PROGRAMOZÁS 1

3. labor

Statikus osztályok és tagok (Static Classes and Static Class Members)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members

Konstruktorok (Constructors)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors

String osztály (String Class)
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netframework-4.8

DateTime struktúra (DateTime Struct)
https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=netframework-4.8

Generikus lista (List<T> Class)
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netframework-4.8

Nullable típusok (Nullable types)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/nullable-types/

IEnumerable Interfész (IEnumerable interfész)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/nullable-types/

yield kifejezés (yield)
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/yield

DirectoryInfo osztály (DirectoryInfo Class)
https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo?view=netframework-4.8

FileInfo osztály (FileInfo Class)
https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo?view=netframework-4.8

Directory osztály (Directory Class)
https://docs.microsoft.com/en-us/dotnet/api/system.io.directory?view=netframework-4.8

Készítette: Dr. Katona József, PhD
egyetemi docens, tanszékvezető
email: katonaj@uniduna.hu

A gyakorlati anyag letölthető:
https://dufoffice365-my.sharepoint.com/:f:/g/personal/katonaj_uniduna_hu/En5DMMegfwhIh4xhJXvJuO0BVvPlU3FNz5bH5QQd1bPFg?e=ZByoQ6

A példaprogramok letölthetők:
https://dufoffice365-my.sharepoint.com/:f:/g/personal/katonaj_uniduna_hu/EoLZVO8EvUhCjWcZCxxEVeYB2Iwot6K0qbEBWMSbXFc_gA?e=AoWlRY

TARTALOMJEGYZÉK

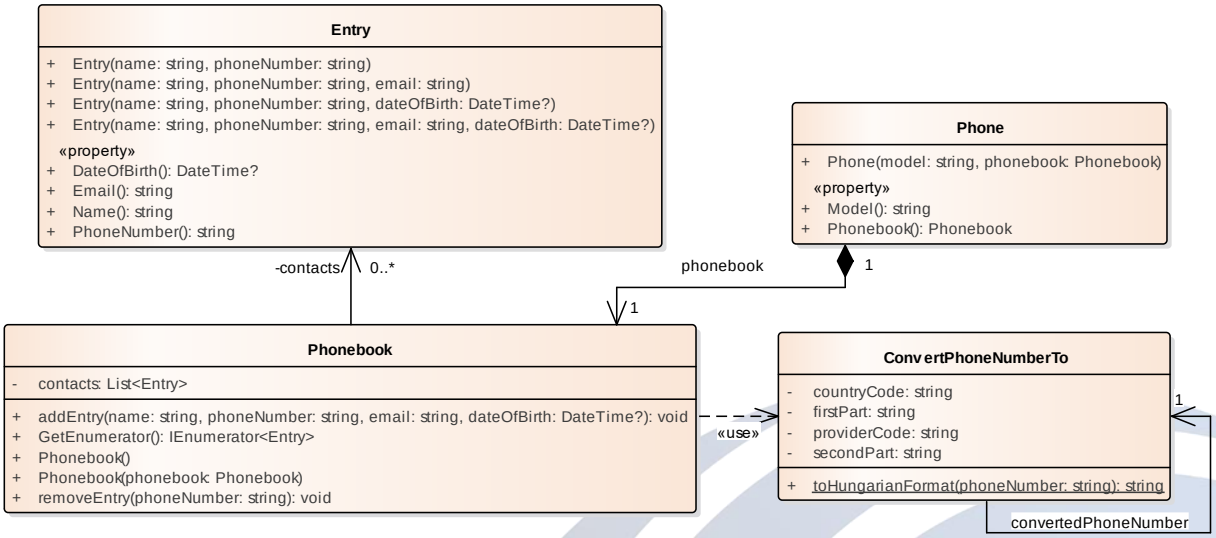
1. PhoneApp: „Telefonkönyv”

