Java SE 3
Objektum Orientalt Programozas Alapjai

Szabo Daniel
daniel.szabo99@outlook.com

2021. június 5.

Kivonat

Ebben a feladatsorban az objektum orientalt programozas (OOP) alapjaival ismerkedunk meg: az osztalyokkal, objektumokkal, metodusokkal, osztalyvaltozokkal es az ezek kozotti kapcsolatokkal, lehetosegekkel. Az OOP nem csak a Java, hanem szamos mas programozasi nyelv fontos resze, ezert nagyon fontos, hogy az OOP alapfogalmait hianytalanul elsajatitsd.

Tartalomjegyzék

1. Bevezetes az Osztalyokba **3**
1.1. Osztaly deklaralasa . . . 3
1.2. Peldanyositas . . . 5
1.3. Metodusok . . . 6
1.4. Feladatok . . . 8
1.4.1. Feladat . . . 8
1.4.2. Feladat . . . 8
1.4.3. Feladat . . . 8
1.4.4. Feladat . . . 9
1.4.5. Feladat . . . 9

2. Osztalyokrol Bovebben **10**
2.1. Konstruktor es a this kulcsszo . . . 10
2.2. Feladatok . . . 12
2.2.1. Feladat . . . 12
2.2.2. Feladat . . . 12

3. Objektum tombok **13**
3.1. Objektum Tomb Feldolgozasa . . . 14
3.1.1. Bejaras, Megjelenites . . . 14
3.1.2. Extrem Ertek Kereses . . . 14
3.2. Feladatok . . . 15
3.2.1. Feladat . . . 15
3.2.2. Feladat . . . 16
3.2.3. Feladat . . . 16
3.2.4. Feladat . . . 16

1. Bevezetes az Osztalyokba

Bevezetes Az osztaly egy olyan szerkezet ami leirja egy valos targy, esemeny, koncepcio stb. jellemzoit es egyedi metodusait (fuggvenyeit). Az osztaly csak egy minta, onmaga nem fog adatot tartalmazni, hanem majd peldanyositaskor hozunk letre a deklaralt osztaly alapjan egy valtozot ami rendelkezni fog az osztalyban meghatarozott jellemzokkel es metodusokkal.

1.1. Osztaly deklaralasa

Egy osztaly hasznalatanak az elso lepese mindig az osztaly deklaralasa. Jellemzoen minden osztaly sajat .java fajlba kerul, aminek a neve megegyezik az osztaly nevevel. A lenti Kutya osztaly peldaul egy Kutya.java nevu fajlba kerulne.

Ebben az elso peldaban kutyakat szeretnenk modellezni. Minden kutyanak eltaroljuk a nevet, korat es azt, hogy be lett-e oltva. Minden valtozot kulon deklaralok megfelelo adattipussal. Mivel minden kutyanak mas lesz a neve, kora es oltottsaga, ezert nem allitok be egyiknek sem kezdoerteket. Amikor en nem allitok be manualisan kezdoerteket egy ilyen valtozonak, olyankor a tipus alaperteket fogja felvenni (lasd: Java SE 1, 1. tablazat).

```
1 class Kutya {
2     String nev;
3     int kor;
4     boolean oltott;
5 }
```

Kodreszlet 1. Pelda osztaly

Ebben a kovetkezo peldaban egy rendelesi rendszerben egy rendelest irok le. Minden rendelesnek van egy cikkszama, ami mivel betuket is tartalmazhat, String tipusu lett. Jellemzoen(de nem mindig) a vasarlok 1db-ot rendelnek egy termekbol, ezert a darabszamnak rogton 1 erteket adok.

```
1 class Rendeles {
2     String cikkszam;
3     int darabszam = 1;
4     String vasarloNeve;
5 }
```

Kodreszlet 2. Pelda osztaly

Ezeket a valtozokat osztalyvaltozoknak nevezzuk, es a tipusuk barmi lehet, akar tomb, akar egy egyedi adattipus, vagy akar egy egyedi adattipusokat tarolo tomb is.

```
1  class Tanulo {
2      String nev;
3      int szuletesiEv;
4  }
5  ...
6  class Osztaly {
7      String osztalyNev;
8      Tanulo[] nevsor;
9  }
```

