1. előadás

Áttekintés I.

Programozás (2) előadás

2022. Szeptember 5.

Halász Gábor

Debreceni Egyetem

Általános tudnivalók

Ajánlott irodalom:

- Nyékyné G. Judit (szerk): Programozási nyelvek, Kiskapu, 2003.
- Juhász, István: Magas szintű programozási nyelvek 2, elektronikus egyetemi jegyzet, 2009
- Tarczali, Tünde: UML diagramok a gyakorlatban, Typotex Kiadó, 2011.
- Angster, Erzsébet: Objektumorientált tervezés és programozás: JAVA, 4KÖR Bt., 2002, ISBN: 9632165136
- Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python, O'Reilly Media, 2009

Félév teljesítésének feltételei: jelenlét + 2 gyakorlati + 1 elméleti ZH

Érdemjegy: $1 < 60\% \leq 2 < 70\% \leq 3 < 80\% \leq 4 < 90\% \leq 5$

További részletek: `https://elearning.unideb.hu/`

Áttekintés

Tematika

- OO paradigma alapfogalmai. (OO $\equiv$ ObjektumOrientált)
- Egységbezárás, osztály, objektum, adattagok, metódusok, lekérdező/beállító metódusok.
- Konstruktorok, destruktorok, inicializáló függvények, példányosítás.
- Öröklődés, polimorfizmus, statikus és dinamikus kötés, láthatósági szintek.
- Operátorok túlterhelése.
- Többszörös öröklődés, absztrakt osztályok, belső osztályok. Interfészek, kollekciók.
- OO tervezés folyamata, heurisztikák.
- UML diagramok (használati eset, osztálydiagram).
- Problémák modellezése OO szemlélet alapján.
- Szövegelemzés, szövegbányászat az üzleti életben.
- Webes szövegbányászat.

Programnyelvek osztályozása

Programnyelvek osztályozása

- Imperatív nyelvek
  - imperatív ≡ parancsoló (hang), kényszerítő (körülmény)
  - parancs orientált nyelvek
- Deklaratív nyelvek
  - deklaratív ≡ kijelentő, ellentmondást nem tűrő
  - „probléma leíró” nyelvek
- Máselvű (egyéb) nyelvek

Imperatív nyelvek

- Algoritmikus nyelvek: a programozó mikor egy programszöveget leír, algoritmust kódol, és ez az algoritmus működteti a processzort.
- A program utasítások sorozata.
- Legfőbb programozói eszköz a változó
- Szorosan kötődnek a Neumann-architektúrához.
- Alcsoportjai:
  - Eljárásorientált nyelvek
  - **Objektumorientált nyelvek**

Deklaratív nyelvek

- Nem algoritmikus nyelvek.
- Nem kötődnek szorosan a Neumann-architektúrához
- A programozó csak a problémát adja meg, a nyelvi implementációkba van beépítve a megoldás megkeresésének módja.
- A programozónak nincs lehetősége memóriaműveletekre (csak korlátozott módon.)
- Alcsoportjai:
  - Funkcionális (applikatív) nyelvek
  - Logikai nyelvek

OO paradigma fogalomrendszere

Objektum-orientáltság jellemzői

- az adatmodell és az eljárásmodell elválaszthatatlan (így szemléli a világot)
- absztrakt eszköz és fogalomrendszer:
  - Az újrafelhasználhatóságot olyan magas szintre elviszi, ameddig lehetséges, a valós világot nagyon megközelíti.
- szemlélete: imperatív eszközrendszer
  - (algoritmus – kódolni kell)

(Az OO területén többféle iskola létezik, amelyek bizonyos pontokon élesen vitatkoznak egymással, nem csak nüansznyi különbségek vannak köztük.)

Objektum-orientáltság jellemzői

Objektum (object): Az eljárásorientált nyelvek változó fogalmának kiterjesztése (általánosítása). Jellemzői:

- Attribútum (attribute):
  - ez az adatrész, a struktúra, tetszőleges bonyolultságú adatszerkezet. Szokás ezt az objektum statikus részének is nevezni. Minden objektum mögött van egy jól definiált tárterület, ezen vannak az attribútumok értékeit reprezentáló bitsorozatok.
  - Terminológia: az obektumok állapotairól (state) beszélünk, ahol minden egyes állapotot egy-egy bitkombináció ír le, ami egy jóldefiniált címen van.

