SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

Java programozás

3. óra

Absztrakt és interfész

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OKTATÓ

Németh Richárd

Egyetemi tanársegéd
Széchenyi István Egyetem, Informatika Tanszék
Okl. gazdaságinformatikus
Jog-Állam-Politika (MTA-"A" kat.)
Olvasószerkesztő
Java fejlesztő

Iroda: A-602
E-mail: nemeth.richard@ga.sze.hu
Konzultációk: online, ill. személyesen (előtte 2 nappal jelezni e-mailben)
Fogadóóra: hétfő, 13:50 – 14:50

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

TANANYAG, SZÁMONKÉRÉS

ELEKTRONIKUS JEGYZET

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

Diák: feltöltésre kerülnek a Moodle rendszerbe

SZÁMONKÉRÉS

- A tárgy gyakorlati beadandó feladattal teljesíthető
- A feladat a később közzétett feladatok közül választható
- A feladat önállóan kidolgozandó (max. idő: 2 hét)
- Határidő: szorgalmi időszak vége

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

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

NÉHÁNY BILLENTYŰPARANCS A KEZDETEKHEZ

Funkció	NetBeans	Eclipse
Auto kiegészítés	CTRL + SPACE	CTRL + SPACE
Szövegkiíratás	sout (+ CTRL+ SPACE)	syso (+ CTRL + SPACE)
Adott sor törlése	CTRL + X	CTRL + D
Adott sor duplázása	CTRL + SHIFT + ↑ ↓	CTRL + ALT + ↑ ↓
Kód formázása	ALT + SHIFT + F	CTRL + A, utána CTRL + I
Kódrész beszúrás	ALT + Insert	ALT + SHIFT + S
Inputok rendezése	CTRL + SHIFT + I	CTRL + SHIFT + O
Blokk kommentezés ki/be	CTRL + SHIFT + C	CTRL + 7
Build futtatása	F6	CTRL + F11
Refaktorálás	CTR + R	ALT + SHIFT + R
Gyorstipp mutatása	ALT + ENTER	CTRL + 1

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

SZÖVEG KIÍRATÁSA KONZOLRA

1. Feladat:

Mindenki írassa ki a nevét, születési évét és a PI értékét a konzolra – előzőleg megadott változóból!

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

SZÖVEG KIÍRATÁSA KONZOLRA

1. Feladat megoldása:

```
String nevem = "Richárd";
int szul_evem = 1982;
double pi_erteke = 3.1415926535;

System.out.println("A nevem: " + nevem);
System.out.println("A születési évem: " +
szul_evem);
System.out.println("A PI értéke: " + pi_erteke);
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

HASONLÍTÁS: IF

- 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)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

HASONLÍTÁS: IF

03. 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
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

HASONLÍTÁS: IF

03. Feladat megoldása:

```
int szam = 20;
if (szam % 2 == 0 && szam%5 == 0) {
	System.out.println(szam + " maradék nélkül osztható 2-vel és 5-tel is");
}
else if (szam % 2 == 0) {
	System.out.println(szam + " maradék nélkül osztható 2-vel, de 5-tel
nem");
}
else if (szam % 5 == 0) {
	System.out.println(szam + " maradék nélkül osztható 5-tel, de 2-vel
nem");
}
else {
	System.out.println(szam + " 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");
}
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

NÖVEKMÉNYES (LÉPTETŐ) CIKLUS: FOR

04. Feladat:
Írassuk ki konzolra a páros számokat -10-től 100-ig,
ha a szám osztható 7-tel!

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

NÖVEKMÉNYES (LÉPTETŐ) CIKLUS: FOR

04. Feladat megoldása:

```
for (int i = -10; i<101; i++) {
    if (i%7 == 0 && i%2 == 0) {
        System.out.print(i + "\t");
    }
}
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

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 példánya van megosztva a példányokkal.
  Pl. public final static String KONSTANS = "Konstans";

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

KONSTRUKTOR

- Konstruktor: Az osztály belépési pontja, ami az osztály létrehozásakor automatikusan meghívódik.
- Létrehozhatom kézzel, de bill.paranccsal beszúrható:
  Alt + Insert ► Constructor (Source ► Complete Code)
- Mindig az osztályon belül hozom létre!

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

- A konstruktor megadása a példányosítás (new kulccszó!)
- Konstruktor neve ugyanaz, mint az osztályé
- Nincs visszatérési értéke

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

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

PÉLDÁNYOSÍTÁS

07. Feladat:
Hozzunk létre egy Kutya osztályt, aminek adattagjai:
kutyaNeve, kutyaFajtaja, kutyaKora, isIvartalanitva, tulajdonos.

Hozzunk létre belőle 3 önálló példányt!

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

PÉLDÁNYOSÍTÁS

07. Feladat megoldása:

```
public class Kutya {

    private String kutyaNeve;
    private String kutyaFajtaja;
    private int kutyaKora;
    private boolean isIvartalanitva;
    private String tulajdonos;

    public Kutya(String kutyaNeve, String kutyaFajtaja, int kutyaKora, boolean isIvartalanitva, String tulajdonos) {
        super();
        this.kutyaNeve = kutyaNeve;
        this.kutyaFajtaja = kutyaFajtaja;
        this.kutyaKora = kutyaKora;
        this.isIvartalanitva = isIvartalanitva;
        this.tulajdonos = tulajdonos;
    }

    @Override
    public String toString() {
        return "Kutya [kutyaNeve=" + kutyaNeve + ", kutyaFajtaja=" + kutyaFajtaja + ", kutyaKora="
                + kutyaKora + ", isIvartalanitva=" + isIvartalanitva + ", tulajdonos="
                + tulajdonos + "]";
    }
```

```
Kutya bloki = new Kutya("Blöki", "labrador", 4, true, "Zolika");

System.out.println(bloki);
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OBJEKTUMORIENTÁLT ALAPELVEK

- Osztály és objektum

- 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.

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ALAPELVEK MEGVALÓSULÁSA 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?

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

EGYSÉGBEZÁRÁS

- Egységbezárás (Encapsulation) – 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)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ADATREJTÉS (DATA HIDING)

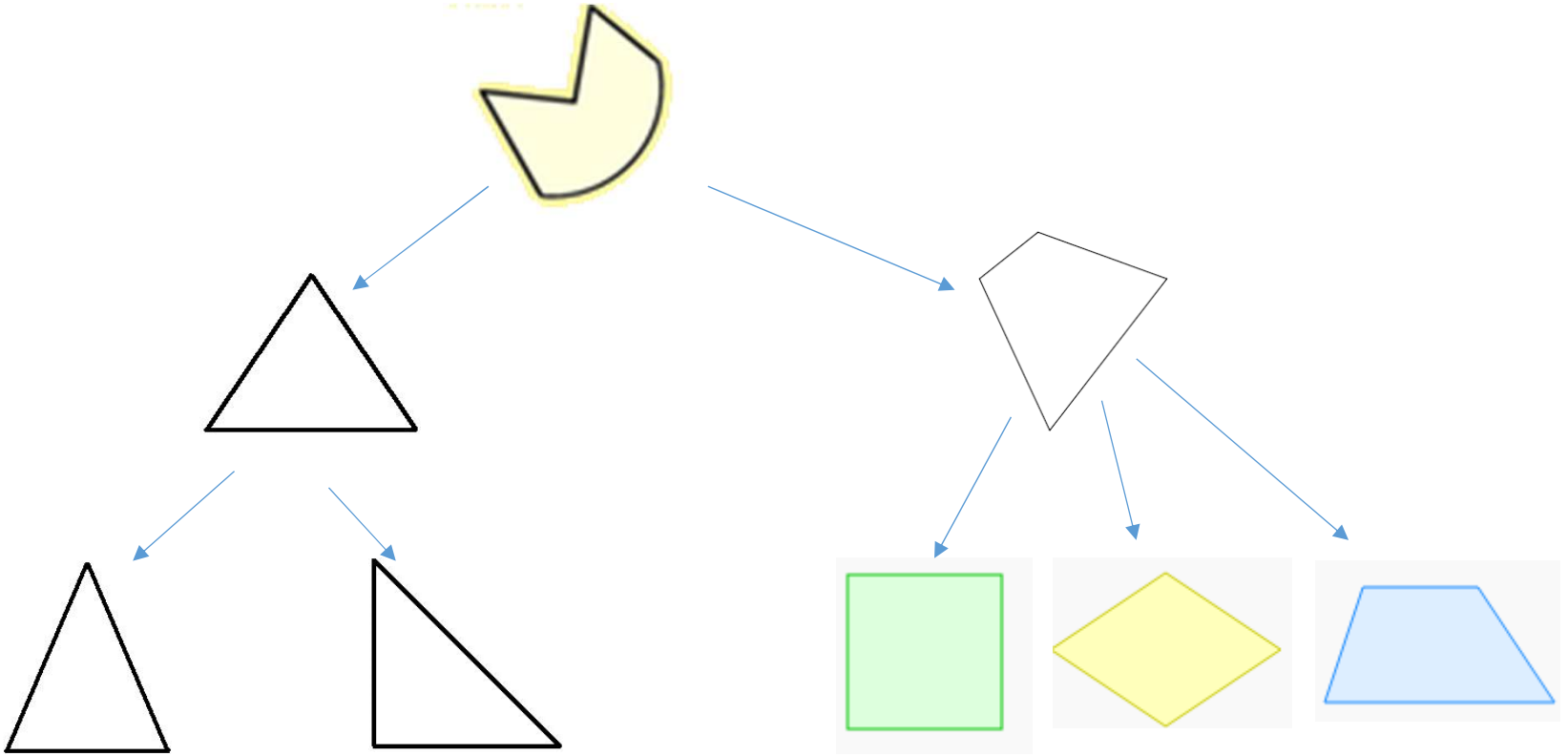
- Java-ban négyféle minősítő osztály van, kulcsszavak:
  - private (privát, belső) – színésznő súlya, PIN-kód, SMS
  - # protected (védett) – saját lakás (én, örökösök)
  - + public (publikus, nyilvános) – lakcím, név
  - csomagszintű (nincs jelölése) – TV-műsor a lakásban (akit beengedsz, látni fogja)

