Haladó Fejlesztési Technikák

Gyakorló Feladatok Gyűjteménye

Sipos Miklós
sipos.miklos@nik.uni-obuda.hu

2019

v2.1

Tartalomjegyzék

Útmutató

A dokumentum célja, hogy az órán elhangzottakon és látottakon túl is szolgáltasson gyakorlásra alkalmas feladatokat, leírásokat. A dokumentumban található feladatokat az órán és előadáson elhangzottak mentén javasolt megvalósítani. A feladatok és a várható zárthelyi dolgozatok között semmilyen korreláció nincs, sem tartalomra, sem struktúrára, sem pedig mennyiségre. A cél elsősorban a szükséges ismeretanyag elsajátítása gyakorláson keresztül, majd ezek alkalmazása egyedi problémákra a zárthelyi dolgozat keretein belül.

A feladatok kiírásakor az osztálynevek és metódusok ékezetes formában kerültek kiírásra a könnyebb olvashatóság érdekében. A megvalósításkor a kódban viszont (kommenteket leszámítva) ne használjon ékezetet, sem osztályok nevében, sem metódusokban, sem pedig változókban.

Probléma, meglátás vagy észrevétel esetén a borítón található email elérhetőségen állok szíves rendelkezésre – melyeket ezúton is előre köszönök.

A laboralkalmakon vett kódok, valamint az esetlegesen megoldott feladatok kódjai a https://nik.siposm.hu oldalról érhetők el.

1. Delegáltak, névtelen függvények, lambdák

1.1. Egyszerű feladatok

Valósítsa meg a következő feladatokat:

- hozzon létre egy `SzámVarázsló` delegáltat amelybe három darab `int` típusú bemeneti paraméterrel, valamint `int` visszatérési értékkel rendelkező függvényeket tudunk elhelyezni
- hozzon létre egy `SzámListázó` delegáltat amelybe egy `int[]` típusú bemeneti paraméterrel, valamint `List<int>` visszatérési értékkel rendelkező függvényeket tudunk elhelyezni
- hozzon létre egy `SzámGenListázó` delegáltat amely legyen generikus a T paraméterre nézve; ebbe a delegáltba egy darab `T[]` típusú bemeneti paraméterrel, valamint `List<T>` visszatérési értékkel rendelkező függvényeket tudunk elhelyezni
- hozzon létre egy `SzámListázó` delegáltat amely generikus a T és K paraméterekre nézve; `T[]` és K bemeneti paraméterrel rendelkező metódusokat tudunk feliratkoztatni rá és `List<T>` típussal tér vissza

Az egyes delegáltakból hozzon létre példányokat és helyezzen el beléjük a feltételeknek megfelelő névtelen függvényeket. Az utolsó két delegált esetében az alábbiaknak megefelelően járjon el a névtelen függvény belsejében:

- a `SzámGenListázó` delegált esetén a kapott paraméter int típusokat tartalmazzon, ezeket adja hozzá egy listához, majd a listával térjen vissza
- a két generikus paraméterrel rendelkező `SzámListázó` delegált egyik paramétere (T) legyen `Személy` típusú (név és életkor tulajdonságokkal hozza is létre a megfelelő osztályt), a másik paramétere (K) pedig int típusú; a névtelen függvény feladata, hogy a az adott számnál idősebb elemeket egy listába menti és ezzel a listával térjen is vissza

1.2. Összetett feladatok

Készítse el a következő osztályokat, amelyeket majd a későbbiek mentén kell felhasználni:

- `Élőlény` ősosztály
  - név tulajdonság
  - írja felül a ToString metódust, amely írja ki az élőlény nevét nagybetűs formában majd fűzze hozzá az élőlény típusának nevét (ezt automatikusan kérje le és végezze)
- `Ember` és `Kutya` osztály
  - származzanak le az élőlény osztályból

Készítsen láncolt listát az alábbi instrukciók mentén:

- a `LáncoltLista` osztály generikus legyen a T típusra nézve
- készítse el a listához szükséges ListaElem osztályt is
- a lista rendelkezzen egy SpéciElemBeszúrása eseménnyel, amely egy `SpéciEventArgs` generikus paramétert fogad
- készítsen beszúró és bejáró metódusokat
- beszúráskor a következőre figyeljen:
  - amennyiben speciális elem kerül beszúrásra, úgy süssön el eseményt, amelyben adja át magát a beszúrandó elemet és egy üzenetet is küldjön
  - a paramétereket a `SpéciEventArgs` eseményargumentm osztály segítségével valósítsa meg, amely szintén legyen generikus a T paraméterre
  - speciális elem alatt egy külső függvény által meghatározott elemet értünk; biztosítsa az osztály konstruktorán keresztül, hogy ezt a függvényt/metódust ott tudja átadni a láncolt lista osztálynak; ehhez használjon `Predicate` beépített típust
  - végezzen ellenőrzést az elsütés előtt mind az esemény, mind pedig a delegált esetén

A `Main` részből iratkoztasson fel egy metódust a lista eseményére, illetve készítsen egy névtelen függvényt is lambda segítségével, amely ugyan azt csinálja: kiírja az esemény argumentumán keresztül kapott üzenetet és elem típusát. A lista lértehozásakor adjon meg egy predicate delegáltat, amelyhez a main részben írjon egy névtelen függvényt lambda segítségével; egyik esetben azt vizsgálja, hogy az adott elem neve egyenlő-e egy megadott karaktersorozattal; másik esetben pedig azt, hogy a megadott elem az Kutya típusú-e. Magyarán mondva, ennek segítségével első esetben név alapján tudunk speciális elemet vizsgálni, utóbbi esetben pedig típus alapján. A listához adjon hozzá ember és kutya típusokat, vizsgálja meg a speciális függvény működését, események elsülését.