Kodreszlet 3. Pelda osztaly

1.2. Peldanyositas

Miutan meghataroztunk egy uj osztalyt, peldanyositassal tudjuk azt felhasznalni. A peldanyositas tehat az a folyamat amivel egy osztaly alapjan uj valtozot (objektumot) hozunk letre. A sajat osztalybol valo valtozo letrehozasnak a szabalyai ugyan azok mint ami barmilyen mas tipusra vonatkozik.

```
1 class Termek {
2     int azonositoSzam;
3     String megnevezes;
4     int ar;
5 }
```

Kodreszlet 4. Webaruhaz termekeit leiro osztaly deklaralasa

A fent definialt Termek osztalybol fogunk most peldanyokat letrehozni, osszesen 2 darabot. Mind a harom valtozo felvett alapertekeket, de en ezeket szeretnem felulirni az igazi ertekekkel amiket hasznalni fogunk a programban. Egy objektum jellemzoit a `peldany.jellemzo` szintaxissal erem el, es a jellemzovel minden olyan muveletet vegezhetek amit akkor is vegezhetnek vele, ha nem egy osztaly valtozojakent lenne jelen, tehat akadaly nelkul tudom kiolvasni, beallitani es modositani.

```
1 Termek t1 = new Termek();
2 Termek t2 = new Termek();
3
4 t1.azonositoSzam = 2021001;
5 t2.azonositoSzam = 2021002;
6 t1.megnevezes = "Eper";
7 t2.megnevezes = "Csokolade";
8 t1.ar = 225;
9 t2.ar = 360;
```

Kodreszlet 5. Termekek valtozoinak beallitasa

```
1 int osszeg = t1.ar + t2.ar;
2 double atlag = osszeg / 2d;
3 System.out.println("Termekek␣atlagara:␣" + atlag + "Ft");
```

Kodreszlet 6. Termekek atlagaranak kiszamolasa

1.3. Metodusok

Ebben a peldaban egy szamologep osztalyt hozunk letre aminek ket metodusa van: osszead() es kivon(). Mind a ket metodus 2-2 parametert fogad be bemenetkent es egy eredmenyt ad.

```
1  class Szamologep {
2      int osszead(int a, int b) {
3          int osszeg = a + b;
4          return osszeg;
5      }
6
7      int kivon(int a, int b) {
8          return a - b;
9      }
10 }
```

Kodreszlet 7. Szamologep osztaly metodusokkal

Egy metodus deklaraciojanak reszeit/lepeseit figyeld meg:

- `int` a visszateresi tipusa a metodusnak: milyen tipusu valtozot ad eredmenyul,
- `osszead` a metodus neve, amire majd `peldany.metodus()` szintaxissal tudok hivatkozni
- `(int a, int b)` a metodus bemenetei: azoknak tipusai es helyi elnevezesei
- 3. sor: a metodus testenek elso sora, ami a metodus hivasakor fog futni.
- 4. sor: a `return` kulcsszo hatarozza meg a fuggveny eredmenyet

A villanykapcsolo peldajaban mar egy olyan osztalyt hozunk letre aminek osztalyvaltozoi es metodusai is vannak: ez a leggyakoribb, olyan osztalyokra amik csak adatot tarolnak vagy csak metodusokat deklaralnak, viszonylag ritkan van szuksegunk.

```
1  class Villanykapcsolo {
2      boolean felkapcsolva = false;
3
4      void felkapcsol() {
5          felkapcsolva = true;
6      }
7
8      void lekapcsol() {
9          felkapcsolva = false;
10     }
11 }
```

Kodreszlet 8. Villanykapcsolo osztaly

Figyeld meg ebben a peldaban azt is, hogy a metodusoknak nincs bemenete, se kimenete. Mivel nincs kimenetuk, a visszateresi tipusuk `void`-ra van allitva. Amig a nem-`void` metodusok minden esetben felvesznek valami erteket (meg akkor is, ha esetleg csak `null`-t), addig a void metodusok semmilyen erteket nem vesznek fel.

```
1 Villanykapcsolo nappali = new Villanykapcsolo();
2 int i = nappali.felkapcsol();
```

Kodreszlet 9. Ez a kod hibat okoz

```
1 Szamologep sz = new Szamologep();
2 int eredmeny = sz.osszead(10, 20);
```