1.1. A feladat ismertetése

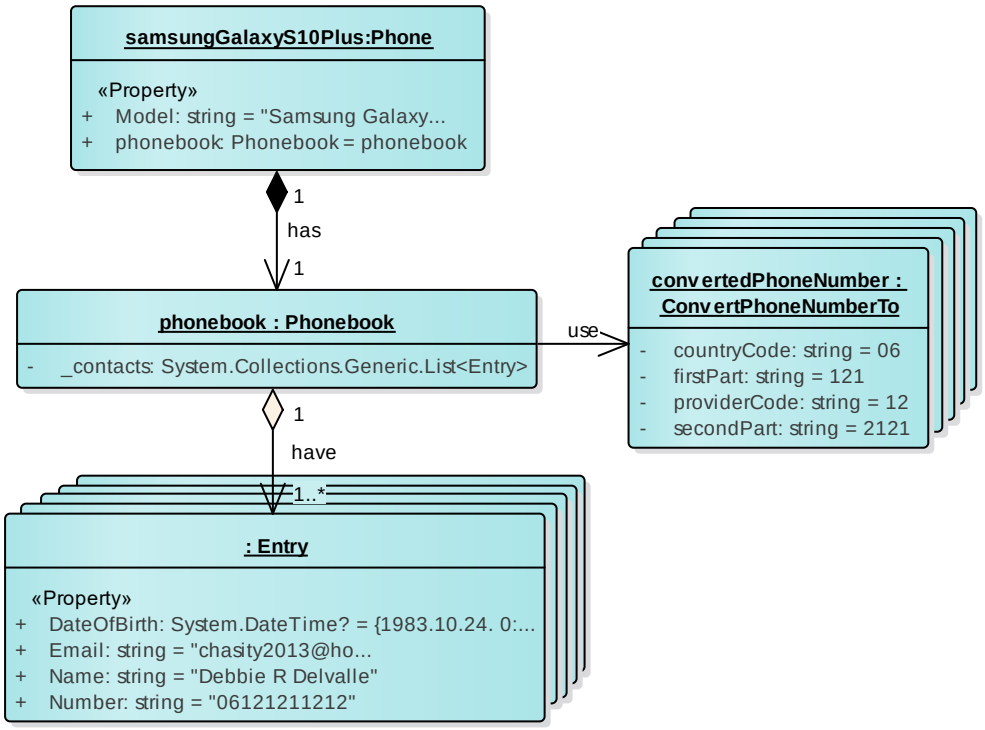
Tervezzen meg és készítsen el egy olyan alkalmazást, amely telefonkönyveket menedzsel. Az alkalmazás legyen képes új kontaktokat felvenni és törölni. A felvétel során legalább a nevet és a telefonszámot meg kell adnunk, de opcionálisan adjunk lehetőséget az e-mail cím és/vagy a születési dátum eltárolására is. A telefonszámok esetében külön figyeljünk a {ccpp/fff-ssss} formátumra, ahol c=countryCode, p=providerCode, f=firstPart és s=secondPart. Törlés a telefonszámok alapján történhet. Az alkalmazás több telefon telefonkönyvét is legyen képes menedzselni, ezt illusztráljuk legalább két telefonpéldány segítségével, ahol az egyik készülék telefonkönyvének adatait a másik készülékre másoljuk, majd az első készülék esetében végezzünk adattörlést, a másodiknál rögzítsünk új adatokat.

1.2. A megvalósítandó alkalmazás egy lehetséges UML tervezete

1.2.1. Osztálydiagram



1.2.2. Objektumdiagram



1.3. A feladat megoldásának részletezése

A tervek alapján elkezdtük a megvalósítást. Összesen négy osztályt hoztunk létre; *ConvertPhoneNumberTo*, *Entry*, *Phone* és *Phonebook*.

A *ConvertPhoneNumberTo* osztályban, magán láthatósági szint mellett, adatmezőkben került tárolásra valamely telefonszám országának és szolgáltatójának előhívószáma, valamint a két részre osztott hétjegyű telefonszám. Egy statikus tagfüggvény írja le a telefonszám átalakítást, amelyen belül egy lokális példányt és megfelelően felparaméterezett *Substring* függvény hívásokat használva a tárolni kívánt értékek átadódnak az egyes adattagoknak, majd a telefonszámot a *string.Format* segítségével a kívánt formában visszaadja. Mivel a metódus statikus, ezért a hívásához nincs szükség példányra.