Facebook-nál?

Gyakorlatban private-tal és public-kal fogunk dolgozni. (Hol?)

- Private: belső művelet tudja használni (kormány elfordítása)
- Public: kívülről lehet hívni (rendszám)

ÖRÖKLŐDÉS (INHERITANCE)

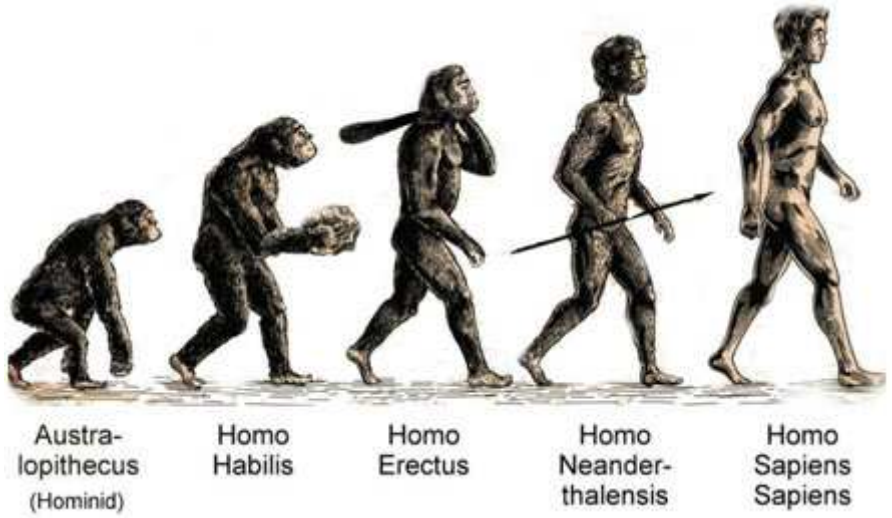


SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

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:



SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

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

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

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

FELADAT: ÖRÖKLŐDÉS

14. Feladat:

Egy kis cégnél hárman dolgoznak:
- Adél alkalmazott, 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 (50.000), és a cég bérel számára egy BMW-t is.