Kodreszlet 10. Ez a kod nem okoz hibat

1.4. Feladatok

1.4.1. Feladat

Mi tortenik, ha

- ket azonos nevu osztaly hozok letre,
- egy osztalyban se osztalyvaltozok, se metodusok nincsenek,
- egy osztalyvaltozo neve megegyezik egy metodusban levo valtozoval,
- egy `void` metodusbol ertekkel probalok visszaterni,
- egy nem-`void` metodusbol nem terek vissza ertekkel
- egy metodusbol egy masik metodust meghivok
- egy metodus meghivja onmagat
- deklaralok egy valtozot a sajat osztalyombol de nem inicializalom, majd egy fuggvenyet megprobalom meghivni

1.4.2. Feladat

Mit csinal es mit ad eredmenyul a kovetkezo metodus?

```
1 int metodus(int a, int b, boolean c) {
2     int x;
3     if(c) {
4         x = a + b;
5     } else {
6         x = a - b;
7     }
8     return x;
9 }
```

Kodreszlet 11. Feladat

1.4.3. Feladat

Mit csinal es mit ad eredmenyul a kovetkezo metodus?

```
1 String metodus(String s, String s2) {
2     return s + s2;
3 }
```

Kodreszlet 12. Feladat

1.4.4. Feladat

Mit csinal es mit ad eredmenyul a kovetkezo metodus?

```
1 void metodus(boolean a, boolean b) {
2     if(a) {
3         return;
4     }
5
6     System.out.println(b);
7 }
```

Kodreszlet 13. Feladat

1.4.5. Feladat

Hozz letre a kovetkezo osztalyt:

Nev:

`Ora`

Jellemzok:

- `ora` (egesz szam)
- `perc` (egesz szam)

Metodusok:

- `printIdo` Kiirja a konzolra az ora jelenlegi idejet a kovetkezo formatumban: `Az ido 17`ora 23 perc.. Eredmenyt nem ad es bemenete sincsen.
- `beallit` Bevesz parameterkent ket egesz szamot es beallitja oket a jelenlegi idonek. Eredmenyt nem ad.
- `ora` Ez a metodus visszaadja az ora valtozo jelenlegi erteket, parametere nincsen.

2. Osztalyokrol Bovebben

Bevezetes Az elozo fejezetben az osztalyokkal kapcsolatos alapveto szabalyokat es koncepciokat fektettuk le, osztalyvaltozok es metodusok deklaralasat, hasznalatat. Ebben a fejezetben ezeket a koncepciokat bovitjuk ki uj, opcionalis lehetosegekkel amivel jobb, komplexebb adatmodelleket hozhatunk letre.

2.1. Konstruktor es a this kulcsszo

Eddig amikor egy peldanyt hoztunk letre egy valtozobol, a kovetkezo keppen tettuk:

```
1 Kutya kutya1 = new Kutya();
2 kutya1.nev = "Buksi";
3 kutya1.kor = 5;
4 kutya1.oltott = true;
```

Kodreszlet 14. Peldanyositas es osztalyvaltozok beallitasa

Szepen olvashato es ertheto a kod, viszont lathatoan sokat kell irnunk ahhoz, hogy ezt a 3 rovid adatot elmentsuk a peldanyunkban, ezert jo lenne, ha rovidebben ezt le tudnank irni. Az is jo lenne, hogy ha megszabok egy jellemzot egy osztalyomban, tudjam szabalyozni, hogy azt a jellemzot kotelezoen minden peldanynak be kelljen allitani. Ezzel elkerulhetnem, hogy veletlenul valamikor kor vagy nev nelkul hozzak letre egy kutya peldanyt. Ezekre es szamos mas problemara is a konstruktor a megoldas. A konstruktor az egy olyan metodus ami a peldanyositaskor hivodik meg. A kovetkezo peldaban meghatarozok egy konstruktort a Kutya osztalyomban, majd az uj konstruktorom segitsegevel ujra letrehozom ugyanazt a peldanyt amit az elozo kodreszletben.

```
1  class Kutya {
2      String nev;
3      int kor;
4      boolean oltott;
5
6      Kutya (String nev, int kor, boolean oltott) {
7          this.nev = nev;
8          this.kor = kor;
9          this.oltott = oltott;
10     }
11 }
12 ...
13 Kutya kutya1 = new Kutya("Buksi", 5, true);
```

