Java és web programozás

Kovács Kristóf

Budapesti Műszaki Egyetem

2015. 02. 11.

3. Előadás

```
public class Position {
    private float x;
    private float y;

    public Position(Position other) {
        this.x = other.x;
        this.y = other.y;
    }
    public Position(float xn, float yn) {
        x = xn;
        y = yn;
    }
    public Position() {
        this(0, 0);
    }
    public void translate(float xt, float yt) {
        x += xt;
        y += yt;
    }
```

```
    public float distance(Position other) {
        float xDist = this.x - other.x;
        float yDist = this.y - other.y;
        double dist = Math.sqrt(xDist * xDist + yDist * yDist);
        return (float)dist;
    }

    public void print() {
        System.out.println("(" + x + ", " + y + ")");
    }

    public static void main(String[] args) {
        Position p1 = new Position();
        Position p2 = new Position(3f, 4f);
        p1.translate(2, 1);
        float dist = p1.distance(p2);
        System.out.println(dist);
        p1.print();
    }
}
```

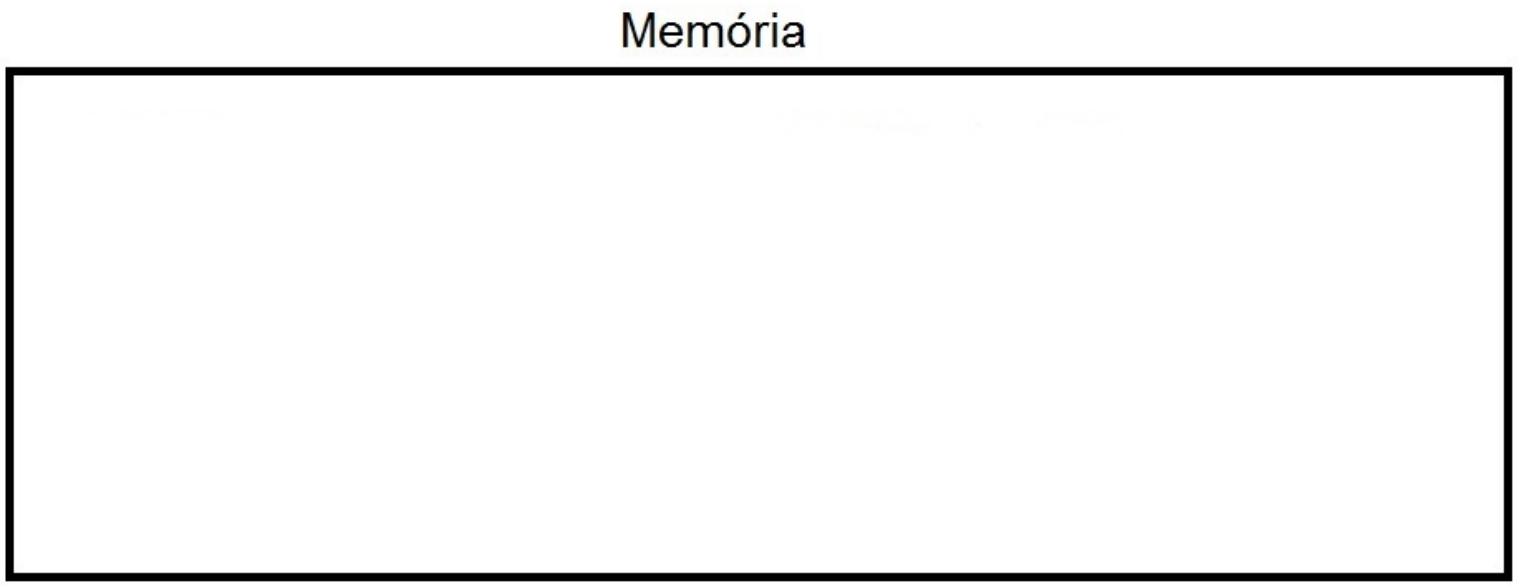
Memória kép

- Az előbbi *Position* osztály is elhelyezkedik valahol a memóriában. Rá úgy kell tekinteni mint egy mintára amiből lehet készíteni *Position* objektumokat.
- Amikor létrehozunk (*konstruálunk*) egy *Position* objektumot akkor használnunk kell az egyik *konstruktorát*.
- A java először lefoglalja a memóriában a helyet az objektumnak. Majd lefuttatja a konstruktort amivel a objektum adattagjai kezdőértéket kapnak.