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

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

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

FELADAT: ÖRÖKLŐDÉS

14. Feladat megoldása (ősosztály):

```
public class Alkalmazott {

    private String nev;
    private int fizetes;

    public Alkalmazott(String nev, int fizetes) {
        super();
        this.nev = nev;
        this.fizetes = fizetes;
    }


    @Override
    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.

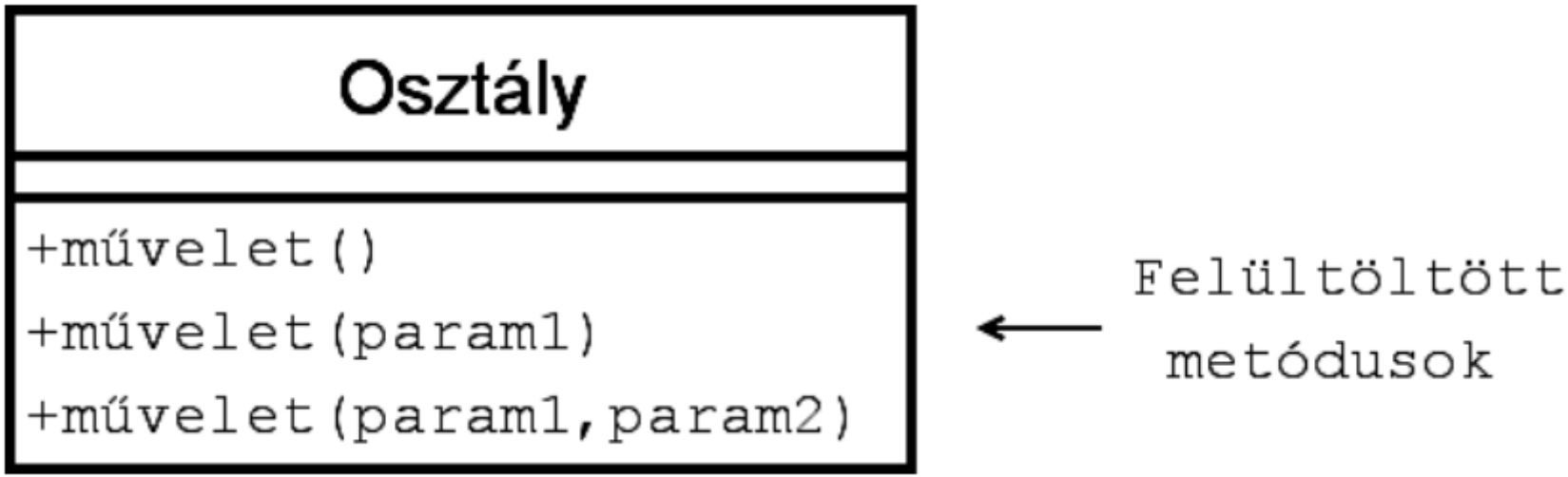
- Kontextustól függően ugyanarra az üzenetre más reakció

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

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

FELÜLTÖLTÉS (OVERLOAD)

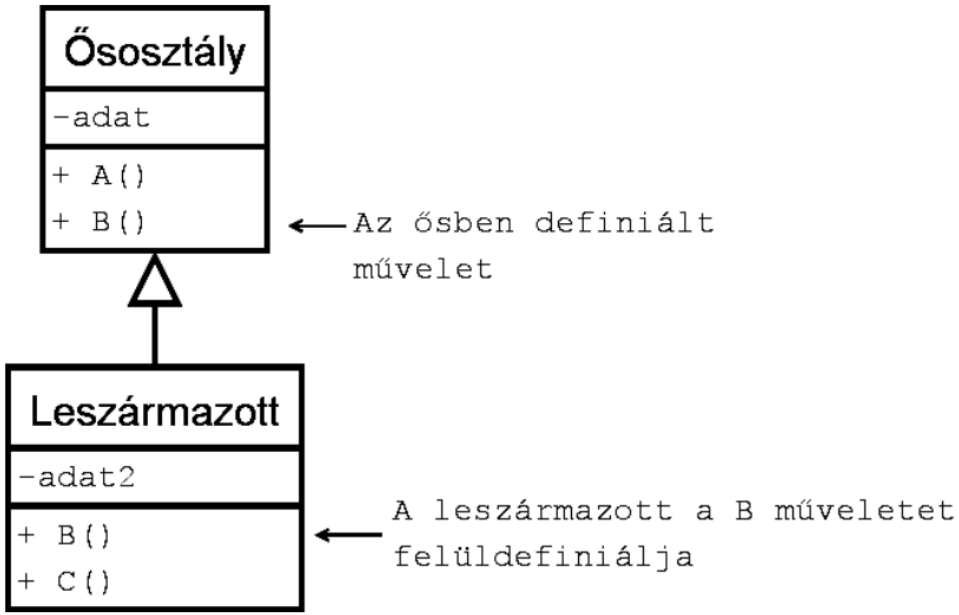
- Egy objektum egy-egy üzenetre másképp reagálhat.
- A hasonló feladatokat azonos névvel jelölhetjük.



SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

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

FELÜLDEFINIÁLÁS PÉLDA (VÁLTOZÓ ToString();)

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

    Tanulo norbi = new Tanulo("Kiss Norbert");
    System.out.println(norbi);
    Vizsgazo vizsgaNorbi = new Vizsgazo("Kiss Norbert", 5);
    System.out.println(vizsgaNorbi);
```

```
@Override
public String toString() {
    return "A tanuló neve: " + nev + ".";
}
```

```
@Override
public String toString() {
    return super.toString() + " Érdemjegye: " + this.vizsgajegy + ".";
}
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ABSZTRAKT, INTERFÉSZ - BEVEZETÉS

Absztrakció szintjei:

- Absztrakt osztályok
- Absztrakt metódusok
- Interfészek

- A „class” utáni kód az osztály definíciója
- Main-ben példányosítok: new kulcsszó (konstruktor hívás!)

- Osztály: leíró, az adott objektum tervrajza
  pl. egy Porsche Carrera tervrajza:
  tudom, hogy kell legyártani, de még nem gurul

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ABSZTRAKT OSZTÁLYOK

- Konkrétan nem példányosítjuk (nem is lehet!)
- A leszármazottakat implementáljuk
- „Abstract” kulcsszó + üres törzsű metódusok:

```
public abstract class Jarmu {

// megjelenik egy olyan metodus, amit nem tudom még, h hogyan működik

 public abstract int getSzallithatoSzemelyekSzama();
```

- Pl. jármű egy absztrakt fogalom, nem konkrét
  - lehet bicikli, traktor, roller, lovaskocsi, gépkocsi…
  - lehet állati erővel vont vagy gépi erővel működő
  - minden közlekedési eszköz jármű, de még nem konkrét

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ABSZTRAKT OSZTÁLYOK - FOLYTATÁS

- Vannak olyan elemek, amit meg kell adni (pl. világítás)

- Mitől absztrakt?
- Közvetlenül NEM PÉLDÁNYOSÍTHATÓ, mert nincs még pontos definícióm adott műveletre/lekérésre (pl. hány kereke van? Nem tudjuk. De! van kereke)

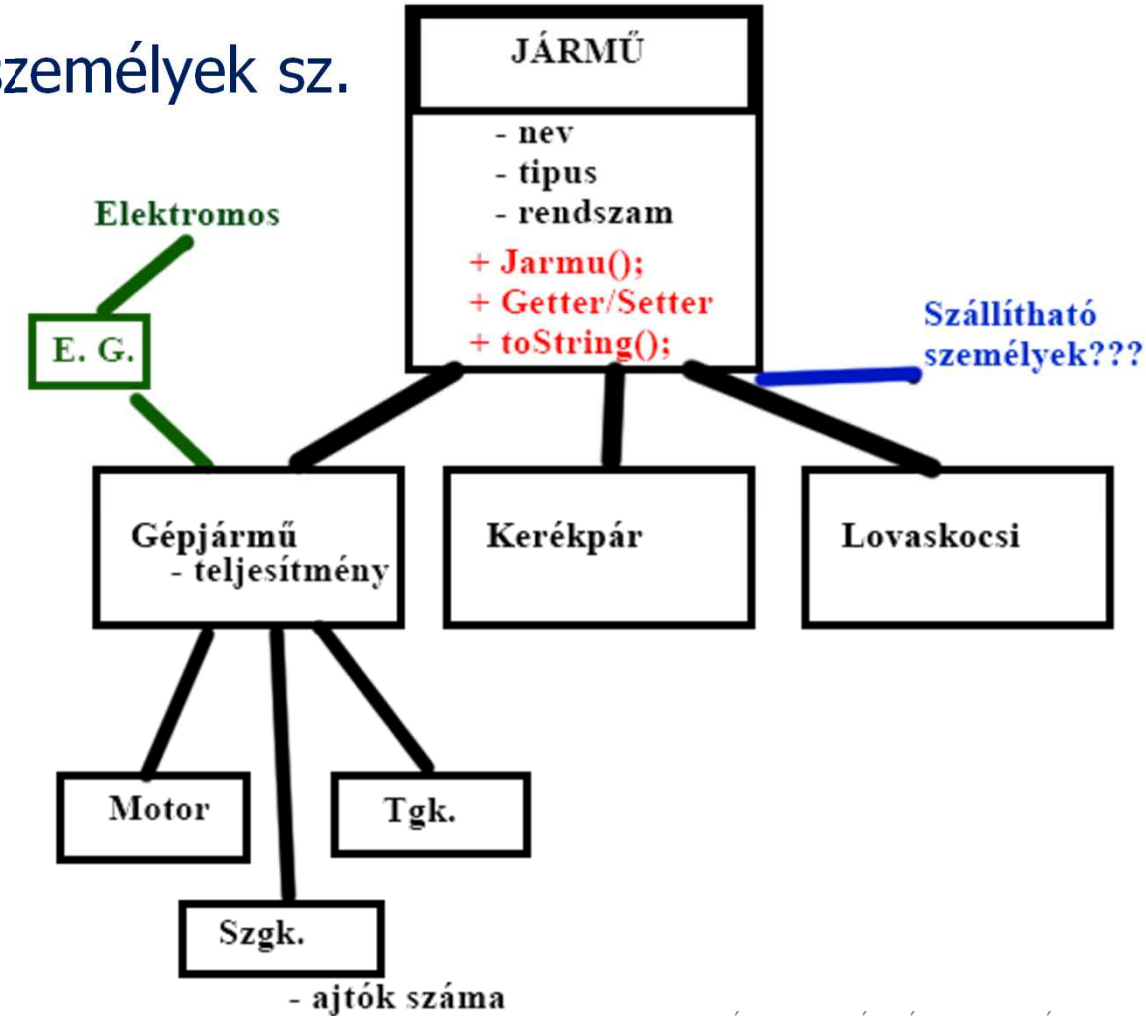
- Mindig egy ősosztállyal kezdem, és ennek lesznek leszármazottai

ABSZTRAKT OSZTÁLYOK - FOLYTATÁS

- Absztrakt: szállítható személyek sz.
- Előny:
  1. Nem kell mindig megadni
  2. Módosításkor 1x

→ Közös ős metódusok
Ős: szállíthat
Leszármazott:
pontosan hány főt

IMPLEMENTÁLNI KELL!



SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ABSZTRAKT OSZTÁLYOK - FOLYTATÁS

- Példánál maradva: csak az lehet Jarmu, aki az ősosztály egy leszármazottja (pl. Jarmu jarmu1 = new Gepjarmu(…);
  de NEM! Jarmu jarmu1 = new Jarmu(…); - absztrakt!)

Java szigorúan típusorientált
→ ha valamit int-nek definiálok, az végig int marad.
Viszont objektumoknál a dinamizmus megengedett!
(Ha Jarmu-nek definiálom (bal oldalon), csak azokat a tulajdonságokat érem el, amit Jármű szinten definiáltam).

- Mire jó? Pl. ha ArrayListben akarom felhasználni, ahol maga az AL Jármű → de mindenféle leszármazott beletartozhat

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

