SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

Java programozás

3. óra

Absztrakt és interfész

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

GYAKORLÁS: KARAKTER ÉS STRING MŰVELETEK

15. Feladat:

Ezzel a gyerekverssel kapcsolatosak a feladatok→

„Alma alma
piros alma
odafönn-a-fán.
Ha elérném,
nem kímélném,
leszakítanám.”

a) Hány nagybetűvel kezdődő sor volt?
b) Volt-e „alma” tartalmú TELJES sor?
c) Volt-e olyan sor, amiben szerepelt az „alma” szó?
d) Az első szóban hány betű és hány szám volt?
e) Hogy nézne ki az első szó az első és az utolsó betű nélkül?
f) Ahol volt kötőjel, ott daraboljuk fel, és írjuk ki a részeit!

g) (megjegyzés: az idézőjel nem része a versnek! ☺
A feladatban használjunk ArrayList-et!)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

15. FELADAT HELYES KIMENETE:

```
A) Ennyi sor kezdődött nagy betűvel: 2
B) Volt-e "alma" szöveg (sor)? Nem volt
C) Volt-e olyan sor, amiven volt "alma" szo? Igen, volt
D) Az első szóban (Alma) 4 betű és 0 szám volt.
E) Az első szó (Alma) az első és utolsó kar nélkül: lm
F) A kötőjelet tartalmazó sor(ok) szavanként:
odafönn
a
fán.
```

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!

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OSZTÁLYOK EGYMÁSRA HATÁSA

- Az osztályok egymásnak üzenetet küldenek, és ezzel valamilyen művelet elvégzésére, vagy tulajdonsága megváltoztatására kényszeríti a küldő a fogadót, azaz:

- Az adott osztály private tulajdonságát egy másik osztályból annak public metódusán keresztül állíthatjuk.

Tankcsata: a harctéren egy Tigris tank rálő egy katonára.
- Tank adatai: típus: Tigris, sebzés: 100, páncél: 1000.
- Katona adatai: név: Béla, sebzés: 10, életerő: 120.

Tank metódusa: rálő (tigris.ralo(bela)).
Bela meghívott metódusa: sebzodik(sebzes).

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OSZTÁLYOK EGYMÁSRA HATÁSA – MAIN CLASS

- Figyeljük meg, hogy az egyik osztály végrehajt egy műveletet, ami a másik osztályon fejti ki a hatását
  → ralo(objektum) metódus

```
public class Tankcsata {

    public static void main(String[] args) {
        katona bela = new katona("Béla", 100, 200);
        Tank tigris = new Tank("Tigris", 100, 1000);
        tigris.ralo(bela);
    }
```

- Egy szuicid hajlamú tank rá tud lőni saját magára?

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

OSZTÁLYOK EGYMÁSRA HATÁSA – TANK ÉS KATONA

- Tank „sebzes” értékét vonjuk le Katona „eletero” értékéből

```
public void ralo(katona aldozat) {
    System.out.println(this.getClass().getSimpleName() +
            " rálő " + aldozat.getNev() + " katonara");
    aldozat.sebzodik(this.sebzes);
    System.out.println("Eltalálta. " + aldozat.getNev() +
            " életereje most már csak: " + aldozat.getElet
}
```

```
new Tank("Tigris", 100, 1000);
```

```
public void sebzodik(int sebzes) {
    if (this.eletero>sebzes) {
        this.eletero = this.eletero - sebzes;
    } else {
        this.eletero = 0;
    }
}
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

16. FELADAT – OSZTÁLYOK EGYMÁSRA HATÁSA

Készítsünk Gyumolcsfa osztályt (név, gyümölcsök száma)
Készítsünk Gyermek osztályt(becenév, éhes-e)
Mivel a Gyermek éhes, ezért gyümölcsöt eszik a fáról.
Megehet bármennyi gyümölcsöt a megadott fáról (de nyilván ne többet, mint ahány gyümölcs van rajta).

- Gyermek metódusa: megeszik(Gyumolcsa fa, int mennyit);
- Évéskor a fa setGyumolcsokSzama(int gyumolcsokSzama) metódusa kerül meghívásra
- Ha nincs elég gyümölcs, írjuk ki.
- Az elfogyasztott gyümölcsök számot vonjuk le a fától (akár többet is).
- Ha sikerül enni, akkor a Gyermek már nem éhes.
- Vegyünk fel 1 fát, 2 gyermeket, és felváltva egyenek a fáról.
- Minden evés után írjuk ki az aktuális állapotokat (Fa, Gyerek1, Gyerek2)

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

A 16. FELADAT EGY LEHETSÉGES KIMENETE:

```
run:
Almafa        : 12 db gyümölcse van
Évike         : éhes
Gerzsonka     : éhes
Évike megevett 5 db gyümölcsöt.
Almafa        : 7 db gyümölcse van
Évike         : jóllakott
BUILD SUCCESSFUL (total time: 1 second)
```

SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

TÖBBALAKÚSÁG (POLIMORFIZMUS)

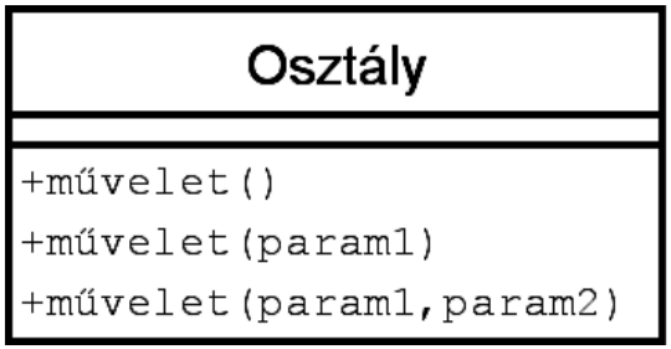
- Egy változó élettartama alatt (futás közben) más osztálybeli értéket is felvehet az öröklődési láncban.

- 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

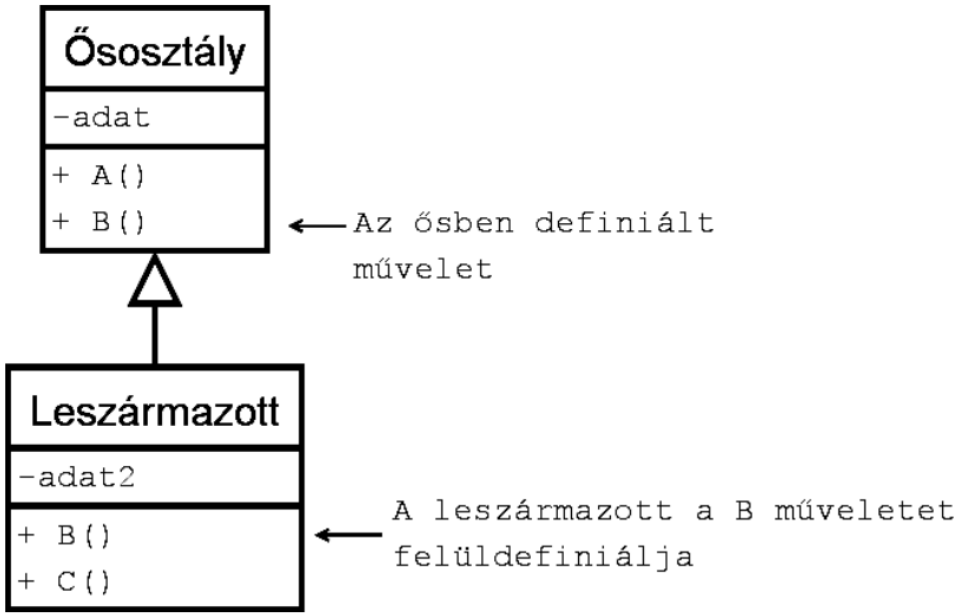
```
public void kiirNev(String keresztNev) {
    System.out.println("A tanuló keresztneve: " + keresztNev + ".");
}

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

```
zoli.kiirNev(zoli.getKeresztNev());
```

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

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)