Objektum-orientáltság jellemzői

Objektum (object): Az eljárásorientált nyelvek változó fogalmának kiterjesztése (általánosítása). Jellemzői:

- Metódus (method):
  - a viselkedés leírására szolgál (eljárásmodell leírására) az eljárásorientált nyelvek eljárásai és függvényei. A módszerek adják meg nyelvi szinten az objektum viselkedésmódját (behavior).

Objektum-orientáltság jellemzői

Objektum (object): Az eljárásorientált nyelvek változó fogalmának kiterjesztése (általánosítása). Jellemzői:

- Azonosság (van azonosság tudata):
  - bármely objektum csak és kizárólag önmagával azonos, minden mástól megkülönböztetett.
  - Minden objektumnak van azonosítója (OID: object identifier).
    - Nyelvi szinten ezzel nem foglakozunk.
  - Analógia: (változó – név) (objektum – OID)
    - A változó neve igazából soha nem azonosító csak hatáskörön belül egyértelmű a névhivatkozás. Az OID viszont tényleg egyedi, még programok között is!

Objektum viselkedése

Az objektum állapota időben módosul(hat).

Metódusok csoportjai:

- le tudja kérdezni az objektum állapotát
- meg tudja változtatni az objektum állapotát

Objektumok élettartama

- Az objektumot létre kell hozni, és addig él, amíg meg nem szűnik.
- A megszüntetés lehet a nyelvi rendszer feladata, vagy a programozóé.
- Az objektumazonosító minden szinten él, mindig léteznie kell.

Az osztály (class) fogalma

- Absztrakt eszköz, az eljárásorientált nyelvek típusfogalmának általánosítása (gyakran itt is típusként említjük - szinonimák).
- Az osztály azonos attribútumú és módszerű objektumok együttese.
- Az osztályhoz köthetőek az objektumok; az osztályból származtathatóak az objektumok.

Példány (instance)

Az osztályon belül létrehozok egy objektumot: példányosítás (instantiation).

- Az adott objektum adott osztály példánya.
  - Minden objektum tudja, hogy melyik osztálynak példánya.
- Adott osztályhoz tartozó minden példány ugyanolyan attribútumokkal és metódusokkal rendelkezik.
  - Minden példány tudja, hogy milyen metódusokkal rendelkezik.

Példány (instance)

Az osztályon belül létrehozok egy objektumot: példányosítás (instantiation).

- A metódusokkal mindig konkrét példányon futtathatom le, ezen értelmezhetők:
  - az aktuális példányon.
- Példány létrehozása:
  - ugyanaz az adatszerkezet újra és újra megjelenik a tárban.
  - A módszereket nem többszörözi !

Osztályattribútum, Osztálymetódus

- Létezhetnek olyan attribútumok és olyan metódusok, amelyek nem arra szolgálnak, hogy az egyes példányok állapotait és viselkedését vizsgáljuk velük, hanem magához az osztályhoz tartoznak. (Példányattribútum, példánymetódus; osztályattribútum, osztálymetódus)
  - Osztályattribútum: pl. hány darab példánya van (az osztály kiterjedése).
  - Az osztályattribútumok nem többszöröződnek.

Osztály – példányosítás – objektum

Az OO szemlélet szerint

- először létre kell hozni egy osztályt,
  - leírni, hogy a hozzá tartozó objektumoknak milyen attribútumai és metódusai legyenek.
- És ezek után az osztályhoz kapcsolódóan és osztályon belül létre lehet hozni objektumokat.
- Példányosítás után az osztály példányairól beszélünk.

Öröklődés (inheritance)

Az újrafelhasználhatóság:

- az osztályok nem függetlenek egymástól,
  - speciális viszony értelmezhető közöttük, ez az **öröklődés**.
  - Ez a viszony aszimmetrikus.
- Az öröklődés osztályokhoz kötött fogalom:
  - két vagy több osztály között értelmezhető.
  - A szuperosztályhoz kapcsolódóan tudunk létrehozni alosztályokat.

Öröklésnél azonnal megvan az újrafelhasználhatóság, rendelkezésre áll az összes eszköz.

Öröklődés (inheritance)

- Superclass
  - szuperosztály / szülőosztály / ősosztály / alaposztály
- Subclass
  - alosztály / gyerekosztály / származtatott osztály
- Az alosztály átveszi, örökli a szuperosztály attribútumait és metódusait
  - (azokat, amelyeket a láthatóság módszerével nem tiltottunk le).