Memória kép

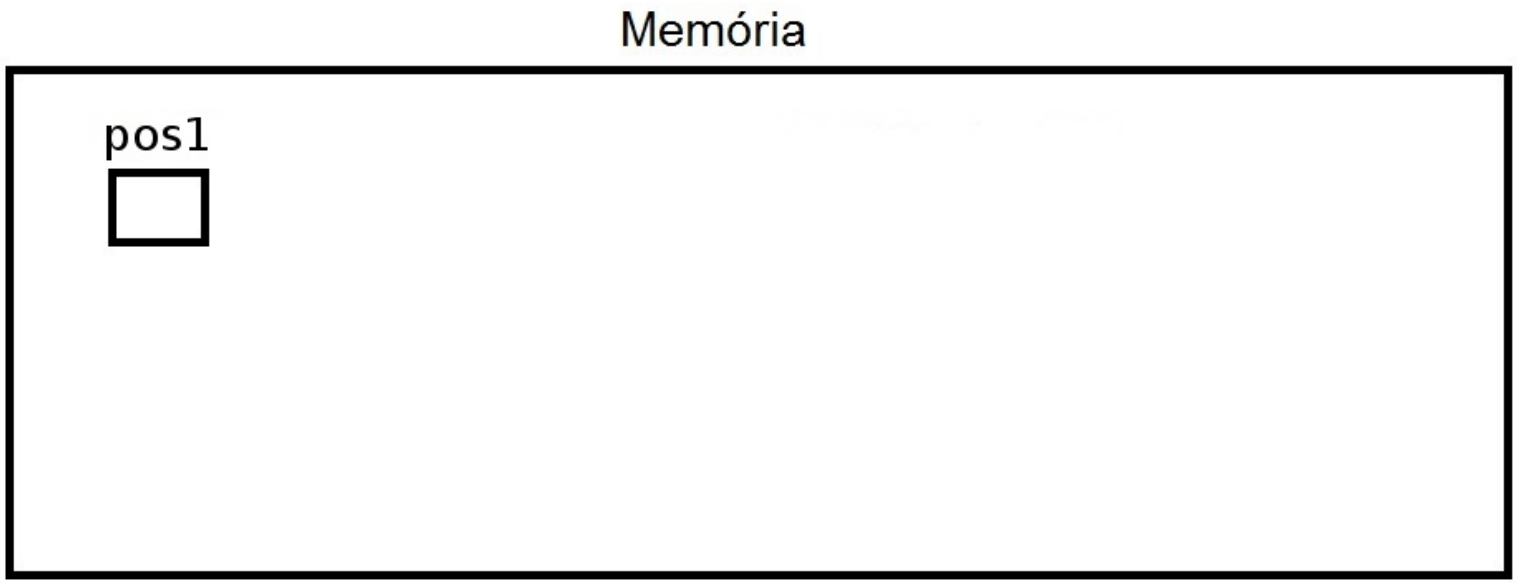
- Az előbbi *Position* osztály is elhelyezkedik valahol a memóriában. Rá úgy kell tekinteni mint egy mintára amiből lehet készíteni *Position* objektumokat.
- Amikor létrehozunk (*konstruálunk*) egy *Position* objektumot akkor használnunk kell az egyik *konstruktorát*.
- A java először lefoglalja a memóriában a helyet az objektumnak. Majd lefuttatja a konstruktort amivel a objektum adattagjai kezdőértéket kapnak.
- Amikor csak deklarálunk egy nem primitív változót ahhoz nem tartozik semmilyen tárolt objektum egészen addig amíg nem foglalunk le egyet és adjuk neki értékül.
- Minden változóra úgy kellene tekinteni, mintha pointerek lennének.

Példa



- Ez a memóriánk.

