SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

Java programozás

1. Java alapok

2. Metódusok, öröklődés

NÉMETH RICHÁRD

Egyetemi tanársegéd
Széchenyi István Egyetem, Informatika Tanszék
Olvasószerkesztő
Jog-Állam-Politika (MTA-"A" kat.) | Universitas
Hallgatói kapcsolattartó, szakmentor
Nemzetközi kapcsolatok; Gazdaságinformatikus szak

Iroda: A-602
E-mail: nemeth.richard@ga.sze.hu
Konzultációk: online, ill. személyesen (előtte 2 nappal jelezni e-mailben)

TANANYAG

ELEKTRONIKUS JEGYZET
www.sze.hu/~varjasin/publikaciok/Programozas_III.pdf

Diák: Moodle rendszerben

Interaktív, alapos online tananyag, gyakorlati példákkal:

Introduction to Java
https://www.sololearn.com/learning/1068 (reg szükséges)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

JAVA FŐ SPECIFIKÁCIÓI (SZEMPONTOK)

- Egyszerű (C++ egyszerűsítve, de kibővített lehetőségekkel)
- Tisztán objektumorientált
- Elosztott (hálózatos környezetben – Internet erőforrások)
- Robusztus (hibatűrő – hibák már fordítási időben)
- Biztonságos (hozzáférések, jogosultságok → egységbezárás)
- Hordozható (class fájl bármilyen virtuális gépen)
- Architektúra független (fut bármilyen hw/sw architektúrán)
- Interpretált (kódlétrehozás futásidőben)
- Nagyteljesítményű (részben igaz)
- Többszálú (párhuzamos programozás/algoritmusok)
- Dinamikus (osztálykönyvtárak bővíthetők, fejleszthetők)

ALAPTÍPUSOK, REFERENCIATÍPUSOK

byte	**8 bites előjeles egész**	**byte largestByte = 127; * 2^8**
short	16 bites előjeles egész	short largestShort = 32767; 2^15
int	32 bites előjeles egész	int largestInteger = 2147483647; 2^31
long	64 bites előjeles egész	long largestLong = 92233720368547758O7L;
float	32 bites egyszeres lebegőpontosságú	float largestFloat = 3.4028235E38f; 10^38
double	64 bites kétszeres lebegőpontosságú	float largestFloat = 1.7976931348623157E308; 10 ^308
char	16 bites Unicode-karakter	char aChar = 'S';
boolean	logikai érték (igaz / hamis)	boolean abálóénak = true;

- A referenciatípusok objektumok
- A legtöbb primitívnek van referencia párja, pl.:
  int – Integer, char – String, boolean – Boolean

Különbség: ref → metódusai vannak (objektum!)

Referencia típus: osztály, tömb, enum, interface

FEJLESZTŐKÖRNYEZETEK

- IntelliJ Idea (Community Edition)
  (főleg Androidra, fejlett kódkiegészítés, magas gépigény)

- Oracle JDeveloper
  (Oracle saját környezete, főleg ipari felhasználás)

- Eclipse
  (gépigény jó, kiváló memóriakezelés)

- NetBeans
  (Sun saját környezete volt
  → Apache égisze alatt)
- stb.

NÉHÁNY BILLENTYŰPARANCS A KEZDETEKHEZ

Automatikus kiegészítés

Szövegkiíratás példa sout + CTRL + SPACE (utána: ENTER vagy TAB)

Adott sor törlése

Adott sor duplikálása

Kód formázása

Kódrészlet beszúrása

Blokk kommentezés

Build futtatása

JAVA FÁJL FELÉPÍTÉSE

- /* blablabla */ - fejlesztői információk, törölhető
- Kommentek fajtái:
  - Blokk komment: /* többsoros szöveg */
  - Sor komment: // megjegyzés, adott sorban
  - Javadoc leíró: /** dokumentációs kommentek */ → html

- package csomagnev; - csomag megadása (később)
- public class app_neve { … }
  - Hozzáférés (public, protected… később)
  - Class előtag – osztály jelzése, név == java fájl neve (elírva nem ok)

- Belépési pont (csak main-nál): void – C-ből ismerős lehet
  - public static void main(String[] args {…..} (args: paraméterek)

- Utasítások
  - Itt: szöveg kiíratása (System.out.println("Hello 2023!");)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

SZÖVEG KIÍRATÁSA KONZOLRA

```
System.out.println("Adattípusok kiíratása");
System.out.println("Teszek egy üres sort\n");
System.out.println("x értéke: " + x);
System.out.println("d értéke: " + d);
System.out.println("\n");
System.out.println("Megnövelem x értékét");
x+=3;
System.out.println("Az x most 7 + 3 lesz, azaz: " + x);
```

- Testvére (hibaüzenetekhez) :
  System.err.println(”Hibaüzenet”);

- Adattípusok maximális értékének kiíratása – próbálgassuk!

```
System.out.println("Integer adattípus maximális értéke: " + Integer.MAX_VALUE);
```

HASONLÍTÁS; ÉS-VAGY; „FIXEK”