Ezt követően, a listát egészítse ki egy `RendezésElve` metódussal, amely egy `Comparison` delgáltat fogad. A listába beszúráskor vizsgálja meg, hogy a rendezés elve függvény mit mond – ha nagyobb mint nulla, akkor a lista elejére szúrja az elemet, ellenkező esetben a lista végére. Ehhez a main részben készítsen egy névtelen függvént, amely két paramétert fogad (x, y), és a név jelentse az összehasonlítási alapot.

2. LINQ

2.1. Egyszerű lekérdező feladatok

Készítsen egy `Termék` osztályt, amely rendelkezzen a következő írható és olvasható tulajdonságokkal: termék megnevezése, termék márkája, termék ára, termék bevételének dátuma, termék darabszáma raktáron. Hozzon létre egy `List<Termék>` objektumot és töltse fel elemekkel. A következő feladatokat valósítsa meg első körben metódus szintaktikával, majd pedig LINQ query syntax segítségével. Hasonlítsa össze a kapott eredményeket. Az egyes lekérdezéseket minden esetben külön feladatként kezelje. A megvalósítás során a tanult operátorokat használja, úgy mint orderby, select, where stb.

- válogassa le azokat a termékeket amelyek drágábbak mint 5000 Ft
- válogassa le azokat a termékeket amelyek nevében van 'S' betű
- válogassa le azokat a termékeket, amelyek ára osztható kettővel maradék nélkül és a bevétel dátuma 2019-ben volt; majd rendezze márka szerint csökkenő sorrendbe
- számolja meg hány darab "Apple" márkájú termék van
- számolja meg hány darab "Samsung" márkájú termék van raktáron (azaz az egyes termék márkája legyen Samsung és annak a terméknek a darabszáma legalább 1)
- számolja meg hány darab "Samsung" és "Apple" márkájú termék van együttesen
- kérje le azokat a termékeket amelyek bevételének ideje 2015-ben volt, és rendezze ezen termékeket raktáron lévő darabszám alapján
- értékelje ki, hogy az összes termék ára mennyi, azaz, hogy az aktuális raktárkészlet mekkora értékű
- kérje le, hogy milyen márkájú termékek vannak raktáron, majd rendezze ezeket a márka hossza alapján, a márkákat alakítsa nagybetűs formára egységesen és fűzzön mindegyik elejére egy "$>$" jelet
- számoljuk meg, hogy az egyes márkákból hány darab van raktáron
- számoljuk meg, hogy az egyes márkákból hány darab van raktáron és mennyi azok márkánkénti átlagára

2.2. Összetett XML feladatok I.

Végezze el a következő feladatokat a tanultaknak megfelelően. A feladatok elvégzéséhez szükséges XML alapú adatbázist itt találja meg:
http://users.nik.uni-obuda.hu/siposm/db/workers.xml

- olvassa be a fájl tartalmát, kérdezze le az összes elemet és írassuk ki olyan formában, mintha az XML dokumentumot látnánk
- kérdezze le csak azokat az embereket, akiknek a neve 'T' betűvel kezdődik
- kérdezze le a polihisztorokat
- gyűjtse ki a nevét és telefonszámát egy listába azoknak akiknek az email címe gmail-es
- kérdezze le a mestertanárokat (név és email cím legyen csak lekérdezve)
- kérdezze le az "Alkalmazott Informatikai Intézetet"-ben dolgozók nevét, rendezze őket ABC szerinti csökkenő sorrendbe
- kérdezze le, hogy az egyes intézetekben hányan dolgoznak
- kérdezze le, hogy az egyes irodákban hányan vannak
- kérdezze le, hogy a harmadik legkisebb intézetben hányan dolgoznak
- nevezze át az "Alkalmazott Informatikai Intézetet" "AII"-re minden ott dolgozó esetén, adjon hozzá egy új dolgozót, majd mentse el egy új dokumentumként (ellenőrzés képpen olvassa be és győződjön meg róla, hogy sikeres volt)
- készítsen egy `XMLToList` metódust amely egy `XDocument` típust fogad, hozzon létre egy `List<Oktató>` típusú listát és bejárva a kapott XML dokumentumot a listához adja hozzá az elemeket (a szükséges `Oktató` osztályt is hozza létre a megfelelő tulajdonságokkal)

2.3. Összetett XML feladatok II.

