mezők inicializációja

```
class Point {
    int x = 3, y = 3;
}

class Main {
    public static void main( String[] args ){
        Point p = new Point();
        System.out.println(p.x + " " + p.y);   // 3 3
    }
}
```

Mező alapértelmezett inicializációja

Automatikusan egy nulla-szerű értékre!

```
class Point {
    int x, y = 3;
}

class Main {
    public static void main( String[] args ){
        Point p = new Point();
        System.out.println(p.x + " " + p.y);   // 0 3
    }
}
```

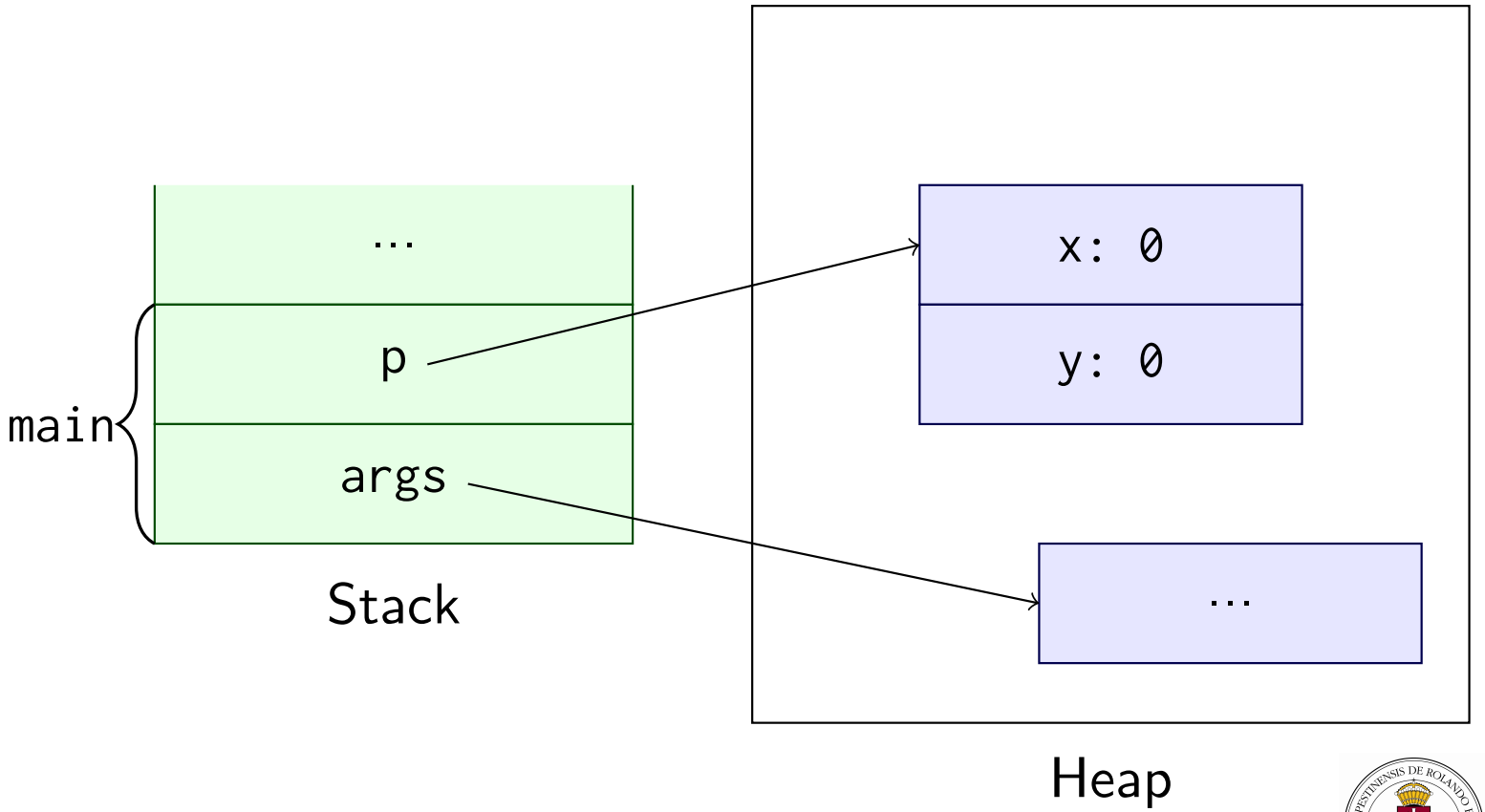
Metódus

```
class Point {
    int x, y;   // 0, 0
    void move( int dx, int dy ){   // implicit paraméter: this
        this.x += dx;
        this.y += dy;
    }
}

class Main {
    public static void main( String[] args ){
        Point p = new Point();
        p.move(3,3);                // p -> this, 3 -> dx, 3 -> dy
    }
}
```