- Feltételek kiértékelése páronként
- Szintaxis: IF… ELSE IF... ELSE

```
int kor = 25;
if (kor < 30) {
        System.out.println("fiatal");
}

if (kor == 25) {
        System.out.println("huszonöt");
}
```

ÉS – VAGY (&& - ||)

```
if (szam % 3 == 0 || szam % 7 == 0) {    // VAGY
        System.out.println(szam + ": 3-mal vagy 7-tel osztható");
}
```

PREFIX, POSTFIX, INFIX: Az operátor és az operandus viszonya

(Prefix: az operátor az operandus előtt áll, post: után, in: közte)

A+B +AB AB+

HASONLÍTÁS FELADATOK

PÉLDA
Írassuk ki konzolra, hogy a korábban megadott tetszőleges kétjegyű szám páros vagy páratlan-e!

FELADAT
Írassuk ki, hogy az alábbi számok maradék nélkül oszthatóak-e 2-vel, 5-tel, vagy 2-vel és 5-tel is: 9, 12, 20, 7, 15
Kimenet az alábbi mintára:

```
10 osztható 2-vel és 5-tel is
 4 osztható 2-vel, de 5-tel nem
25 osztható 5-tel, de 2-vel nem
19 sem 2-vel, sem 5-tel nem osztható
```

NÖVEKMÉNYES (LÉPTETŐ) CIKLUS: FOR

- Művelet(ek) ismételt végrehajtása megadott lépésszámmal
- Szintaxis: FOR (KEZDŐÉRTÉK;BEF. ÉRTÉK;LÉPÉSSZÁM)

```
System.out.println("Írjuk ki a számokat egyesével növelve -7-től 10-ig egy sorban");

for (int i = - 7; i <= 10; i++) {

    System.out.print(i + "\t");
}
```

VÉGTELEN CIKLUS FOR-RAL

```
for(;;) {
    System.out.println("a");
}
```

FELADAT
Írassuk ki konzolra a páros számokat -10-től 100-ig,
ha a szám osztható 7-tel!

SZÖVEGKEZELÉS ALAPJAI: A STRING

- Karakterek (char) tömbjeként kezelendő
- Objektum → példányosítható a new utasítással.
  Pl. String s = new String(„abc”);
- Szérializálható (állapota elmenthető, byte stream konv.)
- Sorbarendezhető (Collections.sort(s); Comparable (később))
- Hasonlítása mindig equals metódussal (nem == infix)
- Mivel objektum, vannak metódusai

STRING METÓDUSOK I.

```
String s = „alma”; // nem kötelező a „new” hívása!
String s2 = new String(„Körte”); // de lehet úgy is

System.out.println("Szöveg: " + s);

System.out.println("Hossza: " + s.length());

System.out.println("2. karaktere: " + s.charAt(2));

System.out.println(„Az ’m’ betű első előfordulása: " + s.indexOf('m'));
System.out.println(„A ’h’ betű első előfordulása: " + s.indexOf('h')); // -1

System.out.println("egyezik-e 'alma' szóval: " + s.equals("alma"));
System.out.println("egyezik-e ‚Alma' szóval: " + s.equals(„Alma"));
```

STRING METÓDUSOK II.

```
System.out.println("üres-e: " + s.isEmpty());     //üres: "" , de nem a null

System.out.println("'ma'-ra végződik-e: " + s.endsWith("ma")); //startsWith

System.out.println("Rész kivágása, pl.: " + s.substring(1, 4));

System.out.println("Az 'a' cseréje '?' jelre: " + s.replace('a', '?'));

System.out.println("Nagybetűs alak lekérése: " + s.toUpperCase());
// maga a String változik ettől? Hogyan változik? – s = s.toUpperCase

System.out.println("Kisbetűs alak lekérése: " + s.toLowerCase());

System.out.println("szerepelt-e az \"al\" szövegrész: " + s.contains("al"));
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

KITÉRŐ: SHORTIF

```
1. System.out.println("szerepelt-e \"al\" szövegrész: " + s.contains("al"));
                                                    // kimenet: true/false
2. String valasz = "";
     if (s.contains("al")) {
         valasz = "Igen";
     } else {
         valasz = „Nem";
     System.out.println("Szerepelt-e? „ + valasz);          //      kimenet:
Igen/Nem

3. String valasz = (s.contains("al") ? "Igen„ : "Nem");
    System.out.println("Szerepelt-e al?” + valasz);   //csak eldöntendőnél!

4. System.out.println("Szerepelt-e al? " + (s.contains("al") ? "Igen" : "Nem"));
```

STRING METÓDUSOK III.

```
//Elé szóköz, Egyesítve,  közé szóköz, utána szóköz, utána egyben");
String egyben = " " + s + " " + s2 + " egyben";
System.out.println("Egyben:" + egyben);

System.out.println("Távolítsuk el a fehér karaktereket!");
egyben = egyben.trim();
System.out.println("Egyben: " + egyben);          // Honnét törli csak?