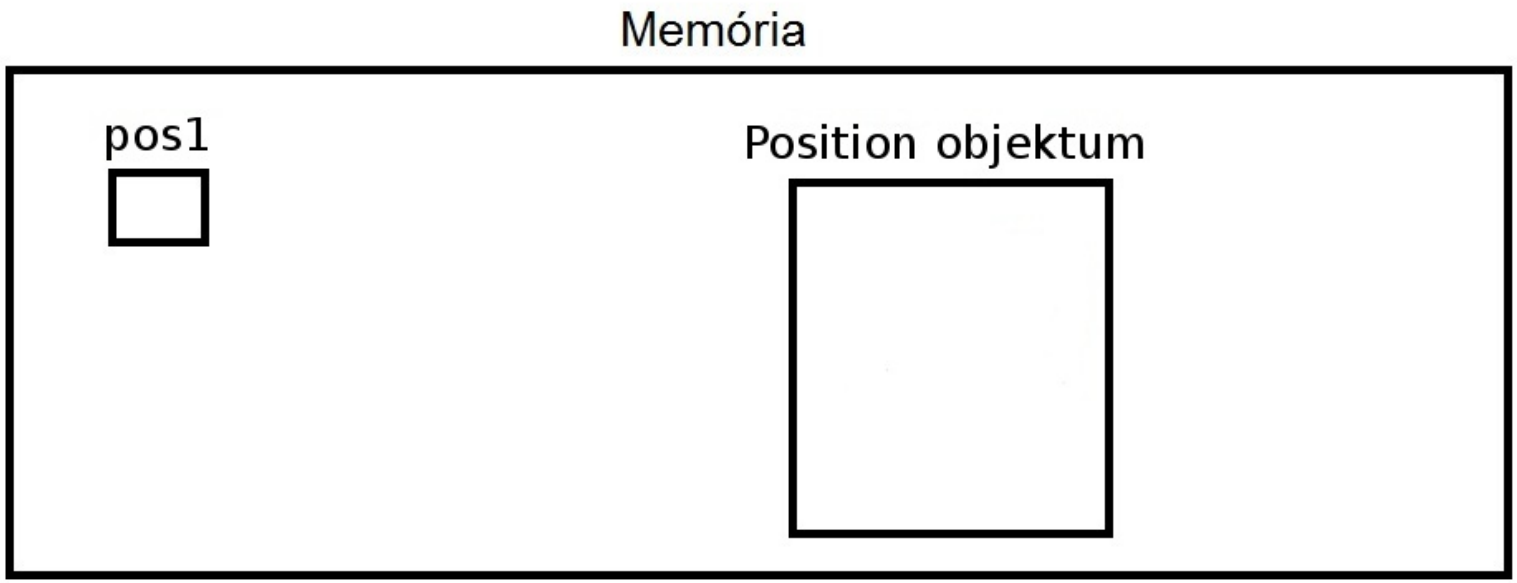
Példa




```
Position pos1;
```

- Létrehoztuk a pos1 *változót*

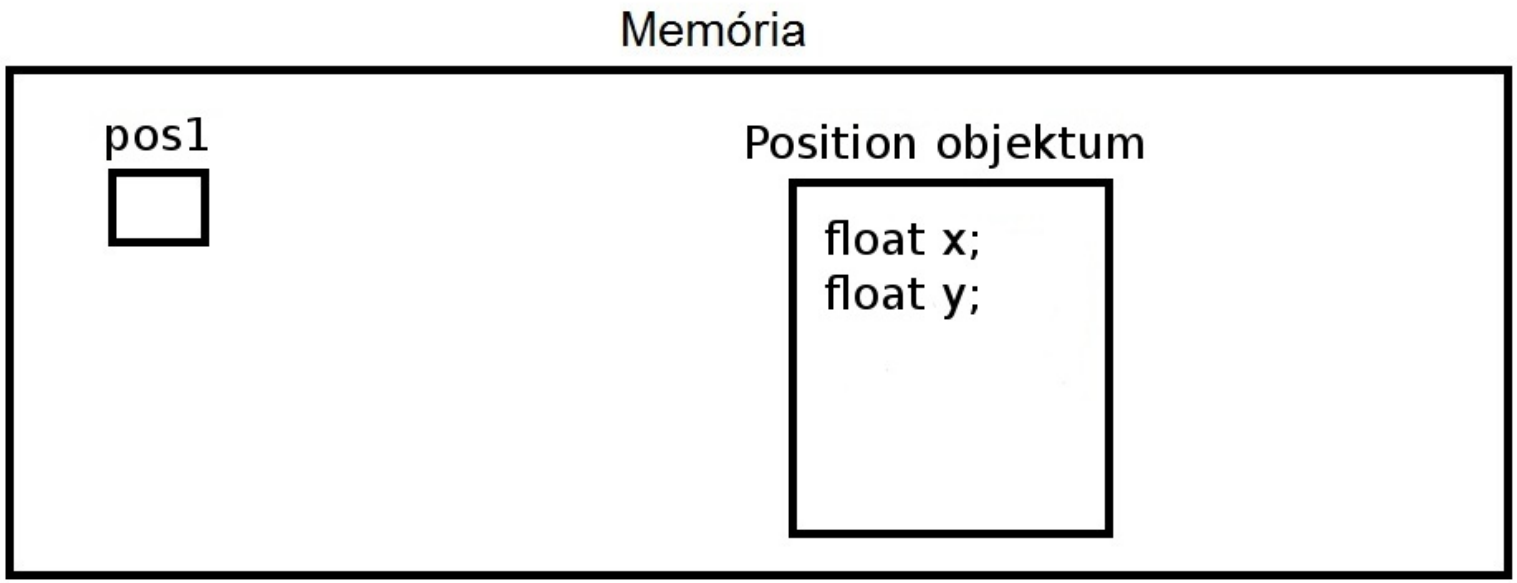
Példa




```
User pos1;
new Position(3.0f, 2.5f);
```