Készítsen egy `BinárisKeresőFa<T>` osztályt a tanult módon, amely a feladat alapját fogja adni. A feladathoz szükséges adatbázist ezen elérhetőségen (http://users.nik.uni-obuda.hu/siposm/db/treeworkers.xml) találja meg. Ennek az XML állománynak egy részét alább láthatja, amelyben bizonyos részek "..." jelölést kaptak, a teljes fájlt érdemes áttekinteni a feladat végrehajtása során.

```
...
<person>
  <name>Tony Stark</name>
  <email>tony.stark@nik.uni-obuda.hu</email>
  <dept>Alkalmazott Matematikai Intezet</dept>
  <rank>polihisztor</rank>
  <phone>+36 (1) 666-1111</phone>
  <room>BA.0.00</room>
  <friends>
    <person>
      <name>Bruce Banner</name>
      <email>bruce.banner@gmail.com</email>
      <dept>Alkalmazott Matematikai Intezet</dept>
      ...
    </person>
    <person>
      <name>Dr. Sergyan Szabolcs</name>
      <email>sergyan.szabolcs@nik.uni-obuda.hu</email>
      <dept>Alkalmazott Informatikai Intezet</dept>
      ...
    </person>
  </friends>
</person>
...
```

A keresőfába hozza létre a szokásos `Bejárás` és `Beszúrás` metódusokat. Utóbbi esetén adjon át a fa osztálynak egy `Comparison` delegáltat, ez fogja majd meghatározni, hogy a fába való beszúráskor bal vagy jobb gyerekbe szúrunk.

Készítsen egy `Személy` osztályt is, amely rendelkezzen a következő tulajdonságokkal: név, email, munkahely, beosztás, telefonszám, szoba. Készítsen továbbá az osztályhoz egy `ToString` felüldefiniálást. Az osztály valósítsa meg az `IComparable` interfészt, és a metódusban a nevet vegye az összehasonlítás alapjául.

A `Main` részben példányosítsa le a fát személy típusra, hozza létre és adja át a szükséges `Comparison` delegáltat és tesztelje le a fa metódusait. A delegált esetében személy típust használjon és készítsen egy névtelen függvényt amely két paraméteres, és név alapján alkalmazzon összehasonlítást.

Olvassa be a `treeworkers.xml` állományt és végezze el a következő feladatokat:

- kérje le az első személyt (és vele együtt az alá tartozó barátokat is)
- az így kapott adathalmazból kérje le csak Tony Stark adatait, és hozzon létre ebből egy `Személy` típust
- állítsa elő a két barátot is Tony esetén, de gondoljon arra is, hogy ha `n` darab barátja lenne
- kérje le a további személyeket az XML fájlból és állítson elő személy objektumokat amiket egy listában tároljon el
- szúrja be az elemeket a fába ebben a sorrendben: Tony, Bruce, Szabolcs majd a többiek a listából

3. Adatbázis kezelés (EFCore)

3.1. Egyszerű feladatok

Készítse el az adatbázist a tanultaknak megfelelően, minta adatbázisnak használja az `Emp` és `Dept` táblákat, melyeket itt ér el http://users.nik.uni-obuda.hu/siposm/db/empdept.sql.

- kérdezze le az összes dolgozót az emp táblából és írja ki a nevüket
- kérdezze le azokat a dolgozókat, akik fizetése 1000-3000 között van
- emelje meg azok bérét 1.3 százalékkal, akik jelenleg 1000 alatt keresnek
- hozzon létre véletlenszerű generálással alkalmazottakat és ezeket rögzítse az adatbázisban
- a SALES osztályon dolgozók juttatását emelje meg a jelenlegi érték duplájára
- a RESEARCH osztályon dolgozókat, akiknek a nevében 'S' betű szerepel büntesse meg, fizetésüket állítsa át az aktuális érték felére
- kérje le azokat a dolgozókat, akik decemberi hónapban léptek be
- kérdezze le azokat a dolgozókat, akiknek a főnökük a 7839-es kódú dolgozó
- gyűjtse ki, hogy az egyes hónapokban hány alkalmazott lépett be

3.2. Kereszttáblás feladatok

- kapcsolja össze az `emp` táblát a `dept` táblával a `deptno` mező alapján
- írja ki a dolgozók nevét, hogy melyik osztályon dolgoznak és az osztály telephelyének nevét
- kérdezze le a dolgozók átlagos jövdelmét (fizetés + juttatás) szervezetenként
- emelje meg azon dolgozók juttatását a fizetésük félszeresére, akik jelenleg nem kapnak egyáltalán juttatást és az ügyosztályuk telephelye New York
- kapcsolja össze az `emp` táblát az `emp` táblával a `mgr` mező alapján és írja ki, hogy adott dolgozó mennyit keres (juttatás nélkül), melyik osztályon dolgozik valamint a főnökének adatait (név, kereset, ügyosztály)
- kérdezze le azokat a dolgozókat, akik valakinek (egy másik dolgozónak) a főnökei

4. Reflexió

4.1. Egyszerű feladatok I.

Hozzon létre egy `Személy` osztályt a következő tulajdonságokkal: név, életkor, nem (logikai), születési dátum. Adjon hozzá az osztályhoz tetszőleges metódusokat, privát és publikus láthatósággal egyaránt.