String[] darabok = egyben.split(" ");             //String-ek tömbje
System.out.println(Arrays.toString(darabok));

System.out.println(Arrays.toString("192.168.1.102".split("\\.")));
```

Vannak kivételek: . + ? …

STRING METÓDUSOK IV.

KARAKTER-MŰVELETEK

```
if (Character.isLetter('k')) {
        System.out.println("Betű!");
    }
if (Character.isLowerCase('k')) {                    // isUpperCase('k');
        System.out.println("Kisbetű!");
    }
if (Character.isDigit('1')) {
        System.out.println("Szám!");
    }
if (Character.isLetterOrDigit('k')) {
        System.out.println("Betű vagy szám!");
    }
if (Character.isWhitespace(' ')) {
        System.out.println("Fehér karakter");
    }
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

PÉLDA: STRING MŰVELETEK

Vegyünk fel egy szöveget: „Holnap csütörtök lesz!"

a) hány "ö" betű volt?
b) utolsó karakter '?' volt-e
c) cseréljük le az ‚l' betűket '+' jelre!

Az egyes feladatok eredményét jelenítse meg a kimeneten!

FELADAT: KARAKTER- ÉS STRING MŰVELETEK

Írjon jelszóellenőrzőt! Szabályok:

A jelszó legalább 8 karakter hosszú legyen!
A jelszóban lennie kell:
a) legalább 1 betűnek,
b) legalább 1 számjegynek,
c) legalább 1 egyéb írásjelnek (bármi, ami nem betű v. szám).

A jelszó hibás, ha egymás mellett két azonos nagybetű áll.

A kész programot próbálja ki többféle (jó-rossz) kiinduló adattal.

FELADAT LEHETSÉGES KIMENETE:

```
AA!1a12345
Hiba! Két nagybetű egymás mellett: AA
```

```
A!12345
Hiba! Legalább 8  karakter legyen!
```

```
A1234As5
Hiba! Szerepeljen betű, szám és írásjel is!
```

```
A!1234As5
Jó a jelszó!
```

ALAPSZINTŰ LISTASZERKEZET: ARRAYLIST

- Azonos típusú elemek listába foglalása
- Testvérei: Vector, LinkedList
- Létrehozás: ArrayList<Típus> neve = new ArrayList<>();

ALAPVETŐ MŰVELETEK

- Hozzáadás → lista_neve.add(elem);
- Bejárás/ForEach: → for (Típus egyed: lista_neve) {… }
- Beszúrás adott helyre → lista_neve.add(index, elem);
- Eltávolítás → lista_neve.remove(index);
- Lekérdezés → lista_neve.get(index);
- Méret lekérdezése: lista_neve.size();
- Lista rendezése ABC szerint: Collections.sort(lista_neve);

ARRAYLIST FELADAT

Hozzunk létre egy legalább 3, legfeljebb 10 elemű, pozitív kétjegyű számokból ArrayListet szamok néven!

- Töröljük a 3. elemet!
- A 2. helyre szúrjuk be a 88-at!
- Írassuk ki a lista elemeit egymás mellé, TAB-bal elválasztva
- Rendezzük növekvő sorrendbe!
- Írassuk ki az első és az utolsó elemet!
- Ismét írassuk ki a teljes listát TAB-bal elválasztva!

MŰVELETEK ARRAYLIST-TEL

- Véletlen szám létrehozása: Math.random(); függvény
- Szintaxisa: x = (int)(Math.random()*100)); //*100?

- Lottószámok generálása listába:

```
ArrayList<Integer> lottoSzamok = new ArrayList<>();
for (int i=0;i<5;i++) {
    lottoSzamok.add((int) (Math.random() * 100));
}
```

hol a hiba?

- Átlagszámítás

```
int osszeg = 0;
for (int i=0;i<lottoSzamok.size();i++) {
    osszeg+= lottoSzamok.get(i);
}

System.out.println("A lottószámok átlaga: " + osszeg/lottoSzamok.size());
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

MŰVELETEK ARRAYLIST-TEL – FELADAT

Esztike gyümölcsöt árul az oviban. Az udvaron az almafáról leszedett 1 darab almát. Hozott otthonról egy banánt és egy kis darab dinnyét. Kevesli: szed még 4 darab almát. Minden gyümölcsöt 50 Ft-ért árul. A következő események mindegyike után írassuk ki a konzolra, hogy épp milyen gyümölcsei vannak!

a) Pannikával elcseréli a banánt egy eperre.
b) Pistike megkérdezi, van-e nála dinnye. Mit válaszol?
   (A kisfiú mégsem veszi meg a dinnyét.)
a) A dadus minden almát megvesz tőle.
b) Észreveszi, hogy az eper már nem ehető. Kidobja.
c) Befejezi az árusítást. Mennyit keresett, és mi maradt neki?

FELADAT HELYES KIMENETE

```
alma    dinnye  banán   alma    alma    alma    alma

Miután elcseréltük a banánt eperre:
alma    dinnye  eper    alma    alma    alma    alma

Igen, Pistike, van dinnye is!