ABSZTRAKT OSZTÁLYOK ELŐNYEI

- A megírt osztályokat később is fel tudjuk használni!
(pl. parkolóház: x db hely van biciklinek, y autónak, 0 kamionnak)

- Minél egyszerűbb a kód, annál könnyebb átlátni
→ közös definíciókat csak egyszer kell megadni
→ közös definíciókat csak egy helyen kell átírni

► KÓD ÚJRAFELHASZNÁLHATÓSÁGA

Példa: Csinálok egy tömböt, aminek elemei Jarmu objektumok leszármazottai → FOR ciklussal mindnél kiírhatom a szállítható személyek számát, mindegy hogy személygépkocsi vagy bicikli

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

INTERFÉSZEK

- Ha már annyira absztrakt, hogy nincs benne konkrétum

- Olyan típusok, amiben csak konstans definíciók vannak
(ne legyen névváltoztatás két vizsga között ☺)

- Interfész publikus → objektumok összekötésére szolgál

- 2 külön szemlélet: absztrakt vs interfész (keressünk rá)

- Osztály: főnév (pl. Személygépkocsi) ← MI?

- Interfész: melléknév, mn. igenév (pl. futtatható) ← MILYEN?
- Java API: Runnable, Comparable… (mire is képes?!)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

MIKOR HASZNOS AZ INTERFÉSZ?

- BPM (Business Process Modelling, pl. RUP)

Üzleti modell → megrendelő akar egy szoftvert
Pl. autószerelő: kocsilejelentés, eredetvizsgálat, áfás számla

Első megbeszélések, specifikációk:

- MIRE LEGYEN KÉPES (számlázható, lejelenthető, határidős…)

- Interfészekben fogalmazzuk meg a specifikációkat
  → Fejlesztő ezek alapján implementálja a feature-öket

Pl. webáruházban lehet online fizetni
(Paypal-lal fizethető; Barion-nal fizethető…)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

INTERFÉSZ – OSZTÁLY KÜLÖNBSÉGEK

Kutya, macska, nyúl → osztályok (menhely)