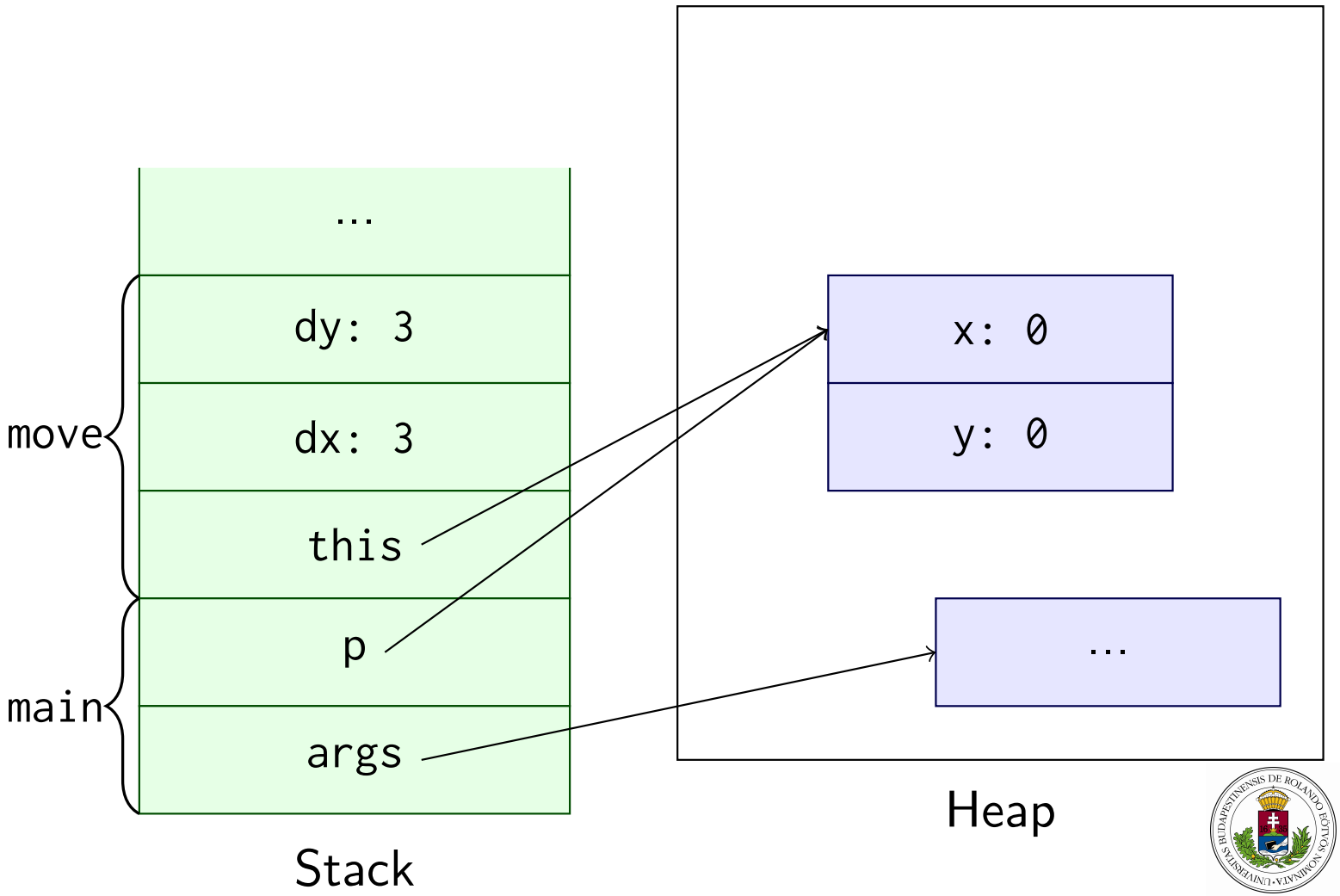
Metódus aktivációs rekordja – 1

```
Point p = new Point();
```