A dadus megvesz minden almát (mindet töröljük)
Ami marad:
dinnye  eper

Kidobjuk az epret.
Ami marad: dinnye
Összesen ennyi pénzt keresett Esztike: 250 Ft.
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Érték cseréje ArrayList-ben: set utasítás.
- Itt:

  szavak.set(cserelendo_indexe, „Új szöveg”);

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OOP, OSZTÁLY, OBJEKTUM

- Objektumorientált program: Egymással kommunikáló objektumok összessége, melyben minden egyes objektumnak jól meghatározott feladatköre van.
- Osztály: ugyanolyan tulajdonságokkal rendelkező objektumok magasabb szintű egysége – példa: hallgatók
- Objektum: valamilyen osztály konkrét, egyedi példánya (tulajdonságokkal (adatok), műveletekkel (metódusok))
- Példa: autó osztály
  - Osztálytulajdonságok – kivitel, ajtók száma, szín
  - Műveletek – lekérdező (getSzin();), saját (megtankol();)
  - Mi az, hogy konkrét példány? (új kocsit veszek)

Egy osztályt 1 adott tevékenységre használunk (név eszerint!)

ÜZENETKÜLDÉS

- OPP lényege: Objektumok üzeneteket küldenek és fogadnak

- Hozzáférés, jogosultság szintje fontos!
  Pl. foci vébén: labdát kicserélem medicinlabdára, stb.
  → bughoz vezet

- Program egy osztályának hívom meg a metódusát
  Pl. autónál a kéziféket a konkrét piros Astrán húzom be
  Ha nem lenne private: más behúzza a fékemet autópályán

Üzenet/metódus: objektumok közötti adatcsere, kommunikáció

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OBJEKTUMORIENTÁLT ALAPELVEK

- Osztály és objektum (ld. korábbi diák)
- Egységbezárás: Az egy osztályban definiált adattagok és a hozzájuk tartozó metódusok összetartoznak és egy egységként kell kezelni őket.
- Adatrejtés: Minden objektum védi a belső adatait, így adatstruktúrája kívülről nem érhető el. Ezekhez az adatokhoz csak a saját metódusai férhetnek hozzá
- Öröklődés: Az osztályok leszármaztatásával a leszármazott örökli ősének minden egyes tulajdonságát, műveletét. A leszármazott osztály új tulajdonságokkal és műveletekkel bővíthető.
- Többalakúság: Egy változó élettartama alatt (futás közben) más osztálybeli értéket is felvehet az öröklődési láncban.

ALAPELVEK A GYAKORLATBAN

- Osztályok üzengetnek egymásnak. De!

- Mindig van egy vezérlő objektum
  → ő kezdeményezi a cselekvést → Main fv.

- Az objektumok teszik, amire utasítást kapnak.

- Példa: reggelire tojásrántotta.
- Ki a vezérlő? Mik az attribútumok? Mik a metódusok?

- Mik kerülnek egy osztályba?

VÁLTOZÓK

A változó olyan adatelem, amely azonosítóval van ellátva.
Java-ban kötelező megadni a típusát (deklarálni).
Egyből kezdőértéket is kaphat (metódusban ez kötelező)

Java-ban háromféle változót különböztetünk meg:

- Lokális változók
  Pl. int i = 7; String nev = ”Béla”;
- Példány változók
  Pl. KocsiEgy.szin (entitástól függetlenül pl. zöld)
- Osztály változók
  Osztályhoz tartozik → Bármelyik objektumból elérem
  Csak egy példányt osztunk meg. Használat: main-en kívül
  Pl. public final static String KONSTANS = "Konstans";

A MAIN FÜGGVÉNY

- A Java-ban írt programok belépési pontja.
- Minden alkalmazásnak tartalmaznia kell a következőképpen:

```
public static void main(String[] args) {..}
```

ahol:

  - public: jelzi, hogy a metódust más osztálybeli objektumokból is meg lehet hívni
  - static: jelzi, hogy a Main osztálymetódus
  - void: a metódusnak nincs visszatérési értéke

- A String[] args részben paramétereket adhatunk meg parancssori futtatáshoz: <Osztály><p1><p2>…
- Az alkalmazás futtatható parancssorból a javac fordítóval: javac Programneve.java param1, param2…

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ÚJ OSZTÁLY LÉTREHOZÁSA

- Package-ben: New ► Java Class
  (Main-nel rendelkező fő osztályom már van!)

- Minden osztálynév nagybetűvel kezdődik
  (DE! adattag, metódus kezdése kisbetűvel!)

- Pl. kiíratásnál a System.out.println-nél
  a System osztálynak van out objektuma, üzenetet kap

- ! Mindig objektumnak üzenek!

