El?z? - Fel - Következ?

Tartalomjegyzék

Referencia, Konstansság

Legyen **Osztaly** egy java osztály, mindegy is, hogy mi van benne. Tegyük fel, hogy van egy **setName** metódusa, ami valamilyen adattagot megváltoztat az osztályban.

```
public class Osztaly
{
    private String string_;
    public Osztaly()
    {
        string_ = "";
    }
    public void SetName(String newName)
    {
        string_ = newName;
    }
    public String GetName()
    {
        return string_;
    }
}
```

Mit kezdhetünk ezzel az osztállyal egy tömbben:

```
public static void main(String[] args)
    {
        Osztaly[] tomb = new Osztaly[3];
        tomb[0] = new Osztaly();        // létrehozok egy üres példányt
        tomb[0].SetName("Steve");       // átállíthatunk egyes adatokat benne

        tomb[1] = new Osztaly();
        tomb[1].SetName("Stefan");
        tomb[2] = new Osztaly();
        tomb[2].SetName("Istvan");

        Osztaly x = tomb[0];
        x = new Osztaly();    // Szemben ezzel:
                              // x.SetName("Adam");
```

```
        System.out.println(tomb[0].GetName()); // ez az eredeti nevet tartalmazza-e?
    }
```

Primitív típusok

Az alábbi típusok *primitívek*, azaz a szóban forgó értéket közvetlenül tárolják.

Minden más típus (pl **String**, **int[]**, saját osztály) referencia, azaz az objektumra mutatnak.

Értékek megváltoztatása

Ez például azt vonja maga után, hogy ha egy tömböt (nem-primitív értéket) így átírunk:

```
int[] x = new int[5];
x[0] = 1
x[1] = 2
int[] y = x;
y = new int [10];
```

Akkor **y** nem írja felül **x**-et, csupán mostantól másra mutat az **y**.

Hasonló okokból lehet két változót összekapcsolni.

```
int[] x = new int[5];
int[] y = x;

x[0] = 1
x[1] = 2
```

Ekor az utolsó két parancs **y**-ra is hatással van, mert lényegében ?k ugyan azok (ugyan arra referálnak).

Copy konstruktor

Feladat

Írjuk meg a **Complex** osztály *copy konstruktor*át!

```
public Complex(Complex other) {
        // TODO
    }
```

Feladat

Írjuk meg a **ComplexVector** osztály *copy konstruktor*át!

```
public ComplexVector(ComplexVector other) {
        // TODO
    }
```

Függvényhívásnál

Ha írunk egy függvényt (metódust) akkor annak paraméterei **mindig érték szerint** adódnak át, viszont ez mást jelent primitív típus és referencia esetén.