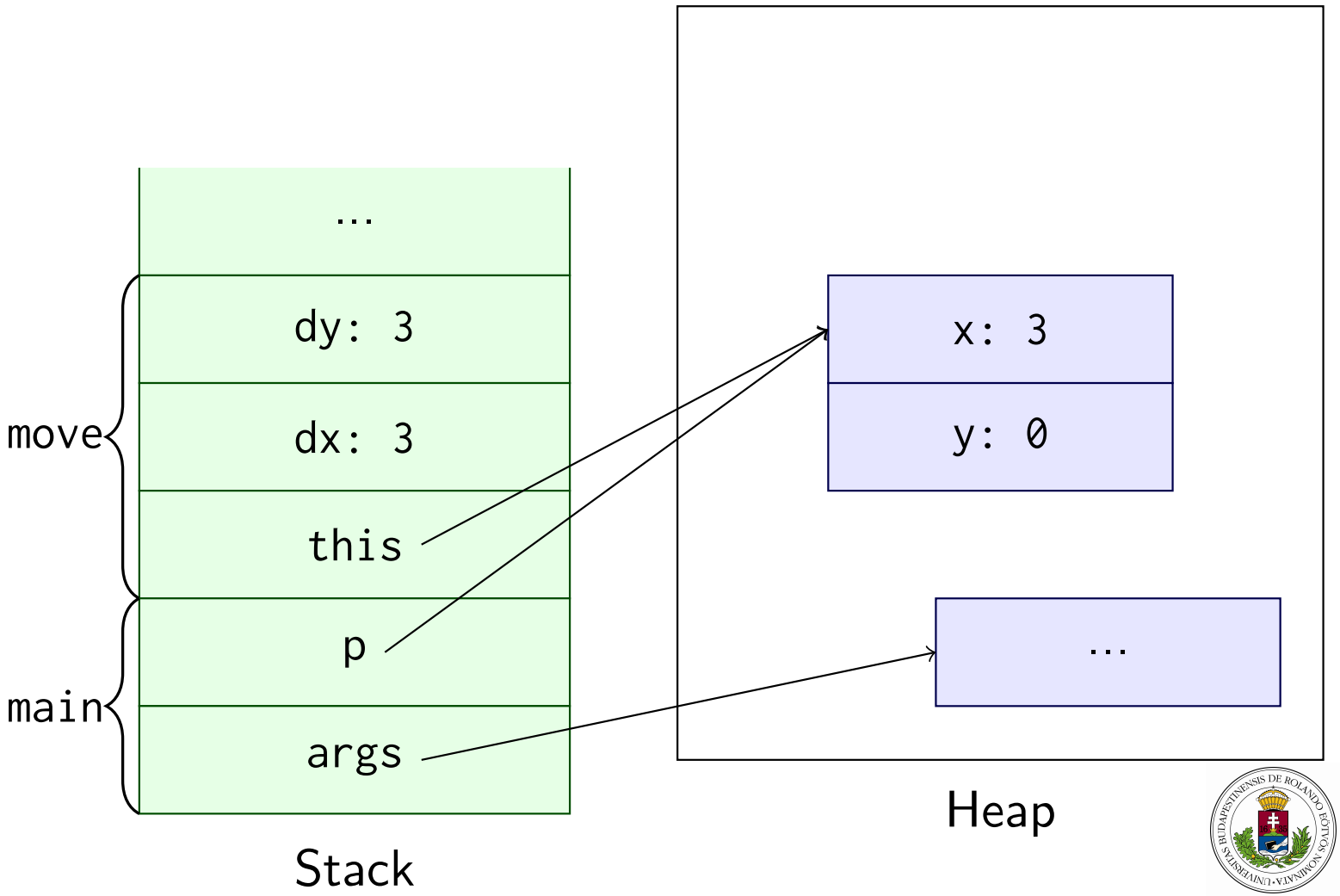
Metódus aktivációs rekordja – 2

```
p.move(3,3);
```