- Elnevezés → magyar vagy angol? Kétféle szemlélet

EGYSÉGBEZÁRÁS (ENCAPSULATION)

- Egységbezárás – OOP egyik alapelve:

Az egy osztályban definiált adattagok és a hozzájuk tartozó metódusok összetartoznak és egy egységként kell kezelni őket.

```
public class Auto {
    private String szin;
    private String kivitel;
    private int ajtokSzama;
}
```

- Adattag = attribútum

- Lényege: az adatok és műveletek „egy helyen” vannak.
- Attribútum: az obj milyen tulajdonságokkal ruházható fel.
- Ezek az obj példány változói (új példány = új foglalás)
- A tulajdonságot private-tal adom meg
  (csak hívással lehessen lekérdezni, módosítani)

KONSTRUKTOR

- Konstruktor: Az osztály belépési pontja, ami az osztály létrehozásakor automatikusan meghívódik.

- Létrehozhatom kézzel, de bill.paranccsal beszúrható:
  Alt + Insert ► Constructor (Source ► Insert Code)

- Mindig az osztályon belül hozom létre!

private String szin; (…)

```
public Auto(String szin, String kivitel, int ajtokSzama) {
    this.szin = szin;
    this.kivitel = kivitel;
    this.ajtokSzama = ajtokSzama;
}
```

- A konstruktor megadása a példányosítás. Csak 1x hívható.
- Konstruktor neve ugyanaz, mint az osztályé
- Nincs visszatérési értéke

PÉLDÁNYOSÍTÁS

- Példányosítás során létrehozzuk az egyes osztályok konkrét példányait. Példányosítás után minden objektum tudja, melyik osztályhoz tartozik.

- Szintaktika: new utasítással hozom létre Main-ben

```
Auto KocsiEgy = new Auto("piros","SUV", 5);
Auto KocsiKetto = new Auto("kék","szedán", 4);
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

GETTER-SETTER SZEREPE

- Minden attribútumhoz célszerű létrehozni adat kezelésére szolgáló metódusokat (akkor is, ha nem használjuk). Miért?

- Getter: visszaadja az adott attribútum aktuális értékét, jellemzően getAttribútum();
  Típusa az attribútum típusának felel meg

- Setter: megváltoztathatjuk az attribútum értékét, jellemzően setAttribútum(); - mindig void típusú (nincs return)

- Alt + Ins lenyomása után hozzáadhatjuk (választható)

Egy metódus akkor helyes, ha csak egyvalamiért felel!
(nem véletlen, hogy a metódusok max 1 értéket adnak vissza!)

GETTER-SETTER SZINTAKTIKA

- Attribútumhoz getter/setter:

```
public String getSzin() {
    return szin;
}

public void setSzin(String szin) {
    this.szin = szin;
}
```

- Meghívása (lekérdezés ill. új érték adása) main-ben:

```
System.out.println("KocsiKettő színe: " + KocsiKetto.getSzin());
System.out.println("Állítsuk be KocsiEgy ajtajainak számát 2-re!");
KocsiEgy.setAjtokSzama(2);
```

TOSTRING METÓDUS

- Egy adott objektum állapotának kiíratására szolgál
- Szintaktikája: public (kintről elérhető) String toString() {…}
- Alt + Insert-tel beszúrható