- készítsen metódust, amely a `Személy` típus tulajdonságait lekérdezi és kiírja a konzolra
- készítsen metódust, amely igazat ad vissza, ha a `Személy` típusnak van "Életkor" nevű tulajdonsága, hamisat egyéb esetben
- készítsen metódust, amely a `Személy` típus metódusait lekérdezi és kiírja egy txt fájlba
- készítsen metódust, amely a `Személy` típus metódusait lekérdezi és kiírja egy xml állományba, egy megfelelően felépített struktúrát követve

4.2. Egyszerű feladatok II.

Készítsen egy új projektet, amelybe hozzon létre tetszőleges osztályokat (pl. Hallgató, Kertész, Ember, Kutya, stb.) tetszőleges metódusokkal és tulajdonságokkal. Alakítson ki az osztályok között öröklődést, amely révén bizonyos metódusok és/vagy tulajdonságok öröklődnek. Készítsen egy `ReflectionAnalyser` osztályt, benne egy metódust amely reflexió segítségével futás időben detektálja, hogy az adott programban milyen típusok (osztályok) vannak definiálva. Kérje le ezeket, és hozzon létre belőlük .txt valamint .xml állományokat, és mentse is el őket. Ennek segítségével olyan programot lehet készíteni, amely lementi, hogy milyen osztályok vannak definiálva. Készítsen ebből .dll állományt, amelyet egy új, teljesen független projekthez adjon hozzá és tesztelje is le abban a környezetben a helyes működést.

4.3. Komplex feladatok I.

Hozzon létre egy `LáncoltLista` osztályt a tanultaknak megfelelően (egyirányú, egyszerű lista elegendő). A lista legyen generikus a `T` paraméterre nézve. Készítsen három segédosztályt az alábbiaknak megfelelően:

- `Állat` ősosztály (absztrakt)
- `Kutya` osztály: név és életkor tulajdonság, valamint ugrik metódus
- `Macska` osztály: név, életkor és életek száma tulajdonság, valamint nyávog és mászik metódus
- `Liba` osztály: súly és életkor tulajdonság valamint, lépegetés metódus

Hozzon létre egy metódust amely a listába beszúrást intézi. Jelen esetben a listába állatokat tudunk elhelyezni úgy mint kutya, macska vagy liba, de lehet, hogy a jövőben lesznek még egyéb állatok is a rendszerben! Beszúrás előtt vizsgálja meg, hogy a paraméternek kapott objektum milyen típusú. Mivel a típus meghatározza, hogy milyen metódussal rendelkezik az entitás (például Kutya esetén ugrik metódus) ezért a következő logikát valósítsa meg a beszúráskor: ugrik metódussal rendelkező entitásokat a lista elejére; nyávog metódussal rendelkező entitásokat a lista végére helyezzen el. Ami nem e kettő közül valamelyik, azok esetén készítsen egy túlcsordulási területet a lista osztályon belül és ide helyezze el az entitásokat. Ez utóbbi esetben esemény formájában jelezzen.

4.4. Komplex feladatok II.

Készítse el a következő osztályokat:

- `CurrentLocation` enumeráció, amelynek lehetséges értékei: Earth, Mars, Vormir, Titan
- `AvengerAttribute` amely egy `CurrentLocation` enumerációt tartalmaz és fogad konstruktorában paraméterként
- `SavedLivesAttribute` amely egy (int) limit és egy (string) percentage tulajdonsággal rendelkezik, amelyeket fogad konstruktorának paramétereiként is
- `Avenger` osztály, `Name` és `SavedQuantity` tulajdonságokkal, továbbá `Fight` nevű metódussal
- `NotAvenger` osztály, `Name` tulajdonsággal

Rövid magyarázat a fentebbi osztályokhoz: a feladatban bosszúállókat kell kezelni, amelyek a megadott paraméterekkel rendelkeznek. Egy bosszúállót akkor vehetünk fel a csapatba (lásd lentebb) ha megfelel a követelményeknek, azaz legalább valamennyi embert megmentett már – SavedQuantity tulajdonság.

Készítsen egy `Headquarters` osztályt, amely rendelkezzen egy `EnrollAvenger` metódussal, amely egy object típust fogad. A kapott paramétert, sikeres felvétel esetén egy belső `List`-ben tároljon el. Sikeres felvétel akkor van, ha a paraméternek kapott leendő bosszúálló már legalább 30 embert megmentett. Ennek vizsgálatát attribútum segítségével valósítsa meg (`SavedLivesAttribute`). Helyezzen el egy ilyen típusú attribútumot a SavedQuantity tulajdonságra 30-as értékkel például, amely azt fogja jelenteni, hogy 30 embert legalább meg kell mentenie a felvételhez; valamint "5%" legyen a második paraméter, amely pedig azt jelenti, hogy a 30-as értéktől 5%-kal térhet el negatív értelemben maximum. Vizsgálatkor első sorban ellenőrizze, hogy megfelelő attribútummal rendelkezik-e az adott típus tulajdonsága, másodsorban pedig figyelje, hogy az itt definiált érték (jelen esetben 30 és 5%) hogyan aránylik a konkrét entitás SavedQuantity tulajdonságának értékéhez. Ha nincs megfelelő

attribútummal rendelkező tulajdonsága az entitásnak, akkor dobjon `NoSavedLivesAttributeException` kivételt, ha pedig nem mentett még meg elég embert akkor `NotValidAvengerException` kivétel keletkezzen.