- Létrehoztunk egy Position *objektumot*

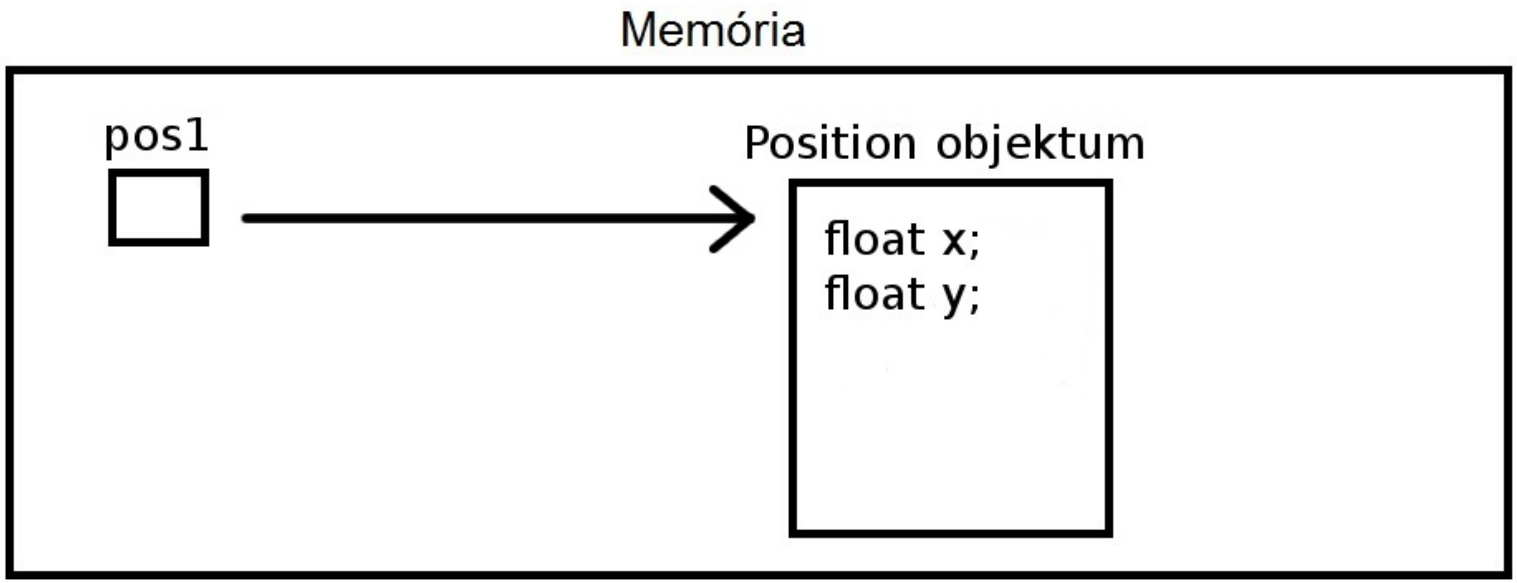
Példa




```
User pos1;
new Position(3.0f, 2.5f);
```

- Ennek lefutott a *konstruktora* így az adattagjainak van értéke

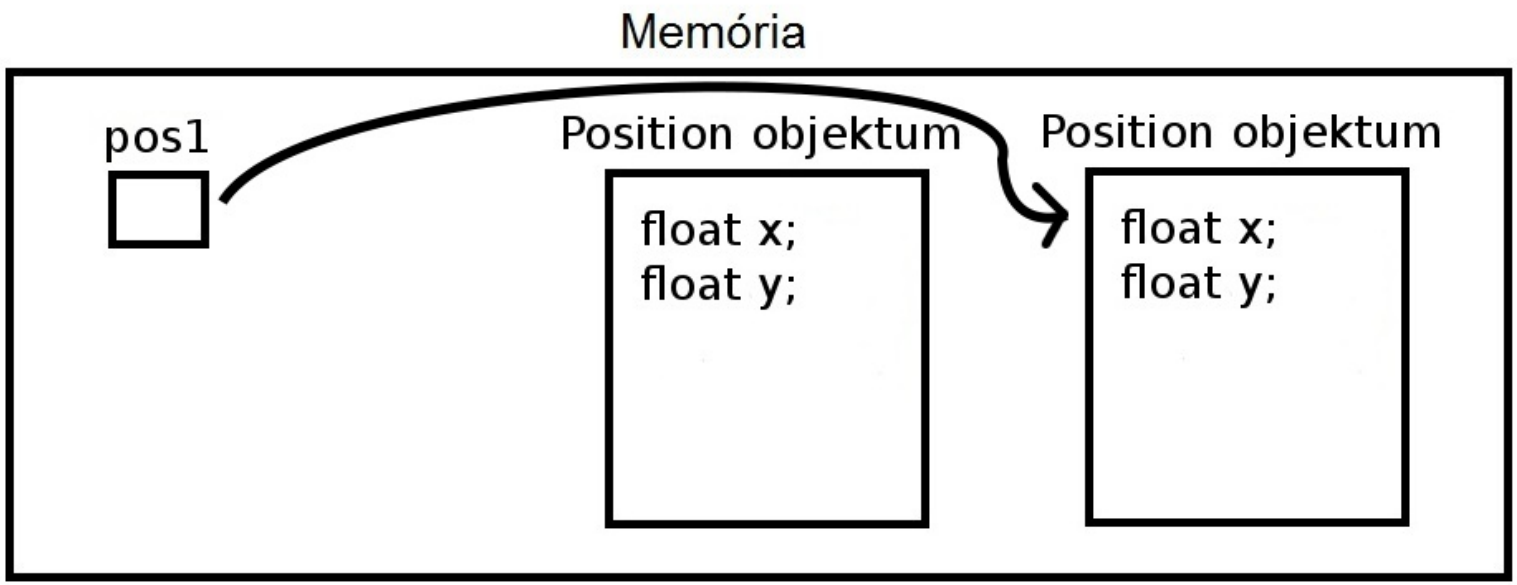
Példa




```
User pos1;
pos1 = new Position(3.0f, 2.5f);
```