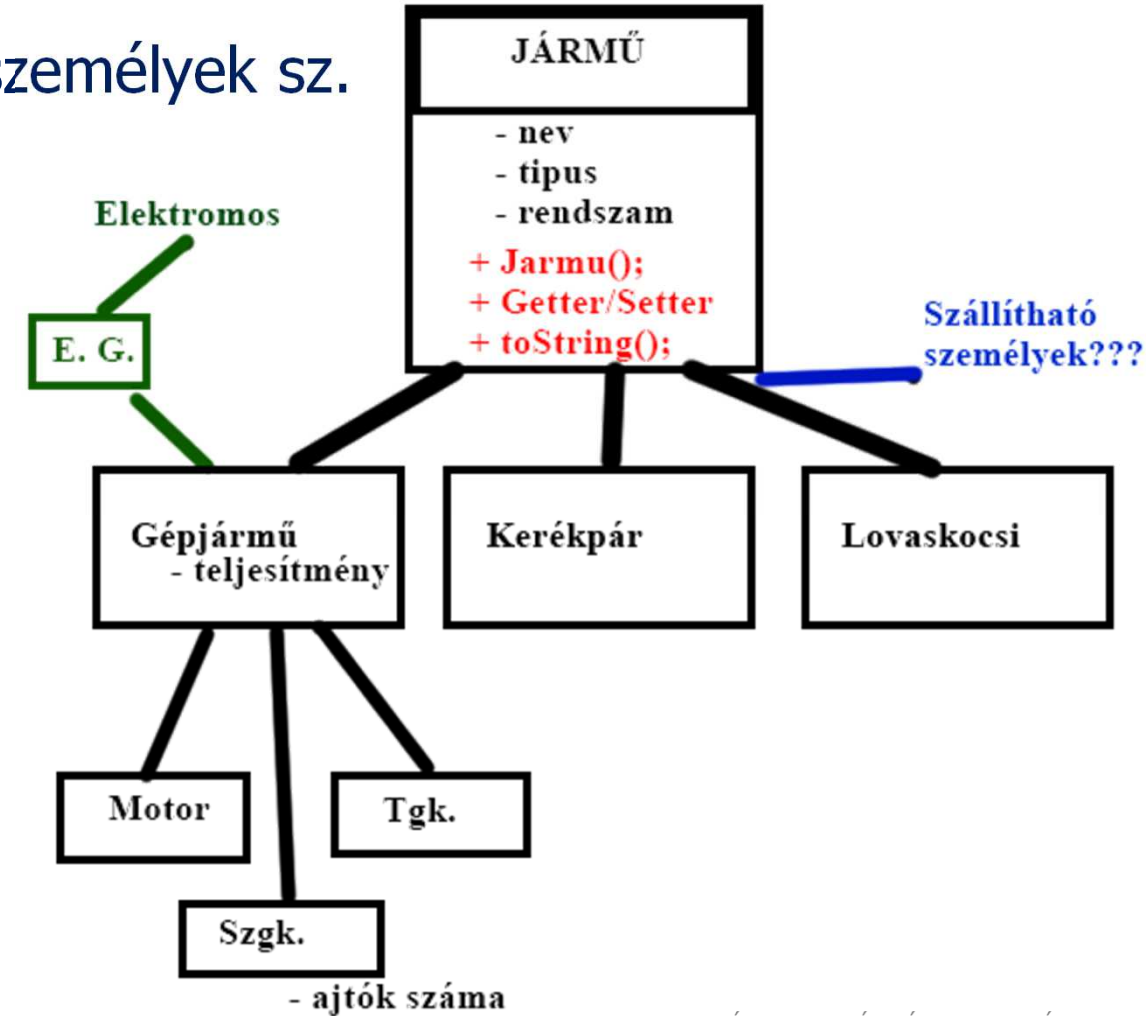
- Mindig egy ősosztállyal kezdem, és ennek lesznek leszármazottai

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?

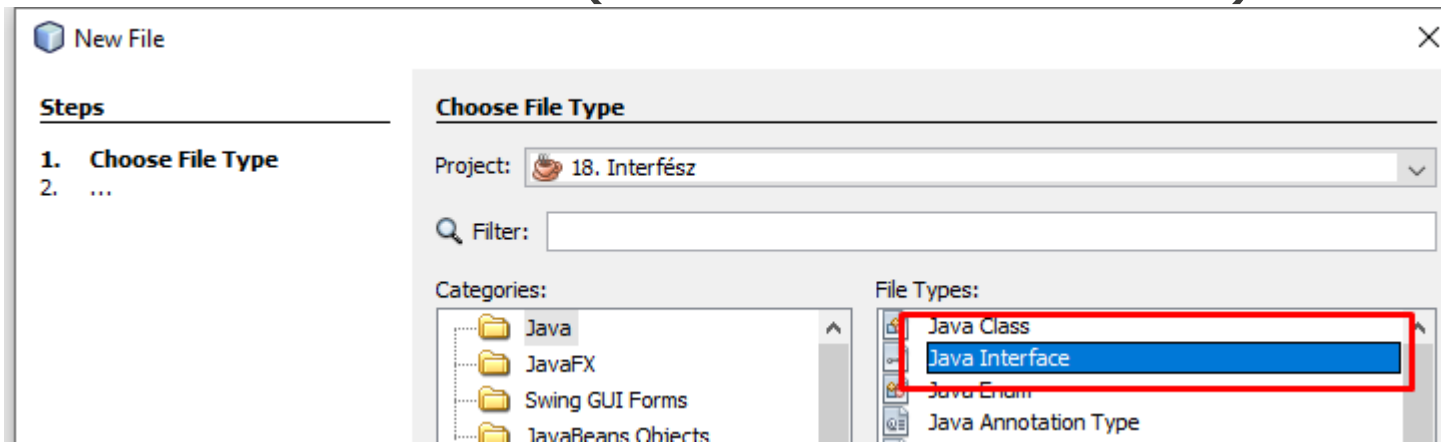
- Előbb csempézzek, vagy előbb fessek?

↓ ↓ ↓

- 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!