Kodreszlet 15. Konstruktor pelda

A fenti peldaban figyeld meg a konstruktor felepiteset. Mint lathatod, nincs visszateresi erteke, csak neve, ami pedig pontosan megegyezik az osztaly nevevel. Ugyanugy vannak parameterei mint barmilyen masik metodusnak, ebben az

esetben minden osztalyvaltozohoz egy darab. Mivel az osztalyvaltozok es a konstruktor metodus parameterei ugyanazokat az elnevezeseket kaptak, igy a `this` kulcsszoval tudom megkulomboztetni a kettot. Tehat a `this`.valtozo vagy `this.metodus` mindig a peldany sajat osztalyvaltozojara vagy metodusara mutat. Ez a kulcsszo a legtobb esetben opcionalis, de a fenti konstruktorban kotelezo, mert kulonben nem tudna Java megmondani, hogy az ertekadas baloldalan az osztalyvaltozora gondoltunk, es nem a parameterkent kapott valtozora.

A kovetkezo peldaban olyan konstruktort hozunk letre a `Jarmu` osztalyhoz, ami a `tulajdonosNeve` valtozot egy parameter helyett mindig arra allitja be, hogy `"Nincs␣Tulajdonos"`, illetve minden peldanyositaskor a konzolon ertesiti a felhasznalot, hogy egy uj jarmu jott letre.

```
1  class Jarmu {
2      String rendszam;
3      String gyarto;
4      String modell;
5      String tulajdonosNeve;
6
7      Jarmu (String rendszam, String gyarto, String modell) {
8          this.rendszam = rendszam;
9          this.gyarto = gyarto;
10         this.modell = modell;
11         this.tulajdonosNeve = "Nincs␣Tulajdonos";
12         System.out.println("Egy␣uj␣autot␣hoztunk␣letre!");
13     }
14 }
```

Kodreszlet 16. Konstruktor pelda

2.2. Feladatok

2.2.1. Feladat

Adj hozza egy teljes konstruktort a kovetkezo osztalyhoz

```
1 class Tanulo {
2     int azonosito;
3     String nev;
4     String osztaly;
5
6     ...
7 }
```

Kodreszlet 17. Feladat

2.2.2. Feladat

Adj hozza egy olyan konstruktort a kovetkezo osztalyhoz, ami 4db `String` parametert ker be es azok alapjan allitja be a `szamok` valtozot.

```
1 class Album {
2     String[] szamok;
3
4     ...
5 }
```

Kodreszlet 18. Feladat

3. Objektum tombok

Egy objektum tobb kulonbozo erteket tarol amik egy bizonyos valamit jellemeznek, tehat ugy viselkeldik mint egy tablazatban egy sor. Ha pedig sok objektumot veszunk (ugyanabbol az osztalybol) akkor pedig egy tablazathoz hasonlithato strukturat kapunk. Sajat osztalyokkal valo tombok keszitesere szinten ugyanazok a szabalyok vonatkoznak, mint barmilyen mas adattipusra.

Ebben a peldaban egy 10 elembol allo `Kutya` tombot hozunk letre. Ha a tombot a lenti modon inicializaljuk, `null` ertekekkel lesz feltoltve.

```
1 Kutya[] kutyak = new Kutya[10];
```

Kodreszlet 19. Kutya tomb keszitese - 1. modszer

Ha szeretnenk rogton ertekekkel feltolteni egy objektum tombot, megtehetjuk, hogy felsoroljuk az ertekeket.

```
1 Kutya kutya1 = new Kutya("Buksi", 5, true);
2 Kutya kutya2 = new Kutya("Folti", 2, true);
3 Kutya kutya3 = new Kutya("Tapi", 9, false);
4
5 Kutya[] kutyak = {kutya1, kutya2, kutya3};
```

Kodreszlet 20. Kutya tomb keszitese - 2. modszer

Ha az ertekek meg nem leteznek, inicializalhatjuk oket a felsorolason belul is.

```
1 Kutya[] kutyak = {new Kutya("Buksi", 5, true),
2                   new Kutya("Folti", 2, true),
3                   new Kutya("Tapi", 9, false)};
```