- Hozzarendeltük a változóhoz az objektumot

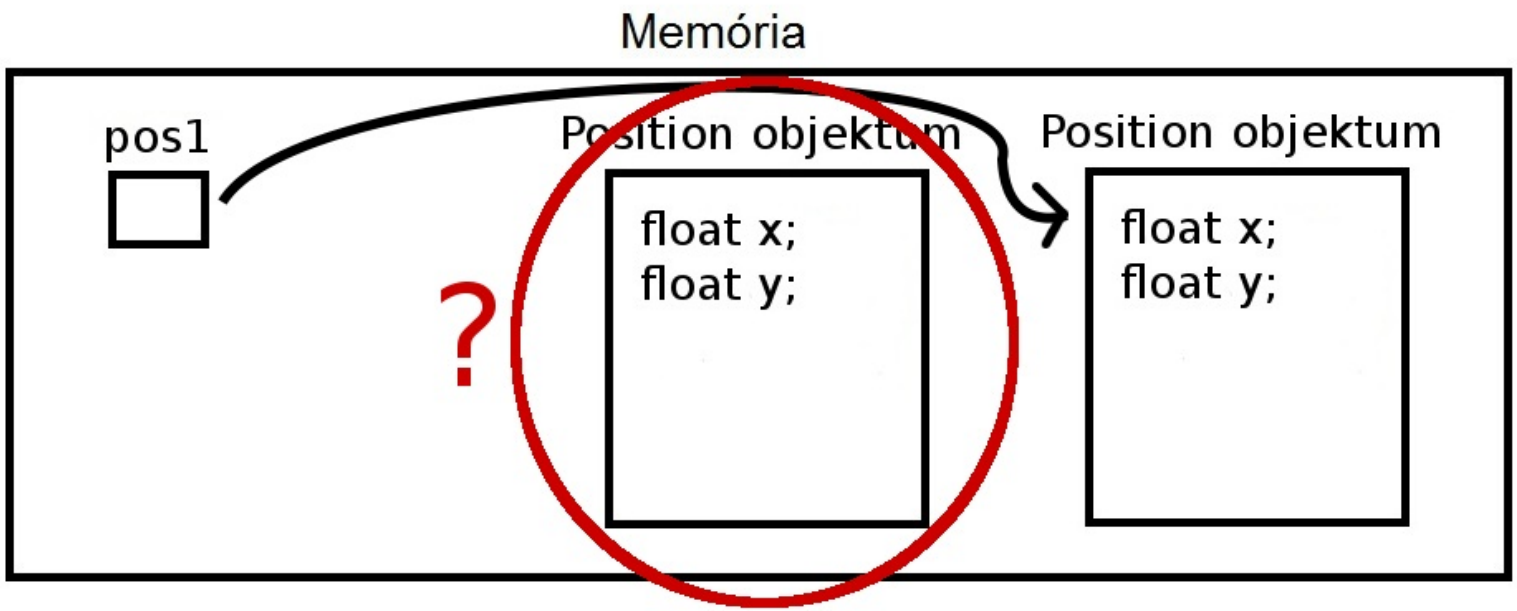
Példa




```
User pos1;
pos1 = new Position(3.0f, 2.5f);
pos1 = new Position(5.0f, -1.0f);
```

- Létrehoztunk egy új objektumot és a *pos1*-hez rendeltük

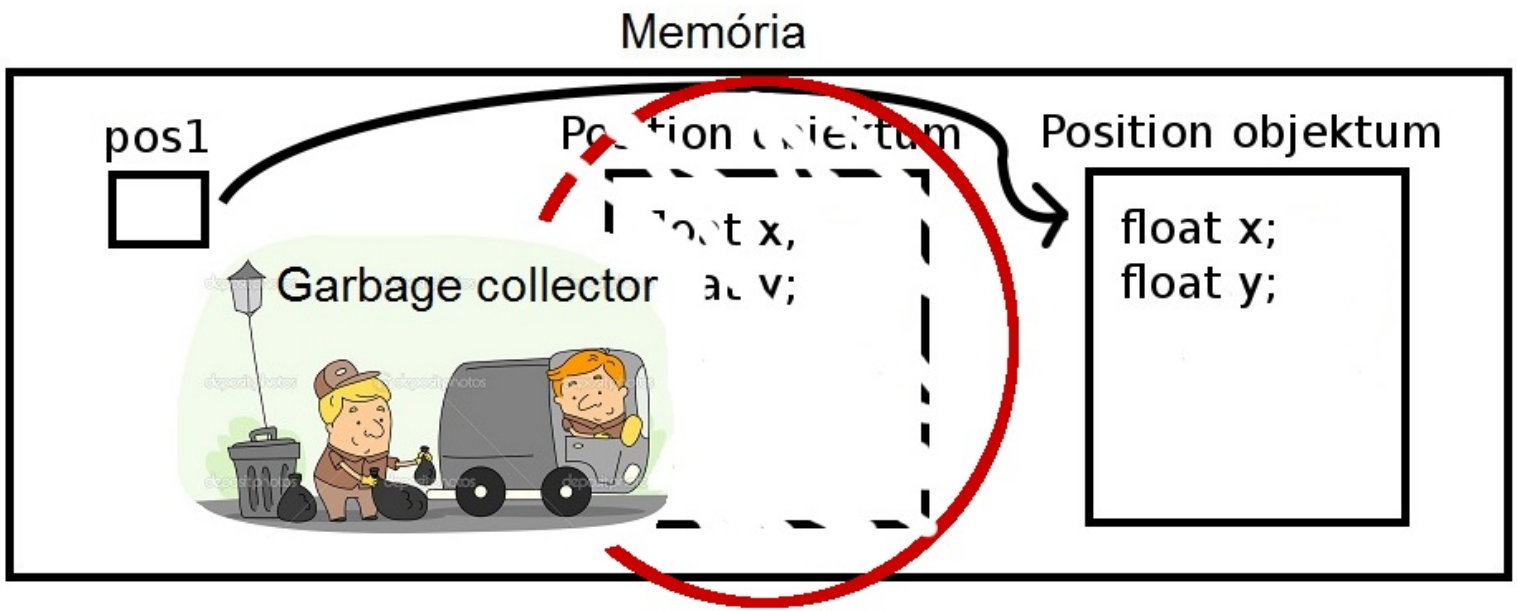
Példa




```
User pos1;
user1 = new Position(3.0f, 2.5f);
user1 = new Position(5.0f, -1.0f);
```