Öröklődés (inheritance)

Az alosztály az öröklésen túlmenően:

- új attribútumokat vezethet be
- új metódusokat vezethet be
- újraimplementálhatja a metódusokat
- törölhet attribútumokat
- törölhet metódusokat
- a láthatósági szabályokat újraértelmezheti, hatásukat felfüggesztheti
- átnevezhet attribútumokat
- duplikálhat attribútumokat
- duplikálhat metódusokat

Aszimmetria

- A szuperosztály nem látja, nem manipulálhatja alosztályait, de fordítva igen.
- A szuperosztályt teljes mértékben látja az alosztály.
- Az alosztály minden objektuma objektuma a szuperosztálynak is! Viszont fordítva ez nem áll fenn.
  - Így, ahol a szuperosztály egy példánya szerepel, szerepelhet az alosztály egy példánya is de ez fordítva nem igaz.

Öröklődés (inheritance)

- Egy osztályból tetszőleges számú alosztály származtatható minden nyelvben.
- Az egyes rendszerekben kérdés, hogy: az alosztálynak hány szuperosztálya lehet?
  - egy: egyszeres (single) öröklődés
  - akárhány: többszörös (multiple) öröklődés. Problémák:
    - azonos nevű attribútumok, módszerek esetén: névütközés;
    - ezt a rendszernek kezelnie kell.
    - Rendszerfüggő, hogy hogyan teszi.

Öröklődés (inheritance)

Alosztályból másik alosztály származtatható:

- öröklési hierarchia.
  - Ez egyszeres öröklődés esetén fa,
  - többszörös öröklődés esetén aciklikus gráf.

Öröklési hierarchia

- pl. Alakzat osztály:
  - attributumai lehetnek:
    - vonalvastagság, nagyság, szín, háttér, kitöltöttség ...
  - metódusai lehetnek:
    - kirajzol( ), elforgat( ) ...
- Az Alakzatot elkezdem specializálni, pl. Zárt_alakzat
  - újabb attributumai lehetnek:
    - terület, kerület ...
  - újabb metódusai lehetnek:
    - területszámítás( ), kerületszámítás( ) ...

A Zárt_alakzat Alakzat is egyben, így a Zárt_alakzat minden példánya az Alakzatnak is példánya

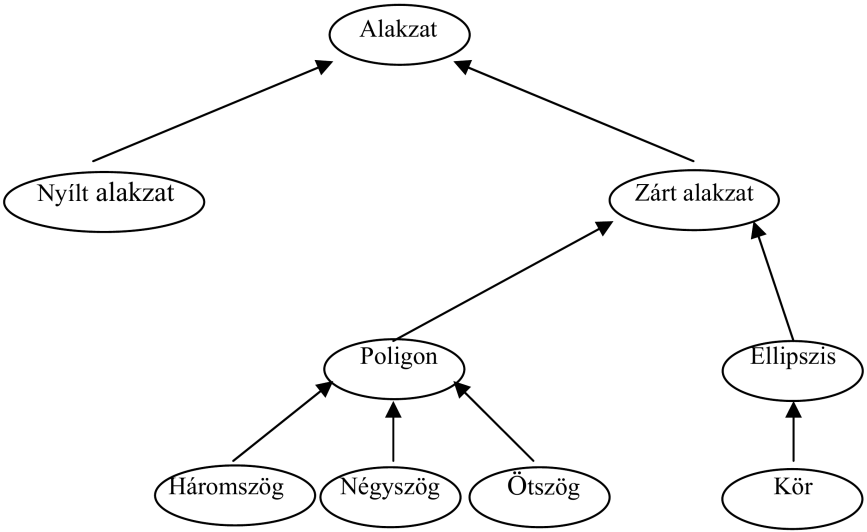
Öröklési hierarchia

Tovább haladva:

- A Zárt_alakzat egyik alosztálya lehet a Háromszög
- A Háromszögnek is lehet egy terület( ) metódusa:
  - átveszem a Zárt alakzattól,
  - de ezt újraimplementálom,
  - hiszen a Zárt alakzat területét csupán közelítőleg tudom megadni, míg a Háromszögét pontosan.

Öröklési hierarchia

Az Alakzat osztály gráfja:



Öröklési hierarchia

- Terminológia:
  - Fa gyökéreleme: ősosztály, amiből az összes többi származik.
  - Előd: pl. a Kör elődjei: Ellipszis, Zárt alakzat, Alakzat.
  - Leszármazott: pl. a Zárt alakzat leszármazottjai: Ellipszis, Kör.
  - Kliens osztályok: azok az osztályok, amelyek között nincsen öröklődési kapcsolat. Pl. Kör – Ötszög.