Hallgató, oktató, rektor → osztályok (egyetem polgárai)

DE! lehet angolt tanuló hallgató, aki közben sítáborba is megy
→ nem származtathatom a hallgatóból, már van egy közös ősük

→ Interfészeket csinálok: angolosok, sítáborosok, kézisek…

Egy osztály beletartozhat akárhány interfésszel meghatározott csoportba → hallgató lehet élsportoló, hacker, angolos stb.

De megadhattam volna boolean-nal is (isAngolos?) - absztrakt

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

INTERFÉSZ VAGY ABSZTRAKT?

- Ízlés kérdése (ha nem írják elő, munkahely vagy vizsgapélda)

- Tyúk vagy tojás?

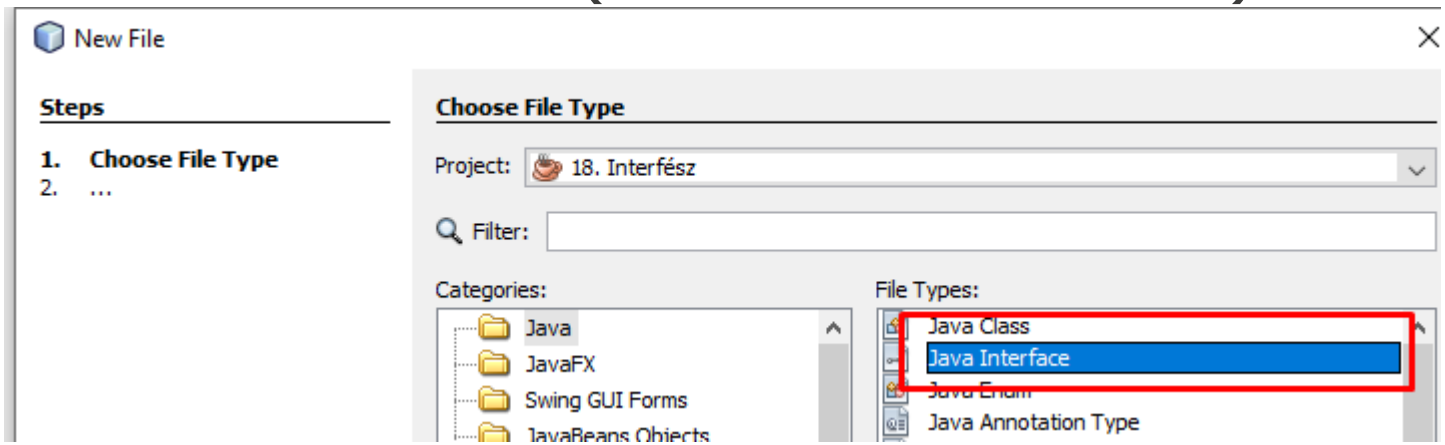
- Előbb csempézzek, vagy előbb fessek?

↓ ↓ ↓

- Két külön iskola:
  - Csak absztrakt, és leszármaztatok
  - Csak osztályok és interfészek

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

INTERFÉSZ PÉLDA (JÁRMŰ FOLYTATÁSA)



- Jarmu típusainkhoz új közös interface: mivel megy?
  - benzines: fogyasztás, liter/100 km
  - elektromos: kapacitás

Kulcsszó: implements

Pl. ElektromosGepjarmu extends Gepjarmu implements Elektromos

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

INTERFÉSZ TOVÁBBI ELŐNYÖK

- Egyszerre több interfészem is lehet
  (pl. kocsi hibrid: egyszerre elektromos és benzines)
- EB-selejtezőn játszó focista játszhat a BL-ben is

- Többszörös öröklődés NINCS, de többszörös interfész IGEN.

- Többszörös interfész a példánkban:
  public class Hibrid extends Személygépkocsi implements Elektromos, Benzines { … }

- Az objektum csak azt tudja, amire szüksége van, de azt biztosan tudni fogja (nyelvvizsgán: Bayes-tétel vs. igeidők)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

Köszönöm a figyelmet!