Öröklés

Térjünk vissza a kályhához, a `Diak` osztályunkhoz.

Adjunk hozzá egy `hazitKeszit()` metódust.

Az aktuális verziónk valahogy így néz ki (minden adattag `private`, minden metódus `public`):

```
public class Diak {
   private String nev;
   private int szuletesiEv;
   private double atlag;

   public Diak() {
   }
   public Diak(String nev, int szulEv, double atlag) {
      this.nev = nev;
      this.szuletesiEv = szulEv;
      this.atlag = atlag;
   }
   public void kiir() {
      System.out.println(this.nev + " (" + this.szuletesiEv + "; " + this.atl
ag + ")");
   }
   public void hazitKeszit() {
      System.out.println("Fogom a kedvenc szövegszerkesztőmet, és megírom, ma
jd elküldöm e-mailben.");
   }
   public void beker() {...}
}
```

Tegyük fel, hogy írtunk is már a segítségével egy csudijó kis nyilvántartó programot:

```
...
Diak[] iskola = new Diak[20];
iskola[0] = new Diak("Nagy Klára", 1999, 4.5);
iskola[1] = new Diak("Kiss Ferenc", 1999, 3.4);
iskola[2] = new Diak("Kovács Dzsesszika", 1995, 2.1);
...
for (Diak d : iskola) {
    d.kiir();
}
...
for (Diak d : iskola) {
    d.hazitKeszit();
}
```

Eszünkbe jut, hogy egy-két tanuló informatikát is tanul, és ahhoz, hogy tudjuk, ki fog tudni nekünk segíteni az iskola nyilvántartási rendszerének fejlesztésében (természetesen ingyen, informatika 5-ösért), tárolni akarjuk az ő kedvenc programozási nyelvüket (`String kedvencNyelv`). Ha meghívjuk a `kiir()` metódust, akkor ők a kedvenc programozási nyelvüket is ki fogják írni. ... Ezen kívül az informatikatanár megfigyelte, hogy ők nagyon sokszor elégedetlenek a Microsoft operációs rendszerével: (`public void windowstSzid()` kiír valami csúnyaságot erről az operációs rendszerről).

Készítsük el a fentieknek megfelelő `InfosDiak` osztályunkat:

```
public class InfosDiak {
   private String nev;
   private int szuletesiEv;
   private double atlag;
   private String kedvencNyelv;

   public InfosDiak() {
   }
   public InfosDiak(String nev, int szulEv, double atlag, String kedvencNyelv
) {
      this.nev = nev;
      this.szuletesiEv = szulEv;
      this.atlag = atlag;
      this.kedvencNyelv = kedvencNyelv;
   }
   public void kiir() {
      System.out.println(this.nev + " (" + this.szuletesiEv + "; " + atlag +
"; " + this.kedvencNyelv + ")");
   }
   public void windowstSzid() {
      System.out.println("Irgum-burgum, már megint lefagyott ez a ...");
   }
   public void hazitKeszit() {
      System.out.println("Fogom a kedvenc szövegszerkesztőmet, és megírom, ma
jd elküldöm e-mailben.");
   }
   public void beker() {.../* itt is változtatnunk kell */...}
}
```

Hogyan készítünk egy ilyen osztályt?

Lemásoljuk az osztályt és átírjuk. Hozzáadjuk a `windowstSzid()` metódust, a `kedvencNyelv`-et, módosítjuk a konstruktort, a `kiir()`-t.

Ez a megoldás elég problémás:

- **Többszöröződés**: Ha találunk egy hibát a `Diak`-ban, akkor azt biztosan megtaláljuk az `InfosDiak`-ban is, azaz két helyen kell javítani, ahogy a módosításokat is mindig két helyen kell elvégeznünk. **(DRY - Don't Repeat Yourself!)**
- **Külön tömb**: Nem tehetjük be az `InfosDiak` objektumainkat a mostani `iskola` tömbbe, mert nem azonos típusúak.

A megoldás: az **öröklés**.

Az *öröklés* (leszármaztatás, inheritance, subclassing) során alapul veszünk egy már létező osztályt (`Diak`), és abból képzünk újat úgy, hogy az eredetihez képest csak a változásokat rögzítjük.
Az eredeti osztályt *szülőosztálynak* (parent class, superclass), az újat *gyerekosztálynak* (subclass) nevezzük.
Az öröklést több szinten is végrehajthatjuk. Egy adott osztály szülőjét, annak szülőjét... összefoglalóan *ősosztályoknak* nevezzük. Egy adott osztály gyerekeit, gyerekeinek gyerekeit... pedig *leszármazott osztályoknak* nevezzük.
Ha az egyértelműséget nem veszélyezteti, tökéletesen jó, ha egy öröklési relációban *ősosztályról* és *leszármazottról* beszélünk.

Milyen változtatásokat tehetünk a gyerekosztályban a szülőosztályhoz képest?

1. új adattagot adhatuk hozzá.
2. új metódust adhatunk hozzá.
3. szülőben meglévő metódust felülírhatunk (felüldefiniálás, override)

Egy adott osztály szülőosztályát a fejlécébe írt `extends SzülőOsztály` kiegészítéssel jelöljük.

Javában minden osztálynak (maximum) egyetlen ősosztálya lehet.

A konstruktor nem öröklődik!

Lássuk az `InfosDiak` osztályunkat (második verzió):

```
public class InfosDiak extends Diak {
    /* a Diakban is meglévő adattagok kivehetők */
    // új adattag
    private String kedvencNyelv;

    public InfosDiak() {
    }
    public InfosDiak(String nev, int szulEv, double atlag, String kedvencNyelv
) {
        this.nev = nev; // HIBÁS
        this.szuletesiEv = szulEv; // HIBÁS
        this.atlag = atlag; // HIBÁS
        this.kedvencNyelv = kedvencNyelv;
    }
    // megváltoztatott metódus
    public void kiir() {
        System.out.println(this.nev + " (" + this.szuletesiEv + "; " + this.atl
ag + "; " + this.kedvencNyelv + ")"); // HIBÁS
    }
    // új metódus
    public void windowstSzid() {
        System.out.println("Irgum-burgum, már megint lefagyott ez a ...");
    }
    /* házitKeszit() nem kell, mert örökli a Diakból */
    public void beker() {.../* itt is változtatnunk kell */...} // HIBÁS
}
```

Van egy pár hibánk amiatt, hogy a `Diak` osztályban előre nem látó módon `private` lett minden adattag. A `private` definíciója az, hogy csak a definiáló osztályból érhető el. Az `InfosDiak` pedig nem a `Diak`, ezért onnan ezek nem érhetők el.

Változtassuk meg a `Diak` osztályunkat:

```
public class Diak {
   protected String nev;
   protected int szuletesiEv;
   protected double atlag;
   ...
}
```

Most már megjavultak a dolgok.

Ami adattagokat a szülőosztály definiál, azoknak a beállítását érdemes a szülőosztályra bízni, ezt nézzük meg a következő anyagban.