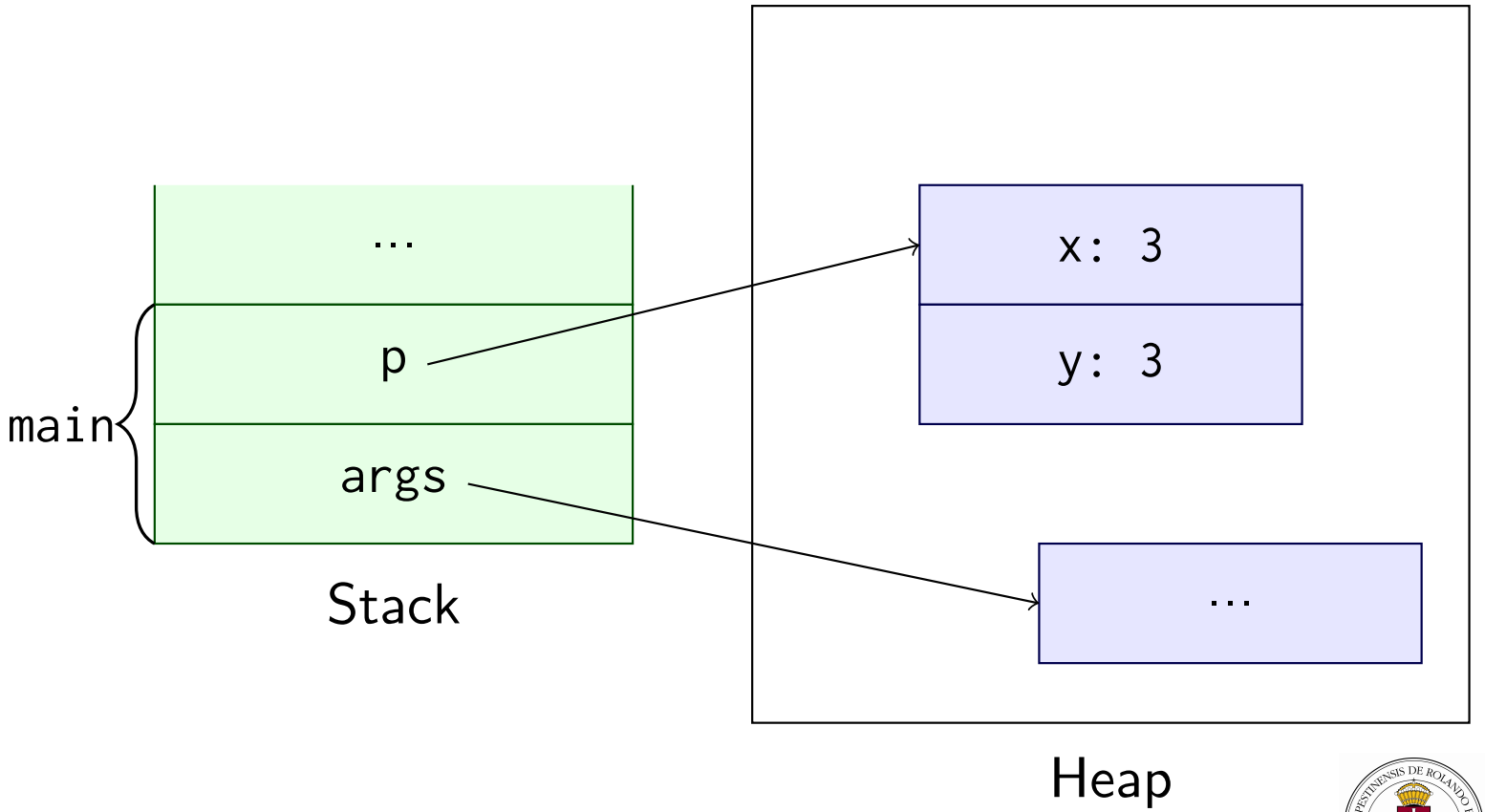
Metódus aktivációs rekordja – 3

```
this.x += dx;
```