Hozzon létre két vagy több entitást, helyes (Avenger) és helytelen (NotAvenger) értékkel majd vizsgálja a kimeneteket.

A `Bosszúálló` osztályban a `Fight` metóduson helyezzen el egy `AvengerAttribute` nevű attribútumot egy konkrét értékkel. Készítsen egy `SendToFight` metódust a `Headquerters` osztályban amely egy object típust fogad és a metódus attribútumának értéke alapján kiírja a konzolra, hogy éppen hol harcol az entitás. Ehhez javasolt switch szerkezet használata az egyszerűség kedvéért.

4.5. Komplex feladatok III.

Egészítse ki az órai (`https://github.com/siposm/oktatas-hft/tree/master/LA-03-reflection/03-xml`) feladatot a következőkkel:

- a metódusok kiválogatásakor vizsgálja meg majd helyezze külön az int/-bool/double/stb. visszatérési értékű metódusokat az XML fájlba
- legyen külön szekció a <methods>-on belül az egyes visszatérési érték típusoknak (ezeket dinamikusan állapítsa meg)
- használjon XAttribute-ot a metódus nevének feltüntetésére
- az XML-be csak a metódusok nevét írja bele, abc szerinti csökkenő sorrendben
- egészítse ki ezt annyival, hogy a metódusok bemeneteinek típusát ezen belül egy szinttel beljebb írja ki az XML dokumentumba

Példa XML dokumentum:

```
...
<methods return="Boolean">
    <method name="Bemutatkozas">
        <param1>System.String</param1>
        <param2>Int32</param2>
        [...további paraméterek...]
    </method>
    <method name="Koszones">
        <param1>Boolean</param1>
        <param2>Double</param2>
        [...további paraméterek...]
    </method>
</methods>
...
<methods return="Int32">
```

```
        <method name="Futas">
            <param1>System.String</param1>
            <param2>System.String</param2>
            <param3>Double</param3>
            [...további paraméterek...]
        </method>
    </methods>
    ...
```

5. DLL

5.1. Feladat

Készítsen egy `LáncoltLista` osztályt amely generikus a T típusra nézve és hozzon létre beszúró, törlő és bejáró metódusokat. Készítsen ebből egy `lancoltlista.dll` fájlt.

5.2. Feladat

Készítsen egy `Élőlény` ősosztályt amely rendelkezik egy név tulajdonsággal. Származzon le ebből egy `Hallgató`, egy `Oktató` és egy `Kutya` osztállyal. Mind a három osztály rendelkezzen további tetszőleges tulajdonságokkal és metódusokkal. Állítson elő ezekből is egy darab `entitasok.dll` fájlt.

5.3. Feladat

Hozzon létre egy új Visual Studio projektet, amelyhez adja hozzá a létrehozott két dll állományt az 1. és 2. feladatból. Tesztelésként hozzon létre egy-egy példányt és írassa ki azok tulajdonságait. Hozzon létre a listából egy példányt, adjon hozzá elemeket és járja be a listát. Származzon le a `Kutya` osztályból és adjon hozzá egy tetszőleges metódust az új osztályhoz.

5.4. Feladat

Készítsen egy `IJátszik` és egy `IBemutatkozik` interfészt. Előbbi rendelkezzen egy double visszatérési értékű `Játszik` metódussal, utóbbi pedig egy string visszatérési értékű `Bemutatkozás` metódussal. Utóbbi esetén terjessze ki az interfészt `IHallgatóBemutatkozik` interfészre, amely egy `Hallgató-Bemutatkozás` metódussal rendelkezzen. Hozzon létre egy újabb dll állományt `interfeszek.dll` néven a fentebb végrehajtott interfészekből. A dll-t adja hozzá egy projekthez és készítsen az interfészek alá való osztályokat. Adja hozzá a projekthez a listás dll-t is, és tároljon el a listában IBemutatkozik intefész referenciával rendelkező elemeket.

Reflexió segítségével futásidőben vizsgálja meg, hogy a listához hozzáadandó elem rendelkezik-e "HallgatoBemutatkozas" nevű metódussal. Amennyiben igen, adja hozzá a listához, egyéb esetben dobjon saját kivételt.

Egészítse ki a létrehozott `interfeszek.dll`-t oly módon, hogy helyezzen el egy `Deprecated` attribútumot a `HallgatóBemutatkozás` metóduson amelyet az interfész ír elő. Ennek megfelelően állítsa elő újra a dll-t és társítsa a projekthez. Reflexió segítségével vizsgálja meg, hogy a beszúrandó listaelem rendelkezik-e ilyen attribútumú metódussal. Ha igen adja hozzá a listához, egyéb esetben dobjon saját kivételt.