Kodreszlet 21. Kutya tomb keszitese - 3. modszer

3.1. Objektum Tomb Feldolgozasa

Miutan letrehoztunk egy objektumokat tartalmazo tombot, azt az eddig tanult modszerekkel kezelhetjuk, peldaul `for` ciklusokkal bejarhatjuk es modosithatjuk oket vagy statisztikakat szamolhatunk beloluk. Tetszes szerint valaszthatunk `for` es `for-each` ciklusok kozott, de a legtobb esetben nem vagyunk kivancsiak az elemek indexeire es nem terunk el az alap bejarasi modszertol, igy a `for-each` jobb valasztas lesz.

```
1 Kutya[] kutyak = {new Kutya("Buksi", 5, true),
2                   new Kutya("Folti", 2, true),
3                   new Kutya("Tapi", 9, false)};
```

Kodreszlet 22. A kutya tomb amivel dolgozni fogunk a kovetkezo peldakban

3.1.1. Bejaras, Megjelenites

```
1 System.out.println("nev\tkor\toltott");
2 for(Kutya k : kutyak) {
3     System.out.println(k.nev + "\t" + k.kor + "\t" + k.oltott);
4 }
```

Kodreszlet 23. Kutyak adatainak megjelenitese

3.1.2. Extrem Ertek Kereses

```
1  Kutya legidosebb = kutyak[0];
2
3  for(Kutya k : kutyak) {
4      if(k.kor > legidosebb) {
5          legidosebb = k;
6      }
7  }
8
9  System.out.println("A␣legidosebb␣kutya␣" + legidosebb.nev
10             + "␣aki␣" + legidosebb.kor + "␣eves." );
```

Kodreszlet 24. Legidosebb kutya megkeresese

3.2. Feladatok

A lenti feladatokban a kovetkezo osztallyal fogunk dolgozni:

```
1  class Termek {
2      int azonositoSzam;
3      String megnevezes;
4      int ar;
5      String[] kategoriak;
6
7      Termek(int azonositoSzam, String megnevezes,
8                  int ar, String[] kategoriak) {
9          this.azonositoSzam = azonositoSzam;
10         this.megnevezes = megnevezes;
11         this.ar = ar;
12         this.kategoriak = kategoriak;
13     }
14
15 }
```

Kodreszlet 25. Feladat - Termek osztaly

3.2.1. Feladat

A `Termek` osztalybol keszits peldanyokat a kovetkezo adatokkal es helyezd el oket egy `termekek` nevu tombben.

Azonosito	**Megnevezes**	**Ar**	**Kategoriak**
10001	Alma	20	elelmiszer, egeszseg
10002	Chips	32	elelmiszer
10003	C-Vitamin	19	egeszseg
10004	Elektromos Fogkefe	262	elektronika, egeszeg
10005	Telefon Tolto	220	elektronika

1. táblázat: Termekek

3.2.2. Feladat

Egy ciklus segitsegevel ird ki az osszes termek adatait a kovetkezo formaban:

```
Azonosito   Megnevezes          Ar      Kategoriak
10001       Alma                20 Ft   elelmiszer, egeszseg
10002       Chips               32 Ft   elelmiszer
10003       C-Vitamin           19 Ft   egeszseg
10004       Elektromos Fogkefe  262 Ft  elektronika, egeszseg
10005       Telefon Tolto       220 Ft  elektronika
```

Kodreszlet 26. Feladat

Egy ciklus segitsegevel ird ki az osszes egeszeg kategoriaban levo termek adatait a kovetkezo formaban:

```
Egeszseg kategoria termekei:
Azonosito   Megnevezes          Ar
10001       Alma                20 Ft
10003       C-Vitamin           19 Ft
10004       Elektromos Fogkefe  262 Ft
```

Kodreszlet 27. Feladat

3.2.3. Feladat

Egy ciklus segitsegevel szamold ki a termekek atlagarat es ird ki a konzolra a kovetkezo formaban:

```
A termekek atlagara 110.6 Ft.
```

Kodreszlet 28. Feladat

3.2.4. Feladat

Egy ciklus segitsegevel keresd meg es jelenitsd meg a legdragabb termek adatait

```
A legdragabb termek:
Azonosito   Megnevezes          Ar      Kategoriak
10004       Elektromos Fogkefe  262     elektronika, egeszseg
```

Kodreszlet 29. Feladat