Metódus aktivációs rekordja – 4

```
System.out.println(p.x + " " + p.y);
```



A this implicit lehet

```
class Point {
    int x, y;   // 0, 0
    void move( int dx, int dy ){
        this.x += dx;
        y += dy;
    }
}

class Main {
    public static void main( String[] args ){
        Point p = new Point();
        p.move(3,3);
    }
}
```

Inicializálás konstruktorral

```
class Point {
    int x, y;
    Point( int initialX, int initialY ){
        this.x = initialX;
        this.y = initialY;
    }
}

class Main {
    public static void main( String[] args ){
        Point p = new Point(0,3);
        System.out.println(p.x + " " + p.y);   // 0 3
    }
}
```

Inicializálás konstruktorral – a this elhagyható

```
class Point {
    int x, y;
    Point( int initialX, int initialY ){
        x = initialX;
        y = initialY;
    }
}

class Main {
    public static void main( String[] args ){
        Point p = new Point(0,3);
        System.out.println(p.x + " " + p.y);   // 0 3
    }
}
```

Nevek újrahasznosítása

```
class Point {
    int x, y;
    Point( int x, int y ){   // elfedés
        this.x = x;          // minősített (qualified) név
        this.y = y;          // konvenció
    }
}

class Main {
    public static void main( String[] args ){
        Point p = new Point(0,3);
        System.out.println(p.x + " " + p.y);   // 0 3
    }
}
```

Paraméter nélküli konstruktor

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

class Main {
    public static void main( String[] args ){
        Point p = new Point();
        System.out.println(p.x + " " + p.y);    // 0 0
    }
}
```

Alapértelmezett (default) konstruktor

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

class Main {
    public static void main( String[] args ){
        Point p = new Point();
        System.out.println(p.x + " " + p.y);    // 0 0
    }
}
```

Generálodik egy paraméter nélküli, üres konstruktor

```
Pont(){}
```

Outline

Egységbe zárás

```
class Time {
    int hour;
    int minute;
    Time( int hour, int minute ){
        this.hour = hour;
        this.minute = minute;
    }
    void aMinutePassed(){
        if( minute < 59 ){
            ++minute;
        } else { ... }
    } // (C) Monty Python
}
```

```
Time morning = new Time(6,10);
morning.aMinutePassed();
int hour = morning.hour;
```

Típusinvariáns

```
class Time {
    int hour;                                   // 0 <= hour < 24
    int minute;                                 // 0 <= minute < 60
    Time( int hour, int minute ){
        this.hour = hour;
        this.minute = minute;
    }
    void aMinutePassed(){
        if( minute < 59 ){
            ++minute;
        } else { ... }
    }
}
```

Értelmetlen érték létrehozása

```
class Time {
    int hour;
    int minute;
    Time( int hour, int minute ){
        this.hour = hour;
        this.minute = minute;
    }
    void aMinutePassed(){
        if( minute < 59 ){
            ++minute;
        } else { ... }
    }
}
```

```
Time morning = new Time(6,10);
morning.aMinutePassed();
int hour = morning.hour;

morning.hour = -1;
morning = new Time(24,-1);
```

Létrehozásnál típusinvariáns biztosítása

```
class Time {
    int hour;                                   // 0 <= hour < 24
    int minute;                                 // 0 <= minute < 60
    Time( int hour, int minute ){
        if (0 <= hour && hour < 24 && 0 <= minute && minute < 60){
            this.hour = hour;
            this.minute = minute;
        }
    }
    void aMinutePassed(){
        if( minute < 59 ){
            ++minute;
        } else { ... }
    }
}
```

