5. Tétel

a) Magasszintű programozási nyelvek II.

Az OOP alapelvei, alapfogalmai

- Egységbezárás: mezők, metódusok védése, adatrejtés: védelmi szintek, érkező elküldött adatok ellenőrzése: propertyk

- Öröklődés:

- 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 virtual és override

- Sokalakúság, polimorfizmus: a gyerekosztályban felüldefiniált metódusokat automatikusan választjuk az ősosztály helyet.

Mezők, metódusok, property-k, adatrejtés

- az osztályban szereplő adattároló, változó

- osztályban szereplő eljárás vagy függvény, az osztály mezőivel végez műveletet

- különleges adatfajta, mezőkhöz hasonlít, a getterrel és setterrel részletesen szabályozható, hogy milyen adatot küld vagy fogad el

- fentebb volt róla szó

Konténerosztályok használata és indexelők

- egy olyan osztályt valósít, ahol lehetőség van a különféle objektumok kívülről történő kezelésére (pl. összehasonlítás, rendezés, szűrés)

- OOP alapelveket itt is be kell tartani

- konténer tárolója főként egy lista -> ICloneable

- Indexelők: speciális property, az adott osztály példányait indexelhetővé teszi (úgy mint egy listánál vagy egy tömbnél) public Klon this[int index] {get {...}}

Osztályszint és példányszint

- mezők:

- osztályszint: static, elérése->Osztálynév.Változónév, élettartama statikus, első hivatkozástól a program futásának a végéig a memóriában maradnak

- példányszint: elérése->Példánynév.Változónév, élettartama dinamikus, a példány élettartamával egyezik meg

- közös: védelmi szint, ha nem adjuk meg automatikusan private, van típusuk, kezdőérték->konstruktor

- konstansok osztályszintűek

- metódusok és propertyk:

- osztályszint: static, kizárólag az osztály metódusait és property-eit használhatja (Oszálynév.Metódusnév())

- példányszint: saját példány: this (Példánynév.Metódusnév())

- közös: védelmi szint, ha nem adjuk meg automatikusan private,

Névterek

- különböző névterekben lehet ugyanolyan nevű osztály (Névtér.Osztály)

- BCL tárolja

- Beépített névterek: pl. System-> System.Windows.Media, System.IO

- teljes hivatkozás kikerülése: using

- Névtér nélküli osztály is lehetséges (globális)

Bővítő metódusok

- csak statikus osztályba írható, csak public static metódus építhető be (pl. megmondja hogy a string 4 betűből áll-e) (public static bool MetodusNev (this string szoveg, ..) {};

Operator overloading

- pl. Date mezőhöz nap hozzá adása

- public static Days operator + (Day datum, Uint hozzaad) {}

- túlterhelhető operátorok: a legtöbb unáris és a bináris (kivéve a =)

b) Architektúrák

A cache használat

- átmeneti információ tároló, célja az információ-hozzáférés gyorsítása (gyorsító-tár)

- (CACHE-HIT CACHE-MISS)

- Felépítése: 2^m sor/db(címtár, valid bit, dirty bit, adatblokk)

- címtár: 32-n biten tárolja az adatblokk tartalmának eredeti helyét

- valid bit: utolsó részben

- dirty bit: utolsó részben

- adatblokk: 2^n byte - egybefüggő memóriarészek

Lokalitási elvek

- Hely szerinti lokalitás: ha szükségem van egy adatra, szükségem lehet a szomszédos adatra is

- Idő szerinti lokalitás: az időben sorban kért adatok valószínűleg a tárolóban is sorban vannak.

A memória hierarchia szintjeinek összehasonlítása

- CPU regiszterei (~0,2ns elérési sebesség)

- CACHE

- Operatív tár (~1ns)

- CACHE

- háttértár (~1ms)

- fontos szempontok: gyorsaság, fajlagos ár (mennyibe kerül 1 byte), méret

A cache(ek) helye a szintek között

- először operatív tár és háttértár közé, majd CPU és operatív tár közé is raktak cache-t

- az operatív tár és CPU között lévő cachenél lett 1.szétválasztott gyorsító-tár és 2.többszintű gy-t.

- 1. külön gy-t. az adatoknak és az utasításoknak

- 2. (L1,L2,L3) nő a méretük és a távolságuk a CPU maghoz képest.

Felépítése, az elérési változatok

- FUC - teljesen asszociatív cache: bármely memóriaterület bármely cache sorba kerülhet, ha a CPU egy memóriacímet keres, a teljes címtárat vizsgálja 2^m összehasonlítás (rugalmas, pontos, de drága és lassú)

- DMC - direkt leképezésű cache: a beolvasott memóriaterület a cache egy meghatározott sorába kerülhet csak, a keresett memóriacímből egyértelműen eldől, mely sorban lehet az adat (1 összehasonlító áramkör, olcsó, gyors, de néha önmagát irkálja felül)

- SAC - csoport asszociatív cache: a beolvasott memóriaterület a cache egy blokkjába kerülhet, átmenet a FUC és DMC között (k blokkokat csinál) (rugalmasabb, kisszámú összehasonlító áramkör kell hozzá)

Címbitek számának meghatározása az elérési módoktól függően

- 32 bites a címbusz, 8 byte-os az adattár n=3, 16 soros cache m=4 k=2

- FUC: 32-3=29 bit | Címtár:címbusz mérete - n

- DMC: 32-3-4=25 bit |Címtár:címbusz mérete - n - m

- SAC: 32-3-4+2=27 bit Címtár:címbusz mérete - n - m + k

Visszaírási módok

- write through: a processzor által módosított adatot kiírja a memóriába (átír a cache fölött) és a cache-be is egyszerre -> lassú, de konzisztensebb

- write back: a módosított adatot csak a cache-be írja ki, a dirty bitet 1-re változtatja, a cache sor törlése előtt írja ki az értéket a memóriába -> más is hozzáférhet a memóriához

A dirty és valid flag-ek szerepe

- valid bit: 1 biten tárolja, hogy az adott cache sorban lévő adat érvénye-e:

- a számítógép bekapcsolásakor minden valid bit értéke 0

- ha egy sorba adatot másolunk, az értéke 1-re változik

- ha fel akarunk szabadítani egy sort mert megtelt a cache, az értékét 0-ra változtatjuk (LeastRU)

- dirty bit: 1 biten ábrázolja, hogy az adott cache sorban lévő adatot a processzor módosította-e, vissza kell-e írni

- ha az adatot nem módosítottuk az értéke 0

- ha az adatot módosítjuk az értéke átvált 1-re és akkor is 1 marad ha a módosítást visszacsináljuk

- ha a dirty bit értéke 1, először vissza kell írni a cache sor tartalmát az operatív tárba és csak utána állítható a sor valid bitje 0-ra