- Mi lesz most az eredeti objektumunkkal?

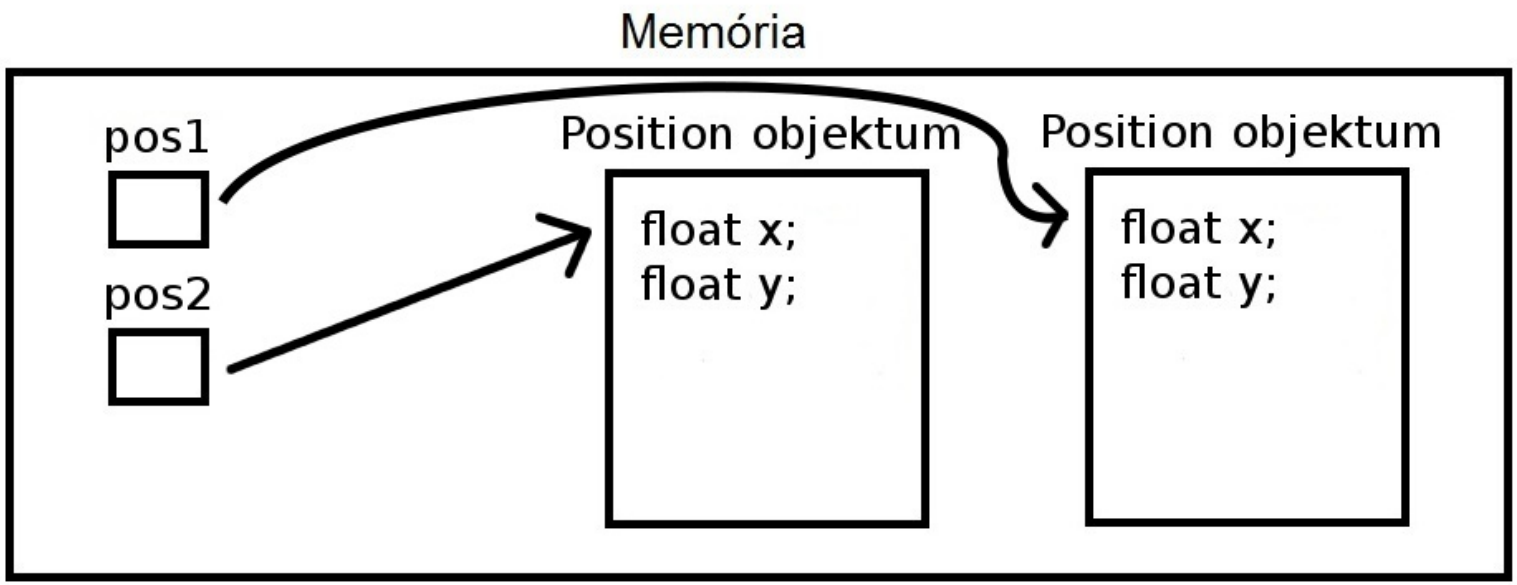
Példa




```
User pos1;
user1 = new Position(3.0f, 2.5f);
user1 = new Position(5.0f, -1.0f);
```