```
@Override
public String toString() {
    return "Auto{" + "szin=" + szin + ", kivitel=" + kivitel + ", ajtokSzama=" + ajtokSzama + '}';
}
```

@Override: felülirás. @-cal az annotációkat jelöljük:
Ezek plusz információk a compilernek és a JVM-nek
(van hogy kötelező, pl. adatbázisnál az @Autowired)

A fenti példa az alapértelmezett.
Tetszés szerint testre szabható, de default nem (Eclipse: igen)

Mit ír ki a „sout”, ha nincs megadva toString?

PÉLDÁNYOSÍTÁS FELADAT

Hozzunk létre egy Könyv osztályt, aminek adattagjai:

- konyvSzerzoje (String)
- konyvCime (String)
- kiadasEve (int)
- kiado (String)
- isEbook (boolean)

Hozzunk létre belőle 3 önálló példányt!

TOSTRING, GETTER-SETTER FELADAT

08. Feladat:
Az előző Könyv osztályunkból kiindulva (adattagok most is: konyvSzerzoje, konyvCime, kiadasEve, isEbook)
Vásároljunk 5 db könyvet, és írassuk ki az adataikat toString-gel az alábbi példának megfelelően:

A(z) 1. könyv szerzője: Frank Herbert, címe: Dűne, kiadás éve: 1965, nem ebook.
A(z) 2. könyv szerzője…

a) A 3. és az 5. könyvről kiderül, hogy az eddigi adatokkal ellentétben Magánkiadás! Lehet ua. a kiadó?!

b) Minden könyvet digitalizálunk (Ebook lesz belőle)

FELADAT: ARRAYLIST OSZTÁLYON BELÜL

Hófehérke céget alapít, és nyilvántartást készít a dolgozókról. Mindenkiről tárolja a nevét, a korát, a beosztását és a havi fizetését. Az adatok a következők:
Hófehérke, 20, ügyvezető, 1200 arany; Tudor, 60, mérnök, 750 arany; Morgó, 65, műszakvezető, 600; Vidor (53), Szundi (55), Szende (42) és Hapci (50), bányászok, 400-400 arany. Kukát (30) nem vették be a brancsba, mert mindig baj van vele.
Készítse el a „Dolgozó” osztályt, és példányosítsa a fenti egyedeket!
Az első, 20 munkanapos hónapban a következő események történtek:

a) Szundi harmadszor aludt el, ezért levontak tőle háromnapi fizetést.
b) Kuka kap egy esélyt – felvették bányásznak, 500 arany fizetésért.
c) A bányászok fellázadtak, amiért Kuka többet keres, ezért 1 hétig nem dolgoztak. A fizetésükből levonták az 5 munkanapot, de kaptak 200 arany fizetésemelést. (Szundi nem mert lázadni, ő végigdolgozta azt a hetet is – neki is emeltek).
d) Kuka megsértődött, amiért neki nem jár emelés, és felmondott. 1 heti fizetés jár neki.
e) Óriási megrendelést kaptak a királyfitól, ezért minden (még állományban lévő) dolgozó 100 arany bónuszt kapott, Hófehérke 300-at, mert ő a főnök.

A konzolon jelenítse meg, hogy hónap végén melyik karakter mennyi fizetést kapott!

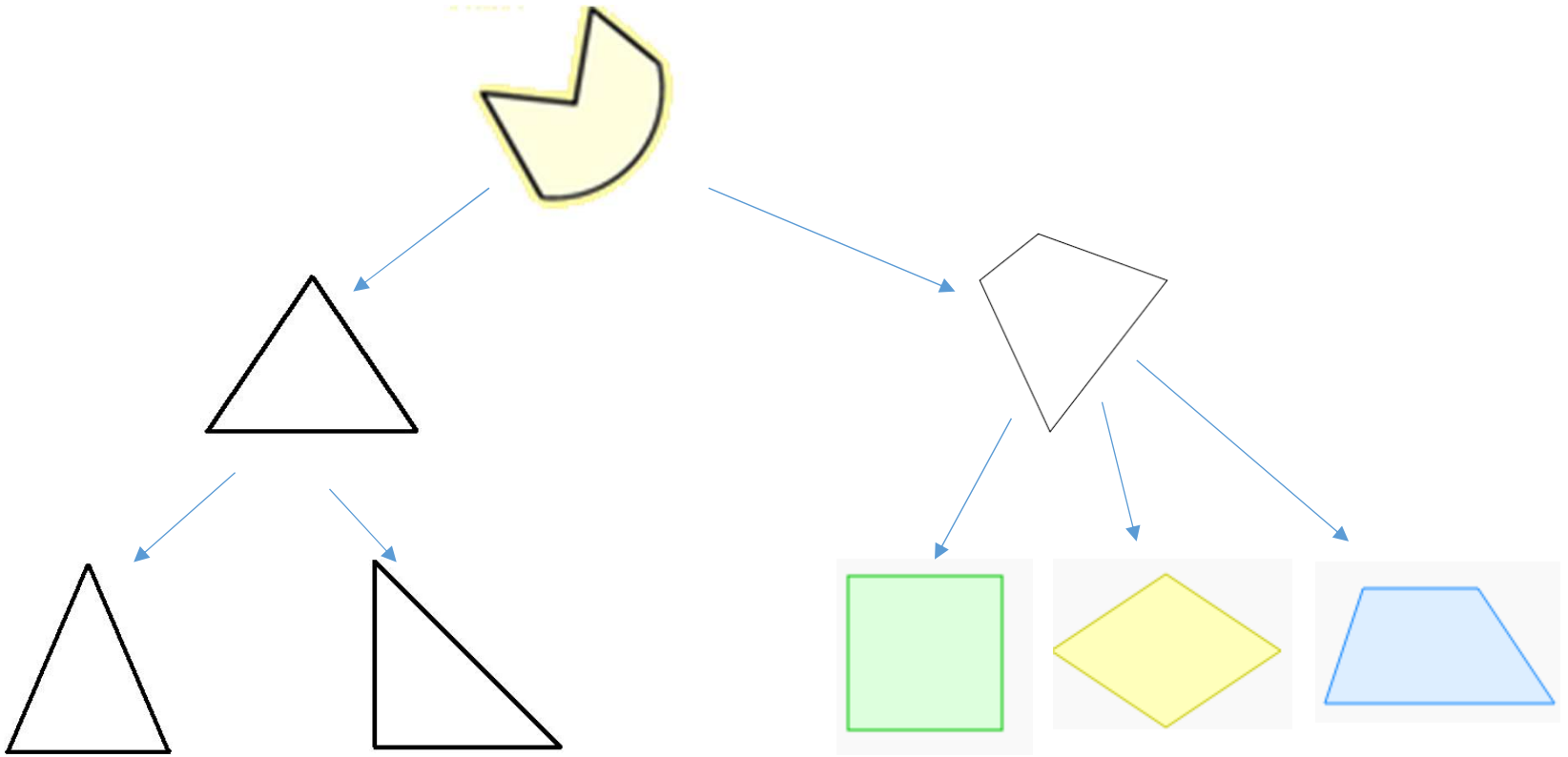
ADATREJTÉS (DATA HIDING)

- Java-ban négyféle minősítő osztály van, kulcsszavak:
  - \- private (privát, belső) – színésznő súlya, PIN-kód, SMS
  - \# protected (védett) – saját lakás (én, örökösök)
  - \+ public (publikus, nyilvános) – lakcím, név
  - csomagszintű (nincs jelölése) – TV-műsor a lakásban (akit beengedsz, látni fogja)

Facebook-nál melyik minek felelne meg?

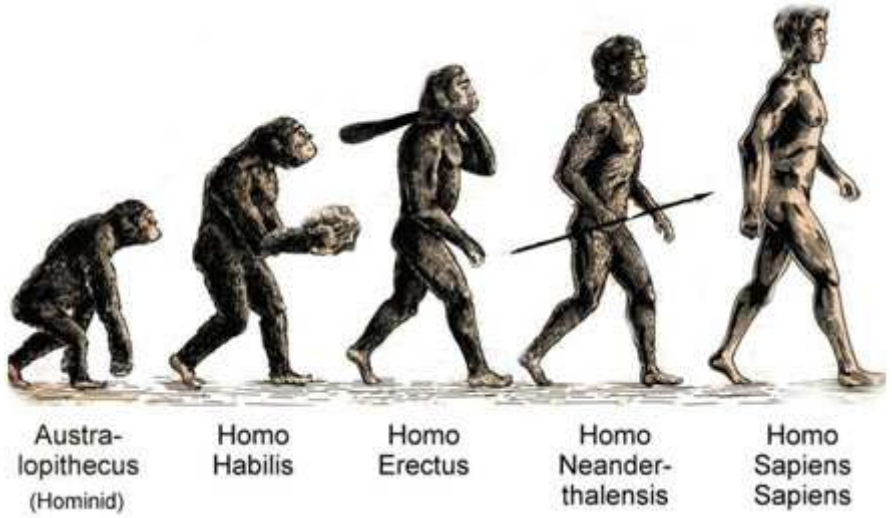
Gyakorlatban private-tal és public-kal fogunk dolgozni. (Hol?)
- Private: belső művelet tudja használni (kormány elfordítása)
- Public: kívülről lehet hívni (rendszám)

ÖRÖKLŐDÉS (INHERITANCE)



ÖRÖKLŐDÉS LÉNYEGE

- Van egy ősosztályom (szülő)
- És vannak leszármazottak (gyermek, örökös)
- Öröklődésnél új attribútumokat, új műveleteket adunk hozzá
- Példa1: mobiltelefon generációk
  (analóg→ digitális GSM→ UMTS→ LTE →szélessáv)
- Példa2: ember evolúciója:



ÖRÖKLŐDÉS A JAVA-BAN

- Extends kulcsszó
- Mindent örököl az őstől (adattagok, műveletek)
- Gyakorlatilag korlátozás nélkül bővíthető új elemekkel
  - Új adatok (attribútumok)
  - Kiterjesztett műveletek
- Minden osztály azért felel, amit benne definiáltak

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ŐSOSZTÁLY („SZÜLŐ”)