Kerüljük el a „silent failure” jelenséget

```
class Time {
    int hour;                                   // 0 <= hour < 24
    int minute;                                 // 0 <= minute < 60
    Time( int hour, int minute ){
        if (0 <= hour && hour < 24 && 0 <= minute && minute < 60){
            this.hour = hour;
            this.minute = minute;
        } else {
            throw new IllegalArgumentException("Invalid time!");
        }
    }
    void aMinutePassed(){
        ...
    }
}
```

Kivétel

- Futás közben lép fel
- Problémát jelezhetünk vele
  - `throw` utasítás
- Jelezhet „dinamikus szemantikai hibát”
- Program leállását eredményezheti
- Lekezelhető a programban
  - `try–catch` utasítás

Futási hiba

```
class Main {
    public static void main( String[] args ){
        Time morning = new Time(24,-1);
    }
}
```

```
$ javac Time.java
$ javac Main.java
$ java Main
Exception in thread "main" java.lang.IllegalArgumentException:
Invalid time!
    at Time.<init>(Time.java:9)
    at Main.main(Main.java:3)
$
```

A mezők közvetlenül manipulálhatók

```
class Time {
    int hour;                          // 0 <= hour < 24
    int minute;                        // 0 <= minute < 60
    ...
}
```

```
class Main {
    public static void main( String[] args ){
        Time morning = new Time(6,10);
        morning.aMinutePassed();

        morning.hour = -1;             // ajjaj!
    }
}
```

Mező elrejtése: private

```
class Time {
    private int hour;                   // 0 <= hour < 24
    private int minute;                 // 0 <= minute < 60
    ...
}
```

```
class Main {
    public static void main( String[] args ){
        Time morning = new Time(6,10);
        morning.aMinutePassed();

        morning.hour = -1;              // fordítási hiba
    }
}
```

Idióma: privát állapot csak műveleteken keresztül

```
class Time {
    private int hour;                   // 0 <= hour < 24
    private int minute;                 // 0 <= minute < 60
    Time( int hour, int minute ){ ... }
    int getHour(){ return hour; }
    int getMinute(){ return minute; }
    void setHour( int hour ){
        if( 0 <= hour && hour <= 23 ){
            this.hour = hour;
        } else {
            throw new IllegalArgumentException("Invalid hour!");
        }
    }
    void setMinute( int minute ){ ... }
    void aMinutePassed(){ ... }
}
```

Getter-setter konvenció

Lekérdező és beállító művelet neve

```
class Time {
    private int hour;                    // 0 <= hour < 24
    int getHour(){ return hour; }
    void setHour( int hour ){
        if( 0 <= hour && hour <= 23 ){
            this.hour = hour;
        } else {
            throw new IllegalArgumentException("Invalid hour!");
        }
    }
    ...
}
```

Reprezentáció változtatása

```
class Time {
    private short minutes;
    Time( int hour, int minute ){
        if (0 <= hour && hour < 24 && 0 <= minute && minute < 60){
            minutes = 60*hour + minute;
        } else {
            throw new IllegalArgumentException("Invalid time!");
        }
    }
    int getHour(){ return minutes / 60; }
    int getMinute(){ return minutes % 60; }
    void setHour( int hour ){
        if( 0 <= hour && hour <= 23 ){
            minutes = 60 * hour + getMinute();
        } else {
            throw new IllegalArgumentException("Invalid hour!");
```

Információ elrejtése

- Osztályhoz szűk interfész
  - Ez „látszik” más osztályokból
  - A lehető legkevesebb kapcsolat
- Priváttá tett implementációs részletek
  - Segédműveletek
  - Mezők

Előnyök

- Típusinvariáns megőrzése könnyebb
- Kód könnyebb evolúciója (reprezentációváltás)
- Kevesebb kapcsolat, kisebb komplexitás