5.5. Feladat

Hozzon létre egy `IÖsszehasonlítás` interfészt, amely az `IComparable` interfészt terjeszti ki egy írható és olvasható tulajdonsággal, amely egy `Comparison<T>` és/vagy `Predicate<T>` delegált. Helyezzen el a tulajdonságon egy `ToInsert` attribútumot. Állítson elő ebből dll-t `mycompare_interface.dll` néven. Egy új projekthez adja hozzá ezt a dll állományt, valamint a korábbi láncolt listás dll állományt. A listából származzon le egy saját `FeltételesLáncoltLista` osztállyal, amelyet egészítsen ki egy `BeszúrásHA` nevű metódussal, amely akkor vesz fel elemet a listába, ha az a comparison vagy predicate delegált alapján megfelelőnek minősül. A `CompareTo` metódust valósítsa meg úgy, hogy két feltételes láncolt listát elemszámuk alapján tudjon összehasonlítani. Tesztelje az elkészítetteket.

6. Unit teszt

6.1. Feladat

Valósítsa meg a következő tesztelési feladatokat NUnit segítségével, valamint a következő osztályok segítségével.

Osztályok az entitásokhoz:

- `Hallgató` osztály, név, életkor és tárgyak listája tulajdonságokkal. Készítsen egy `TárgyFelvesz` metódust, amelyen keresztül (is) lehet kezelni a tárgyak listát. Ez a metódus kapja meg a szükséges adatokat ahhoz, hogy egy tárgyat létre tudjon hozni és a listához tudja adni. Amennyiben nem megfelelő paramétert kap (pl. a tárgy kreditértéke túl sok (pl. 80), vagy a tárgy neve nem megengedett karaktereket tartalmaz (pl. <, >, @ stb.)) akkor dobjon erről saját készítésű kivételt.
- `Tárgy` osztály, név, kreditérték és felvétel éve tulajdonsáokkal.
- `Egyetem` osztály, benne egy hallgató típusokat tartalmazó listával, hozzá írható és olvasható tulajdonsággal, valamint a következő metódusokkal:
  - `SaveHallgatók` metódus, amely a lista tartalmát egy txt vagy xml fájlba írja.
  - `LoadHallgatók` metódus, amely a txt vagy xml fájlból a hallgatókat beolvassa és a listához adja. (Hozzáadáskor érdemes figyelni, hogy ne legyenek duplikációk! Ideális megoldás az, ha a listát teljesen törli majd a fájlból nulláról tölti fel.)
  - `AddHallgató` metódus, amely egy hallgatót (amelyet paraméternek kap) felveszi a listába.
  - `AddHallgató` metódus, amely egy hallgató adatait paraméternek kapja (pl. név), abból létrehoz egy hallgatót és a listához adja. Figyeljen arra, hogy ha a hallgató neve üres string, akkor dobjon kivételt.
  - `SelectMax` metódus, amely a listából azt a hallgatót keresi ki és adja vissza, akinek a legtöbb tárgya van kreditérték alapján.
  - `SelectFrom` metódus, amely egy tárgy nevet és egy évszámot kap paraméternek és azon hallgatókat adja vissza egy új listában, akik ebben az évben vettek fel egy adott nevű tárgyat.

Osztályok teszteléshez:

- Hozzon létre egy tesztelésre alkalmas projektet, végezze el az inicializáláshoz szükséges lépéseket.
- Tesztelje le külön a Hallgató osztály tulajdonságait és metódusait.
- Tesztelje le külön a Tárgy osztály tulajdonságait.
- Tesztelje le az Egyetem osztály metódusait. Használjon `TestCase` attribútumot ahol hasznosnak érzi.

6.2. Feladat

Készítsen egy `Verem` adatszerkezetet amelyben a belső működést egy `List` biztosítja. Készítsen `Berak`, `Kivesz`, és `Megtekint` metódusokat a működésnek megfelelően. Limitálja a verem méretét egy tetszőleges számra, amelyet a konstruktorban állítson be a létrehozáskor. Több elem berakása esetén dobjon saját kivételt. Üres verem esetén történő kivételkor szintén dobjon saját kivételt. A verem legyen generikus. Készítsen tesztelő osztályt amelybe hozzon létre metódusokat, amelyek a verem működését tesztelik le:

- berakás(ok) után a megfelelő elem van-e legfelül illetve legalul?
- kivétel(ek) után a megfelelő elem van-e legfelül illetve legalul?
- megtekintés a megfelelő elemet szolgáltatja-e vissza?
- kivételek megfelelően dobónak-e?

6.3. Feladat

További gyakorlásnak térjen vissza valamelyik korábbi láncolt listás feladathoz, és készítsen hozzá tesztelő részt. Írjon egységteszteket a lista alapvető működéseire (elem berakása, elem törlése, elem keresése stb.).

7. Moq

Hozzon létre egy adatbázist a szokásos és tanult módon, töltse fel elemekkel, majd hozza létre hozzá a szükséges modelleket. Az adatbázisban legyenek `Autó` típusok eltárolva a következőkkel:

- üzembehelyezés éve (int)
- tulaj neve (string)
- tulaj életkora (int)
- futott km-ek száma (int)
- márka (enum, de ha azzal nem megy, akkor string)
- eladási ár (int).

A későbbiekben térjen vissza ehhez a részhez, és módosítsa úgy az adatbázist, hogy tulaj neve helyett konkrét `Tulaj` objektumokkal legyen képes dolgozni.