- Valamikor jön majd a *garbage collector* és felszabadítja

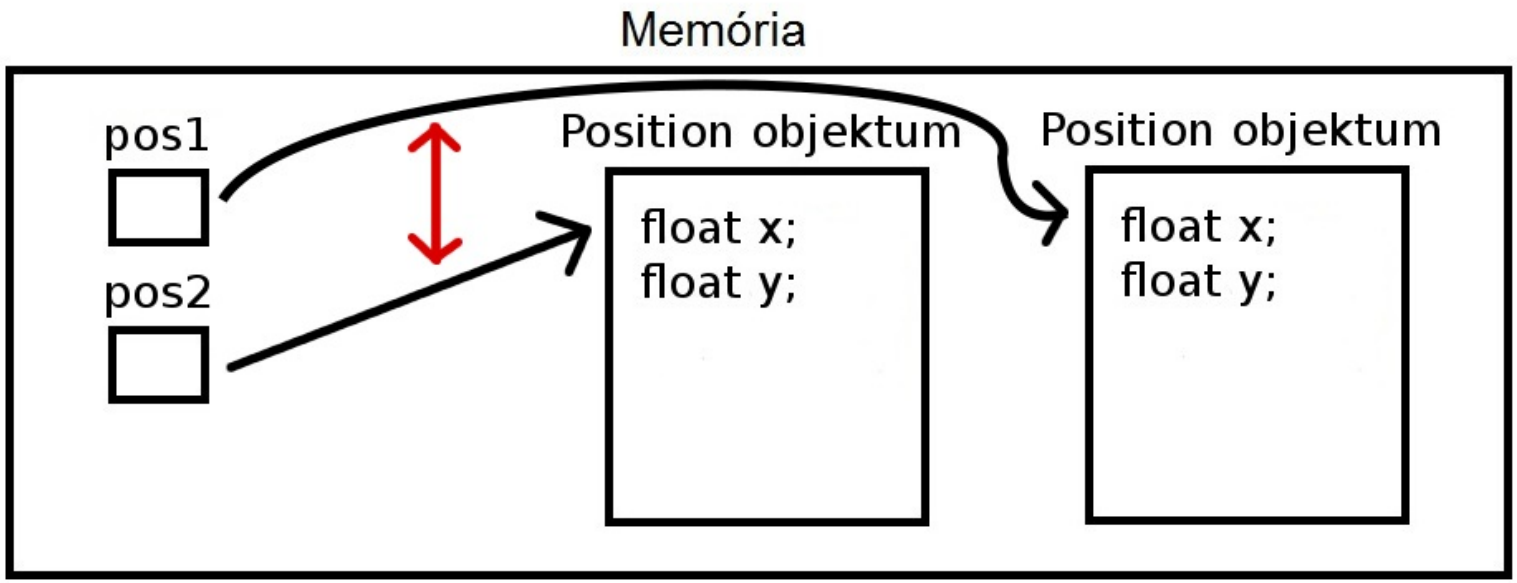
Példa




```
User pos1;
User pos2 = new Position(3.0f, 2.5f);
user1 = new Position(5.0f, -1.0f);
```

- Két objektumot össze szeretnénk hasonlítani

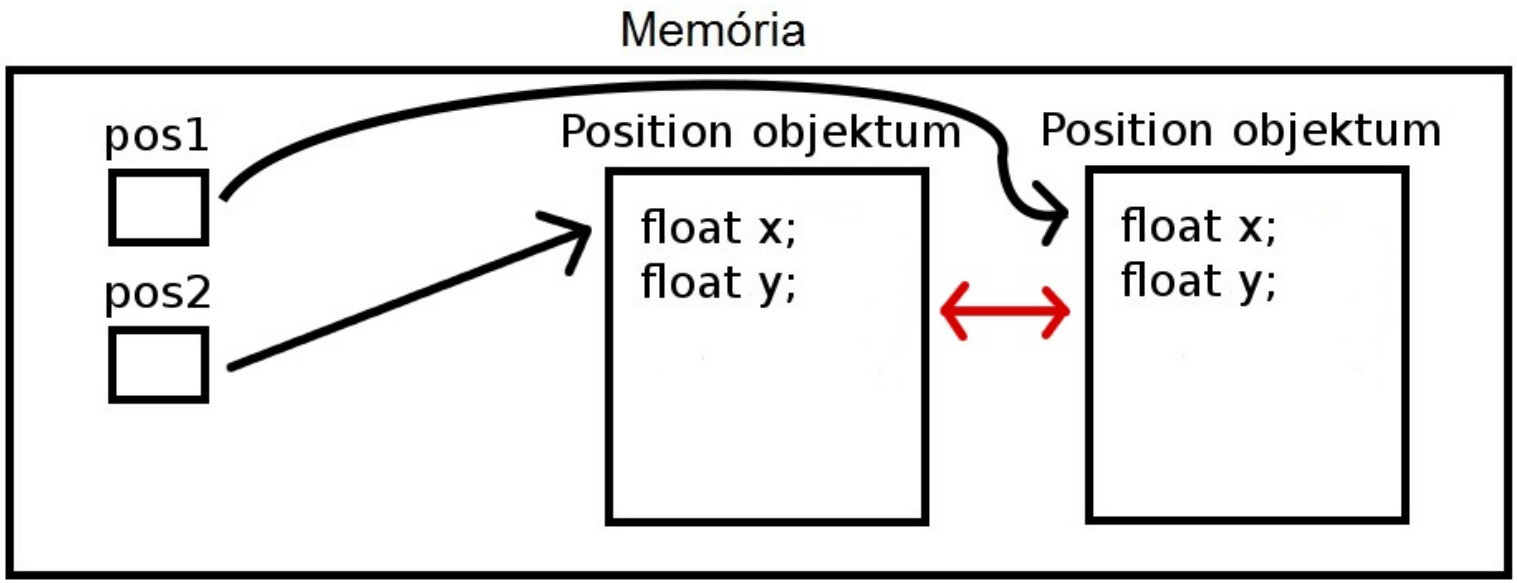
Példa




```
User pos1;
User pos2 = new Position(3.0f, 2.5f);
user1 = new Position(5.0f, -1.0f);
pos1 == pos2;
```