```
public class Ososztaly {

    private int adat1;
    private String adat2;
    public Ososztaly(int adat1, String adat2) {
        this.adat1 = adat1;
        this.adat2 = adat2;
    }


    public int getAdat1() {
        return adat1;
    }


    public void setAdat1(int adat1) {
        this.adat1 = adat1;
    }


    public String getAdat2() {
        return adat2;
    }


    public void setAdat2(String adat2) {
        this.adat2 = adat2;
    }


    @Override
    public String toString() {
        return "Ososztaly [adat1=" + adat1 + ", adat2=" + adat2 + "]";
    }
```

LESZÁRMAZOTT („GYERMEK”)

```java
public class Leszarmazott extends Ososztaly {

    private char adat3;

    public Leszarmazott(int adat1, String adat2, char adat3) {
        super(adat1, adat2);
        this.adat3 = adat3;
    }

    @Override
    public String toString() {
        return "Leszarmazott [adat3=" + adat3 + "]" ;
        //adat1-et nem érem el. Átirhatnám a private-ot publicra
        //az ososztalyban, de az nem jo megoldas. Megoldás: getter-setter
    }
```

- Objektumban használható: this; super

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

THIS ÉS SUPER KULCSSZAVAK

- Konstruktor hívások
- This – mindig az adott objektumra vonatkozik
- Super() – az ősre vonatkozik
  - Elérjük az adatot, mivel örököljük
  - Ősosztály adattagja
  - Ősosztály metódusa
  - Ősosztály konstruktora

PÉLDA: ÖRÖKLŐDÉS

Hozzunk létre egy Jármű ősosztályt (String gyarto, String tipus, int ev), aminek leszármazottja az Autó (új adattagja: int ajtokSzama). Hozzunk létre egy-egy példányt, és írassuk ki konzolra!