Hozza létre a következő többrétegű architektúrát:

- `Repository` osztály amely az adatbázist kezeli.
- `AutóController` osztály amely az üzleti logikát biztosítja, és valósítsa meg a következő metódusokat:
  - Metódus, amely visszaadja a három legkevesebbett futott autót egy listában.
  - Metódus, amely visszaadja azt az autót, ahol a tulaj 50 és 60 év közötti, az autó pedig 100 és 200 ezer km közötti futással rendelkezik - ha van ilyen. Ha nincs dobjon saját készítésű hibát.
  - Metódus, amely visszadja az optimális választást az autó korát és árát figyelembe véve.
  - Metódus, amely egy tulajdonost (vagy annak nevét) fogadja és az alapján megkeresi az ő autóit majd visszaadja azokat.
- `TestAutóController` osztály amely a teszteket tartalmazza
  - Itt hozzon létre `Moq` segítségével egy "dummy-adatbázist", és dolgozzon azzal a későbbiekben.
  - Készítsen egységteszteket amiben a fentebbi metódusok által előforduló lehetőségeket és hibákat lefedi. Igyekezzen a teszteket fókuszálni, egy-egy dolog tesztelésére egy-egy dedikált metódust alkalmazzon.

8. Process

8.1. Feladat

Egészítse ki az órai szöveges állomány feldolgozással foglalkozó feladatot a következőkkel. Hozzon létre egy szinkronizációs pontot amelyben az egyes processek kimeneteit olvassa be és egy darab szöveges állományba rögzítse azokat (a jelenlegi 4 db kimeneti állomány helyett). Építsen a feladatba egy optimalizációs lépést, amely a fájl nagysága alapján elsőnek indítja el az azt feldolgozó process-t, így gyorsítva a működést. Járjon el ugyan így az összes többi fájl esetében, tehát fájlnagyság alapján csökkenő sorrendben legyenek feldolgozva.

8.2. Feladat

Hozzon létre egy konzolos alkalmazást amelyben egy tömbből kell a maximum elemet kiválasztani. A teljes feladatot kisebb részekre bontva, a tömböt részekre osztjuk, ezekben külön keresünk maximum elemeket, majd a részenként előállt elemeket összevetjük egymással. Így megkapva, a teljes tömbből a maximális elemet. Állítsa elő az exe-t és futtassa azt egy másik alkalmazásból n db példányban. Az n számot az alapján válassza, hogy a tömböt hány darabra szeretné vágni (ez a rész legyen dinamikus hatással a teljes működésre). A tömb átadásának logikáját is találja ki. Tesztelje a működő programot 10, 1.000, 1.000.000 elemű tömbbel is, referenciaként soros feldolgozással is lemérheti a rendezés idejét (ehhez használhatja a beépített StopWatch osztályt a Diagnostics névtérből).

9. Thread

Lorem ipsum dolor sit amet

10. Task

Lorem ipsum dolor sit amet

11. Gyakorló zárthelyi I.

11.1. Feladat 0.

Hozzon létre egy dll generálására alkalmas projektet ConsoleLoggerLibrary néven. Hozzon létre benne egy osztály ConsoleLogger néven, benne egy void ConsoleLog metódust, amely egy object típust fogad. Az object-et írja ki a konzolra, a ToString metódus segítségével. Állítsa elő a dll állományt, majd azt adja hozzá a másik projekthez.

11.2. Feladat I.

Készítsen egy IWorker interfészt, ami az alábbi tulajdonságokat írja elő: string Name get; set; string Dept get; set; string Rank get; set; string Phone get; set; string Room get; set;

Készítsen egy Worker osztályt, amely valósítsa meg az IWorker interfészt. Egészítse ki ezt az osztályt egy string Email get; set; tulajdonsággal.

11.3. Feladat II.

Készítsen egy EmailValidatorAttribute osztályt, amely rendelkezzen egy char Character és egy int Length tulajdonsággal. Az osztályra tegyen megszorítást, hogy csak tulajdonságokra lehessen alkalmazni. Az előzőekben létrehozott (Worker) email tulajdonságra alkalmazza az attribútumot, értéknek adja meg a @ karaktert, valamint az 5 értéket, mint hosszt.

11.4. Feladat III.

Készítsen egy Validator osztályt, amelyben egy bool CheckEmail metódus segítségével vizsgálja meg, hogy a paraméternek kapott object rendelkezik-e Email tulajdonsággal, s amennyiben igen, úgy vizsgálja meg, hogy az attribútumban megadottaknak eleget tesz-e az értéke. Ha igen, igaz értékkel térjen vissza, egyéb esetben hamissal. A feladat elvégzését reflexióval valósítsa meg.

11.5. Feladat IV.

Készítsen egy Detector osztályt, benne egy void DetectWorkerClasses metódussal. A metódus futásidőben vizsgálja meg reflexió segítségével az aktuális osztályokat, ezek nevét kérje le fordított ABC sorrendbe rendezve egy tömbbe. Figyeljen, hogy csak azokat az osztályokat kérje le, amelyek az IWorker interfészt megvalósítják. A látványosabb teszteléshez készítsen a Worker osztályból három darab leszármazottat (FirstFloorWorker, SecondFloorWorker, ThirdFloorWorker). Ezekben további dolgok nem lesznek elhelyezve. A lekért típusokat írja ki XML fájlba (workerClasses.xml néven) figyelve az XML struktúra betartására. Írja ki az osztályokat nevét és a nevek hashkódját. A gyökérben attribútumként helyezze el, hogy hány osztály van.