- Így a user változók hasonlítódnak össze, azaz az derül ki, hogy ugyanarra az objektumra mutatnak-e.

Példa




```
User pos1;
User pos2 = new Position(3.0f, 2.5f);
user1 = new Position(5.0f, -1.0f);
pos1.equals(pos2);
```

- Így már valóban az objektumok hasonlítódnak össze.

Garbage collector

- Mint majdnem minden programnyelvben itt is igaz az, hogy egy blokkban létrehozott változó csak a blokkon belül létezik.
- Az objektumokra ez nem igaz, a memóriában maradnak azután is, hogy vége egy blokknak.
- Viszont, ha beindult a garbage collector és talál olyan objektumot amire nincs hivatkozás. Azaz sehonnan nem lehet elérni (mint az előző példában). Akkor felszabadítja.

Garbage collector

- Mint majdnem minden programnyelvben itt is igaz az, hogy egy blokkban létrehozott változó csak a blokkon belül létezik.
- Az objektumokra ez nem igaz, a memóriában maradnak azután is, hogy vége egy blokknak.
- Viszont, ha beindult a garbage collector és talál olyan objektumot amire nincs hivatkozás. Azaz sehonnan nem lehet elérni (mint az előző példában). Akkor felszabadítja.
- A garbage collector nem folyamatosan működik, hanem bizonyos időpillanatokban elindul.
- Rá lehet kényszeríteni a javat, hogy indítsa el a garbage collectort.

Statikus adattagok / metódusok

- A statikus adattagok az osztályhoz tartoznak és nem az objektumokhoz.
- Ez azt is jelenti, hogy ha valami statikus már akkor létezik, amikor még egy objektumot se hoztunk létre az adott osztályból.

Statikus adattagok / metódusok

- A statikus adattagok az osztályhoz tartoznak és nem az objektumokhoz.
- Ez azt is jelenti, hogy ha valami statikus már akkor létezik, amikor még egy objektumot se hoztunk létre az adott osztályból.
- Ezért kell a *main* függvényeknek mindig statikusnak lenniük, mert őket anélkül kell tudnunk meghívni, hogy létrehoztunk volna objektumot.
- Statikus adattagból nem jön létre több ha létrehozunk egy új objektumot, hisz az osztályhoz tartozik. Így egy bizonyos statikus adattagból mindig csak egy van.

Példa statikus adattagra

```
public class Pelda {
    public static int szamlalo = 0;

    public Pelda() {
        Pelda.szamlalo++;
    }

    public static void main(String[] args) {
        Pelda pelda1 = new Pelda();
        Pelda pelda2 = new Pelda();
        Pelda pelda3 = new Pelda();
        Pelda pelda4 = new Pelda();
        System.out.println(Pelda.szamlalo); // 4
    }
}
```

Static

- Láthatjátok, hogy a statikus adattagokat nem az objektumon keresztül (*this*) érjük el, hanem az osztalyon keresztük.
- Statikus adattagok használat nem ajánlott nagyobb projectekben ahol esetleg megjelenik a párhuzamosítás.

Static

- Láthatjátok, hogy a statikus adattagokat nem az objektumon keresztül (*this*) érjük el, hanem az osztalyon keresztük.
- Statikus adattagok használat nem ajánlott nagyobb projectekben ahol esetleg megjelenik a párhuzamosítás.
- Amire viszont mindig hasznos, hogy konstansokat definiáljatok.
- public static final float pi = 3.14;

Konstansok

- A konstansok megváltoztathatatlanok, azonnal értéket kell nekik adni és ezután értekadás jobb oldalán szerepelhet csak.
- Bármilyen változót lehet konstansnak definiálni.
- Metódus is lehet konstans (*final*). Ekkor az adott metódus módosíthatatlan. Ilyenről jelenleg még nem tanultunk, de a későbbiekben elő fog jönni.

Összefoglalás

- Változó és objektum két külön életet él.
- Emiatt sokszor nem egyértelmű egy művelet eredménye. Érdemes mindig pointerekkel gondolkodni, ekkor átláthatóbb lesz.
- Statikus dolgok az osztályhoz tartoznak és nem az objektumhoz.
- Statikus függvények meghívhatók objektum létrehozása nélkül. Ez azt is jelenti, hogy statikus objektumokban a *this* kulcsszónak nincs értelme.
- Konstansokat definiálhattok a *final* kulcsszóval.