Bezárás (encapsulation)

- Az OO nyelvek legkényesebb fogalma
  - általában e fogalom mentén válnak el az iskolák, attól függően, hogy melyik mit vall róla.
- Az eljárásorientált nyelvek hatáskör fogalmának, a láthatóságnak a kiterjesztése.
- A legtöbbet félreértelmezett fogalom.

Bezárás_1

- Az osztály egy absztrakt adattípus.
- Az osztály rendelkezik egy interfész és implementációs résszel.
- Az osztály objektumaihoz csak az interfész részen keresztül férhetünk hozzá, az implementációhoz egyáltalán nem, korlátozott hozáférést jelent.
- Ez az információrejtés elve (Information hiding).
- Egy osztály objektumai egy az osztály által definiált interfészen keresztül érhetők el, és csak így!
- A nyelv a benne definiált attributumokat és metódusokat két részre osztja:
  - Nyilvános rész: amelybe tartozó eszközöket minden kliens osztály lát.
  - Privát rész: kívülről nem látható.

Bezárás_2

- A bezárás eljárásorientált nyelvek hatáskör fogalmának általánosítása OO körökben, ahol
- garantáltan létezik egy olyan eszközrendszer, amellyel a programozó tudja szabályozni, hogy az osztályból mi látható és ki számára.

Polimorfizmus (polimorphism)

- Objektum polimorfizmus:
  - Egy objektum objektuma saját osztályának
  - de az öröklődési hierarchiában objektuma valamennyi elődosztálynak is egyben

    Így minden egyes objektum szerepelhet minden olyan szituációban, ahol az ősosztály objektuma szerepelhet, nem csak a saját osztálya példányaként használható.
- Metóduspolimorfizmus (overriding)
  - Egy leszármazott osztály egy örökölt metódust újraimplementálhat:
  - a metódus specifikációja változatlan marad, de az implementáció más lehet az öröklődési vonalon.
    - Zárt alakzat: terület( ), kerület( ) metódus
    - Háromszög: terület( ), kerület( ) metódus (új implementáció)

Kötés (binding)

- A metóduspolimorfizmushoz kapcsolódik.
- Ha van egy függvény és több implementáció hozzá, kérdés, hogy mikor melyik kód kapcsolódik a specifikációhoz. Eszerint beszélünk:
  - Statikus (static) más néven korai (early) kötésről: a névhez a kód hozzárendelődik fordítási időben.
    (Az OO rendszerek többsége fordítóprogram orientált.)
  - Dinamikus (dynamic) vagy késői (late) kötésről: kötés futási időben történik, így ugyanahhoz a névhez más-más kód tartozhat, attól függően, hogy melyik osztálykörnyezetben dolgozunk: az aktuális példány osztályában definiált kód, vagy (ha nincs) a hierarchián felfele a legközelebbi kód kötődik.
- A nyelvek többsége mindkét kötést ismeri, kérdés, hogy melyik az alapértelmezett.

Üzenet (message)

- Tipikusan Smalltalk fogalom.
- A Smalltalk filozófia szerint az objektum üzenetek segítségével kezelhető.
- Ha az objektumtól kérni akarok valamit, akkor üzenetet küldök.
  - Az objektum veszi az üzenetet,
    - én nem tudom, mi történik közben,
    - nem tudom, hogy az objektum hogyan találja ki a választ,
  - és az objektum válaszol.

Absztrakt osztályok/metódusok

- Absztrakt osztályoknak hívjuk azokat az osztályokat,
  - amelyeknek nincsenek példányaik, amelyek nem példányosíthatók.
  - Csak örököltetésre való.
  - Az abszrakt osztályokból konkrét, példányosítható osztályok származtathatók.
- Beszélnek nyelvek absztrakt metódusokról.
  - Ezek azok a metódusok, amelyeknek csak a specifikációjuk van megadva implementáció nélkül.
- Az egész eszközrendszer az absztrakciót szolgálja.

(A rendszerfejlesztési ciklusban és a programfejlesztésnél lesz érdekes.)

Konténer osztályok (Container)

- Olyan adatszerkezet, amely objektumokat tartalmaz.
- Alapvető a tömb, láncolt lista, verem, sor, stb.
- Nem minden nyelvben vannak realizálva a konténer osztályok, a programozónak kell megvalósítania.
- Alapvető szerepük az adatbáziskezelőknél van.