```
public Jarmu(String gyarto, String tipus, int ev) {
    this.gyarto = gyarto;
    this.tipus = tipus;
    this.ev = ev;
}

@Override
public String toString() {
    return this.gyarto + " " + this.tipus + " <" + this.getClass().getSimpleName()
            + " > Gyártás éve: " + this.ev;
}
```

```
public class Auto extends Jarmu{
    public int ajtokSzama;

    public Auto(int ajtokSzama, String gyarto, String tipus, int ev) {
        super(gyarto, tipus, ev);
        this.ajtokSzama = ajtokSzama;
    }

    @Override
    public String toString() {
        return super.toString() + " Ajtók száma: " + this.ajtokSzama;
    }
```

FELADAT: ÖRÖKLŐDÉS

Egy kis cégnél hárman dolgoznak:

- Adél dolgozó, fizetése 200.000 Ft.
- Feri a főnök, a fizetése 300.00 Ft, és neki jár 50.000 Ft pótlék
- Imre az igazgató, jár neki fizetés (500.000), pótlék (80.000), és a cég bérel számára egy BMW-t is.

Hozzuk létre az ősosztályt és leszármazottait!
Figyeljünk a toString-re a kiíratáskor!

```
public String toString() {
        return getClass().getSimpleName() + " - " + nev  + " " + fizetes + "Ft";
}
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

TÖBBALAKÚSÁG (POLIMORFIZMUS)

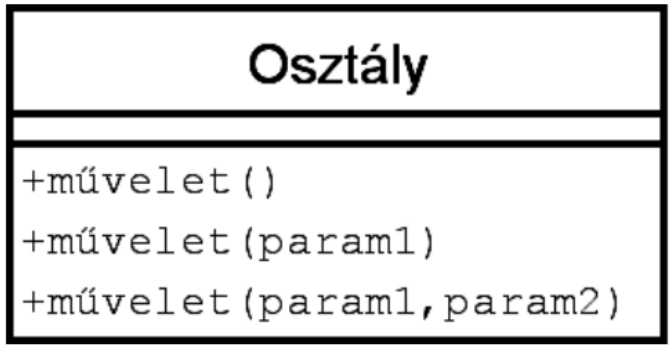
- Egy változó élettartama alatt (futás közben) más osztálybeli értéket is felvehet az öröklődési láncban.

- Kontextustól függően ugyanarra az üzenetre más reakció

- Típusai:
  - Felültöltés (overload)
  - Felüldefiniálás (override)

FELÜLTÖLTÉS (OVERLOAD)

- Egy objektum egy-egy üzenetre másképp reagálhat.
- A hasonló feladatokat azonos névvel jelölhetjük.



Felültöltött metódusok

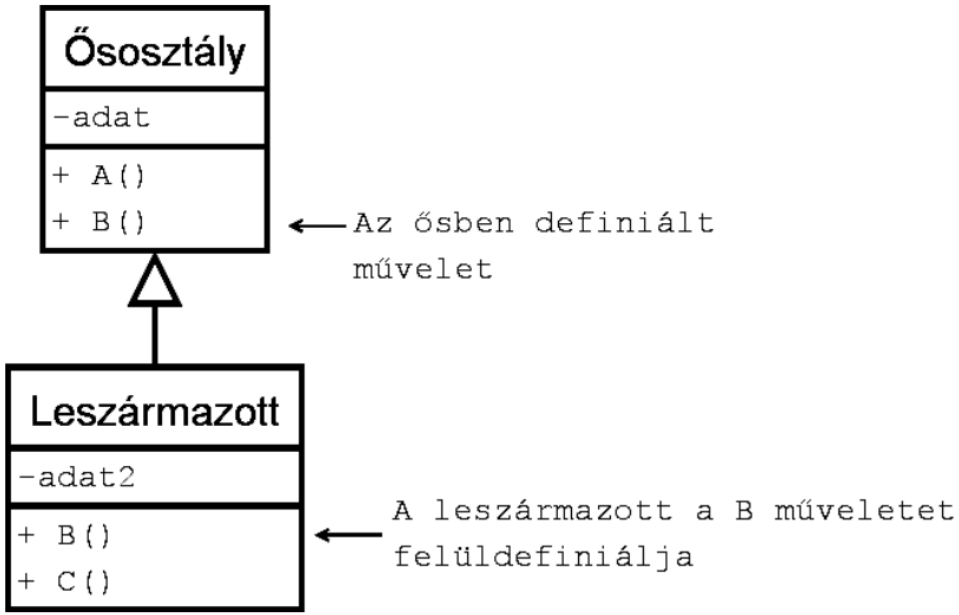
```
public void kiirNev(String keresztNev) {
    System.out.println("A tanuló keresztneve: " + keresztNev + ".");
}

public void kiirNev(String vezetekNev, String keresztNev) {
    System.out.println("A tanuló teljes neve: " + vezetekNev +
            " " + keresztNev + ".");
}
```

FELÜLDEFINIÁLÁS (OVERRIDE)

A leszármazott új műveletet tud bevezetni.
Példa: 2G mobil már tud SMS-t küldeni

Prog. Példa: toString: minden szinten felüldefiniálható



SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

Köszönöm a figyelmet!