7. Tétel

a) Magasszintű programozási nyelvek II.

Öröklődés

- OOP egyik alapelve

- egy osztályból specializált osztályok (ezek felülírják, kibővítik)

- ős osztály -> gyerekosztály (kevesebb nem lehet az ősénél)

- C#: ":", konstruktor="base", ős lehet interfész vagy absztrakt, de ilyenkor a kifejtés kötelező, 1 osztálynak csak 1 ős osztálya lehet (kivéve interfészből bármennyi)

- Védelmi szintek: private(a gyermek nem éri el) public, protected(a gyermek eléri)

- Mezők újradefiniálása: korai kötés = new kulcsszó

- Metódusok, propertyk újradefiniálása: korai kötés = new kulcsszó, késő vritual és override

Korai kötés, késői kötés

- Korai kötés:

- fordításkor hajtódik végre, oldhatatlan, megváltoztathatatlan, gyorsítja a végre hajtást (new)

- Késői kötés:

- futási időben hajtódik végre, ősben virtual, gyermekben override, lassabb futású

- kiküszöböli a korai kötés problémáját, mert mikor az ősbe visszalépve a virtuális metódushoz ér, megkeresi annak a legújabb változatát és azt futtatja le a gyermekosztályban.

Konstruktorok, konstruktorhívási lánc, osztályszintű konstruktor

- Fogalma és feladata: speciális metódus, a példányosításkor fut le, inicializáláskor, alaphelyzetben állításra használjuk, több konstruktor is lehet, public Osztály() {}

- alapértelmezett konstruktor

- Több konstruktor esetén: ":" után this + megfelelő paraméterek

- Konstruktorhívási lánc: a gyerekosztály konstr. hívása előtt láncreakció és az Object osztályból az aktuális gyermekosztályig az összes szülő-gyerek kapcsolatban lévő osztály konstruktora lefut.

- Osztálysztintű konstruktor: a neve megegyezik az osztály nevével, nincs visszatérési típusa, nincs paramétere, static, private (nem kell kiírni), csak 1 egy osztályban

- Object factory: pl. statikus mezők vizsgálatához

Típuskompatibilitás, object osztály

- az öröklődés elvei szerint működik (mindent átvisz)

- a gyermekosztály átkonvertálható azaz kompatibilis az ősosztállyal

- Minden pulikutya emlős, de nem minden emlős pulikutya, azaz fordítva nem igaz, mert a gyermekosztály már többet tartalmaz)

- is: logikai vissz. érték if(pulikutya is Emlős) = true

- as: típuskényszerítés (is operátorral együtt használatos)

- kollekciók: gyermekosztályok x y szinten közös lista (ilyenkor az fő osztályé pl. emlős)

- Object típus és annak virtuális metódusai:

- az Object osztály minden további osztály ősosztálya (mindennel kompatibilis)

- GetType(): objektum típusát adja vissza

- ToString(): objektumot stringgé alakítja

- Equals(): két objektum egyenlőségét vizsgálja (saját osztály this szóval másik obj)

- GetHashCode(): int értéket generál az objektumhoz, determinisztikus

Lepecsételt osztályok és statikus osztályok:

- Sealed: sealed kulcsszó az osztály elő, példányosítani lehet, de ősül választani nem (pl. Console osztály)

- Static osztályok (Singleton): csak egy db jön létre a memóriában, csak osztályszintű elemeket tartalmazhat, nincs és nem is generálódik konstruktor, nem lehet ősül választani

b) Adatbázisrendszerek I.

SQL nyelv

- strukturált (szabványosított) lekérdező nyelv

- nem algoritmikus nyelv, nem rekurzív

- halmaz orientált nyelv, amely a relációkon dolgozik

- felhasználása a beágyazott SQL esetén harmadik generációs algoritmikus nyelv (C/PL/SQL,Pascal)

- felosztható: DDL, DML, QL, DCL

Relációsémák definiálása

- reláció (tábla) szerkezetét írja le

- R(A1,A2,..An) -> R a reláció neve, A1 az attribútumok nevei

- Attribútumok és típusok:

- Szám(INTEGER), szöveg(VARCHAR), dátum(DATE)

- Megjeleníti: Primary key, Foreign Key, Kapcsolatok (1:1, 1:N, N:M)

- a tervezés során fontos a Normalizáció alkalmazása

Indexek

- gyorsítják a keresést

- közvetlenül megkaphatjuk a keresett sor tartalmát

- két féle módszerrel működik: hash kódok és bináris fák

- Hash szerkezet (kevésbé alkalmazzák)

- Fa szerkezet (legjobb: kiegyensúlyozott fa) növekvő vagy csökkenő sorrendben is végigjárhatjuk

Táblák módosítása

- Új oszlop hozzáadása (ALTER TABLE, ADD COLUMN)

- Oszlop módosítása (ALTER TABLE, RENAME TO/MODIFY COLUMN)

- Oszlop törlése (ALTER TABLE, DROP COLUMN)

- Index hozzáadása (CREATE INDEX ON)

- Index törlése (DROP INDEX ON)

- Táblák átnevezése (ALTER TABLE RENAME TO)

- Új elsődleges kulcs hozzáadása (ALTER TABLE ADD PRIMARY KEY)

- Idegen kulcs hozzáadása (ALTER TABLE ADD CONSTRAINT FOREIGN KEY REFERENCES)

SELECT parancs

- (lekérdezés) SELECT \* FROM Diak;

- Szűrés WHERE záradékkal, rendezés ORDER BY záradékkal

- Szűrés halmazokra DISTINCT kulcsszóval (nem lesznek duplikált sorok)

- Csoportosítás GROUP BY záradékkal (COUNT, SUM, AVG, MAX)

- Szűrés HAVING záradékkal (feltétel a GROUP BY után)

- Kapcsolódás más táblákhoz JOIN záradékkal

Beágyazott lekérdezések

- olyan SQL lekérdezés, amely egy másik lekérdezésen belül található

- összetett adatok lekérdezése egyetlen utasításban

Több táblára vonatkozó lekérdezések

- INNER JOIN: mindkét táblában megtalálható sorokat adja vissza

- LEFT (OUTER) JOIN: visszaadja az összes sorokat a bal oldali táblából és a hozzájuk tartozó jobb oldali táblából ha van, ha nincs NULL

- RIGHT (OUTER) JOIN

- FULL (OUTER) JOIN: összes sorokat visszaadja minden táblából, ahol megfelelő sorok párosítva vannak, ha nincs NULL

- CROSS JOIN: N\*M

- SELF JOIN: önmagával való kapcsolódása

Privilégiumok

- jogosultság a felhasználó számára, hogy az adatbázison műveleteket hajthasson végre

- rendszer-privilégium: DDL és DCL parancsok + adatbázis login

- objektum-privilégium: adatbázis műveletekhez

- GRANT ON parancs

Szerepkörök

- felhasználók csoportosítása, a privilégiumok együttese-

- GRANT EXECUTE a Futtatási jogokhoz (PL/SQL-ben még lesz róla szó)

Tranzakció kezelés, ROLLBACK, COMMIT

- ACID

- automatikusak, de expliciten: BEGIN vagy START TRANSACTION;

- COMMIT: véglegesíti a változtatásokat

- ROLLBACK: visszavonja a változtatásokat