El?z? - Fel - Következ?

# Tartalomjegyzék



# Overriding

## Nem statikus

Ha egy osztályban van egy ugyanolyan szignatúrájú metódus, mint az ?sében, akkor beszélünk

- felülírás, felüldefiniálás
- override
- túlterhelés

-r?l

```
public class Parent
{
    float f(float x)
    {
        ...
    }
}

public class Child extends Parent
{
    float f(float x)
    {
        ...
    }
}
```

Ekkor az alábbi **f** hívások mást adnak vissza, attól függ?en, hogy hogyan írtuk felül.

```
Parent parent = new Parent();
Child child = new Child();

parent.f(3.14f); // <- szül?ben lév? f
child.f(3.14f);  // <- gyerekben lév? f
```

Viszont ez akkor is így történik, ha minkett?t az ?sosztlyként tárolom.

```
Parent objects = new Parent[2];
objects[0] = new Parent();
objects[1] = new Child(); // leszármazott osztály az ?sosztállyá konvertálódik

objects[0].f(3.14f); // <- szül?ben lév? f
```

```
objects[1].f(3.14f);  // <- gyerekben lév? f
```

Vagyis az íly módon meghívott **f** ?rzi azt, hogy hogyan hoztuk létre az aktuális példányt.

Ez a *dinamikus polimorfizmus* és a hatását egy felüldefiniált metóduson keresztül láthattuk.

## Statikus

Nem ez a helyzet statikus metódusnál.

```
public class Parent
{
    static float f(float x)
    {
        ...
    }
}

public class Child extends Parent
{
    static float f(float x)
    {
        ...
    }
}
```

Ekkor hiába **Child** konstruktorral jött létre egy **Parent** típusú változó, akkor is a **Parent.f** hívódik meg.

```
Parent[] objects = new Parent[2];
objects[0] = new Parent();
objects[1] = new Child(); // leszármazott osztály az ?sosztállyá konvertálódik

objects[0].f(3.14f); // <- Parent.f
objects[1].f(3.14f);  // <- Parent.f
```

Figyeljük meg, hogy statikus metódust osztályra szoktunk hívni (**Parent.f** vagy **Child.f**), ezért warning-ot kapunk, de akkor is ugyanaz történik.

Vagyis a statikus felüldefiniálás elvész, az számít, hogy mely típus statikus metódusát hívjuk, nem pedig az hogy a konkrét példány hogyan jött létre.

## Tagváltozó

Az ugyanolyan nev? (nem statikus) tagváltozók a leszármazott osztályban **elfedik** az öröklött tagváltozót.

```
public class A
{
    protected int a_;
}

public class B extends A
{
    protected int a_; // ekkor van A.a_ és B.a_ is
    public void f()
    {
        a_        // ez B.a_
        this.a_   // ez is B.a_
        super.a_  // ez A.a_
    }
```

```
}
```

Ezt könny? elvéteni és zavaró is, ezért *ellenjavallott*!

Még könnyebb elrontani, ha van egy *ugyanolyan nev? lokális változó*nk is. Szintén ellenjavallott.

```
public class B extends A
{
    protected int a_;
    public void f(float a_)
    {
        a_        // ez a függvény argumentuma
        this.a_   // ez B.a_
        super.a_  // ez A.a_
    }
}
```

# Interface

Definiáljuk az alábbi **interface**-t

```
public interface Polygon
{
    public float area();
    public void translate(float x, float y);
}
```

Két osztály **implementál**ja ezt: **Square** és **LineSegment**. Valahogy így:

```
public class Square implements Polygon
{
    private float x_, y_, a_;
    public Square()
    {
        ...
    }
    .
    .
    .
    public float area()
    {
        ...
    }
    public void translate(float x, float y)
    {
        ...
    }
}
```

Hasonlóan a **LineSegment**-re (annak mindig 0 a területe).

Ekkor lehet?ségünk van ezeket a síkbeli objektumokat egységesen kezelni.

```
Polygon[] polygons = new Polygon[2];
polygons[0] = new Square(0,0,1);
polygons[1] = new LineSegment();

polygons[0].translate(-1,-1);
polygons[1].translate(1,1);
```

Figyeljük meg, hogy ekkor nem tudok egy **Polygon** példányt létrehozni, annélkül hogy Square vagy LineSegment ne lenne.

```
Polygon p = new Polygon(); // <- hiba!
```

Ez azért van, mert az interface-ek ú.n. *absztrakt osztály*ok, a metódusai (konstruktora is) csak ígéretek arra, hogy van olyan metódusa, de implemntálva nincsen. Vagyis a metódusai a leszármazott osztályokban vannak implementálva.

# Feladat

Örökl?déssel (és esetleg interface-el) érjük el, hogy legyen immutable és nem immutable osztályunk is a négyzetre és a szakaszra!

El?z? - Fel - Következ?