Kollekciók (Collection)

- Objektum-orientált adatbázisok esetén a konténerosztályok helyett a terminológia: kollekció.
- Ezen kollekcióval kapcsolatos az iterátor fogalma.

Iterátor

- Általában ez is egy osztály, típus, ennek példányaihoz tartozó objektumokat be tudjuk járni. A bejárás az adatszerkezeteknek megfelelően történik.

Paraméterezett osztályok

- Egyes objektum-orientált nyelvekben vannak ún. paraméterezett osztályok, a C++ terminológia szerint template-ek. Lényegében megfelelnek osztályszinten az Ada generikusnak.

Objektumok élettartama

A példányosítás mindig egy explicit tevékenység eredménye, minden objektumot minden nyelvben a programozó hoz létre. Meddig él?

- A nyelvek egy részénél az objektumot megszüntetni is explicit módon kell, az objektumok törlése is a programozó feladata. A nem tisztán objektum-orientált nyelvek egy része vallja ezt az elvet. Ld. C++.

Objektumok élettartama

A példányosítás mindig egy explicit tevékenység eredménye, minden objektumot minden nyelvben a programozó hoz létre. Meddig él?

- A nyelvek másik része (nagyobb része) alkalmaz egy automatikus objektum törlési mechanizmust (garbage collection), amelynek a feladata az objektumok megszüntetése aszinkron módon úgy, hogy azzal a programozónak ne kelljen foglalkoznia, és úgy, hogy a törölt objektumok tárhelye ismét felhasználható legyen.

Ez az automatikus tárfelszabadítás nem csak az objektum-orientált nyelvek sajátja, hanem egy tárkezelési technika. Többféle algoritmus van arra, hogy a rendszer hogyan dönti el, hogy mely objektum törölhető. Nyilvánvaló, hogy garbage collection algoritmus sokkal kényelmesebbé teszi a programozást.

Egységesség

- A nyelvben létezik-e más eszközrendszer, mint az objektum?
- Minden objektum, vagy van olyan eszköz, ami nem az?
- Ezek alapján az OO nyelveknek két nagy csoportja van:
  - A tisztán OO nyelvek
  - A hibrid nyelvek

A tisztán OO nyelvek

- A tisztán OO nyelvek azt vallják, hogy minden objektum (osztály, attribútum, metódus, objektum).
- Csak olyan eszközöket tartalmaznak, amelyek obektumorientáltak, és nincs más eszköz.
  - Pl.: Smalltalk, Eiffel csak OO elvek alapján működik.
- A tisztán OO nyelvek esetén e nyelvi rendszer egyetlen osztályhierarchiából áll.
  - Például a Smalltalk egy osztályhierarchia.
- A programozás pedig nem más, mint definiáljuk a saját osztályainkat, és azokat elhelyezem az osztályhierarchiában:
  - az adott osztályhierarchiát bővítük, és ezekből származtatunk objektumokat.

A hibrid nyelvek

- A hibrid nyelvek alapvetően eljárásorientált, logikai, funkcionális, stb. nyelvi eszközöket tartalmaznak, és ez a nyelvi szközrendszer bővül OO eszközrendszerrel.
- Van tehát objektum is, és van nem objektum is.
- Lehetnek bennük eljárásorientált, deklaratív, funkcionális, objektum-orientált eszközök.
- Programozhatunk benne objektumorientáltan is.
- A nem tisztán OO nyelvek esetleg nem is tartalmaznak osztályokat, nincs osztályhierarchia.
- Definiálhatunk önálló osztályokat, és egymástól fügetlen osztályhierarchiákat.
- Itt is vannak szabvány osztálykönyvtárak, csak ezek nem a nyelv részei, és ezektől fuggetlenül is lehet programozni.
- Majdnem minden nyelvnek van olyan kiterjesztése, amelyben szerepelnek OO eszközök. Ilyenek például az OO COBOL, Object Pascal, C++.

Terminológia

- **Objektum alapú nyelvek:** (object-based) ha a nyelvben van objektum fogalom és bezárás, de nincs osztály és öröklés. (Pl. Ada)
- **Osztály alapú nyelvek:** (class-based) van osztály, bezárás, objektum fogalom, de nincs öröklődés. (Pl.: CLU)
- **Objektum-orientált nyelvek:** (object-orinted) minden létezik: bezárás, osztály, öröklődés fogalom. Ezek a nyelvek (imperatív nyelvként) fordítóprogramosak.
- És végül létezik az OO-nak egy olyan speciális nyelve, amelyben nincs osztály fogalom, de minden más OO eszköz megvan benne.