11.6. Feladat V.

Hozzon létre egy Func delegáltat, amely egy fájl nevet kap bemenetnek (string) és egy IEnumerable<Worker> típussal tér vissza. A delegáltba hozzon létre egy névtelen függvényt, amelyben a kapott fájlt (workers.xml) beolvassa és egy List-et állít elő. Elegendő csak az email címeket kiválasztani a Worker objektumok előállításakor. Ezt követően hívja meg a delegáltat és az előállt kimenetet validálja le email címek alapján. Az eredményt a dll-ben kapott ConsoleLogger segítségével írassa ki.

11.7. Feladat VI.

Olvassa be a workers.xml állományt és hajtsa végre rajta a következő lekérdezéseket:

- 6.1. kérdezze le a tamásokat
- 6.2. kérdezze le, hogy az egyes intézetekben hányan dolgoznak, majd rendezze ezeket darabszám alapján csökkenő sorrendbe (a kimenet egy új névtelen osztályban legyen DEPT és COUNT mezőkkel)
- 6.3. kérdezze le a dolgozók neveit és email címeit, akik a BA épület 3. szintjén dolgoznak
- 6.4. kérdezze le, hogy átlagosan mennyi a kereset az egyes intézetekben (a kimenet egy új névtelen osztályban legyen DEPT és AVGSAL mezőkkel)

Az egyes részeket a Main-ból tesztelje is le.

12. Gyakorló zárthelyi II.

12.1. Feladat I.

Készítsen egy Food osztályt a következő tulajdonságokkal: string Name, int Price, int Qty Készítsen egy DiscountException osztályt, amely string-et kapjon konstruktor bemenetnek és adja tovább az ősnek. Készítsen egy GroceryShop osztályt, amiben helyezzen el egy Food listát tulajdonsággal, valamint a következő metódusokat:

- AddToCart: egy Food elemet a listához ad
- RemoveFromCart: index alapján törlünk a listából
- SelectFoodByCriteria: a bemeneti predicate delegált alapján (amelynek generikus típusa: Food típus) leválogatunk elemeket a listából és azokat visszaadjuk újabb listaként
- CalculateFinalSumPrice: bemenetnek egy int discountValue értéket kap, amely ha kisebb vagy egyenlő mint nulla, akkor dobjon saját készítésű kivételt; egyéb esetben számolja meg, hogy mennyi a kosár jelenlegi értéke (vegye figyelembe a terméket, annak árát és darabszámát) majd alkalmazza rá a paraméternek kapott kedvezményt és térjen vissza az eredménnyel 3 tizedesre kerekítve

12.2. Feladat II.

Készítsen egy új projektet amely tesztek írására legyen alkalmas. Hozzon létre egy Tests.cs állományt a projekten belül, és a következő metódusokat tesztelje a leírtak alapján:

- készítsen egy inicializáló metódust, amely minden teszt előtt automatikusan lefut; ebben hozzon létre egy GroceryShop példányt és töltse fel pár elemmel a bevásárlókosarat (javasolt a példány referenciáját osztály szintre kihelyezni)
- tesztelje le az AddToCart metódus működését, ellenőrizze, hogy a kosárba rakott elem tényleg belekerül-e
- tesztelje le a FinalSumPrice metódus működését, hogy megfelelő paraméter esetén valóban dob-e kivételt
- tesztelje le a FinalSumPrice metódust, hogy valóban jó értéket számol-e ki
- teszteljele a SelectFoodByCriteria metódust, hozzon létre egy predicate delegáltat amely az 1 darabszámú termékeket nézi

12.3. Feladat III.

Egészítse ki a GroceryShop osztályt egy PlaceOrder metódussal, amely Taskok segítségével fájlba írja a kosárban lévő termékek nevét / árát / darabszámát. A három tulajdonságra három külön „szálat" hozzon létre, tehát egy szál dolgozik a neveken, egy az árakon és egy a darabszámokon. A működés idejét szimulálja a Thread.Sleep segítségével (2 msp, 5 msp és 10 msp). Írja ki a konzolra, hogy a task elkezdődött vagy befejeződött. A taskokat szinkronizálja és continuation segítségével írja ki a konzolra, ha minden task befejezte futását. A fájlok neveinek ezeket használja: names.xml, prices.xml, quantities.xml

Miután a fájlok kiírásával végzett, continuation segítségével hozzon létre egy új task-ot, amelyben a weatherdata.xml állomány tartalmát feldolgozva, ki kell számolni az átlagos hőmérsékletet és ezt kiírni a konzolra.

Biztosítsa, hogy a konzolra minden kiírásra kerüljön még mielőtt a fő programszál leállna. (Console.ReadXY nem megfelelő!)

A Main részből tesztelje le a párhuzamosságért felelős részt.