Az *Entry* osztályban, nyilvános láthatósági szintek mellett, tulajdonságokban tárolódnak a telefonkönyvben rögzíteni kívánt adatok. A tulajdonságok értékeit kívülről, ellenőrzött módon lekérhetjük, de azok ellenőrzött módosítását csak az adott osztályon belül hajthatjuk végre, mivel a *set* előtt a *private* láthatósági szintet alkalmaztuk. Mivel a referenciatípusokkal ellentétben az értéktípusok alapértelmezetten nem vesznek fel *null* értéket, a *nullable* típus került alkalmazásra. A *DateTime* struktúra esetében a ? operátor segítségével a változó inicializáltalan marad. Az osztály további négy, nyilvános láthatósági szinttel rendelkező, eltérő paraméterlistát alkalmazó konstruktort (túlterhelés) tartalmaz, így megvalósítva az e-mail címek és a születési dátumok tárolásának opcionális lehetőségét. A *this* kulcsszóval

konstruktorból saját másik konstruktor hívása is történhet. A 2. és 3. konstruktor vissza lett vezetve az 1. konstruktor, amíg a 4. konstruktor a 2. konstruktor esetére.

A *Phone* osztályban, nyilvános láthatósági szint mellett, tulajdonságokban tárolódik a készülék neve és a *Phonebook* típust felhasználva a telefonkönyvének tartalma. A tulajdonság értékek módosítása csak az osztályon belül történhet.

A *Phonebook* osztály egy magán láthatósági szinttel rendelkező listát használ az *Entry* objektumainak tárolásához. A lista mellett további két konstruktort alkalmazva került elvégzésre a lista definiálása (paraméter nélküli konstruktor) és a *Phonebook* típusú objektum aktuális állapotának másolása (másoló konstruktor). Az *addEntry* metódus az *Entry* osztály megfelelő konstruktorát felhasználva a listához adja a paraméterekben kapott értékeket. A *removeEntry* metódus implementálásával lehetővé válik, hogy a listából egy megadott telefonszám alapján az elem eltávolítható legyen. A *GetEnumerator*-al (nem egy hagyományos metódust, hanem egy iterátort hoz létre) végig iterálva az osztály listáján kinyerésre kerültek az *Entry* típusú objektumok, ahol a *yield* azokat az értékeket mutatja, amelyeket az egyes iterációnak vissza kell adnia.

A *Program* osztályban található az alkalmazás belépési pontja a *Main* függvény, ahol a telefonkönyv adatokkal (öt darab) töltődik fel, oly módon, hogy nem minden esetben kerül megadásra egy személy e-mail címe és születési dátuma. Ezt követően egy telefont létrehozva, amelyben a telefonkönyvet elhelyezve, az adatok megjelenítődnek. A megjelenítés után a korábbi készülék telefonkönyvének aktuális állapota átmásolásra került egy új telefonra. Végül az első telefonon lévő adatok közül törölve egyet, amíg a második készülék esetében egy újabb bejegyzést készítve megjelenítésre kerültek a változások.

1.4. A feladat megoldása

1.4.1. ConvertPhoneNumberTo.cs

```
namespace PhoneApp
{
    public class ConvertPhoneNumberTo
    {
        private string countryCode, providerCode, firstPart, secondPart;

        public static string toHungarianFormat(string phoneNumber)
        {
            ConvertPhoneNumberTo convertedPhoneNumber = new ConvertPhoneNumberTo();
            convertedPhoneNumber.countryCode = phoneNumber.Substring(0,2);
            convertedPhoneNumber.providerCode = phoneNumber.Substring(2,2);
            convertedPhoneNumber.firstPart = phoneNumber.Substring(4,3);
            convertedPhoneNumber.secondPart = phoneNumber.Substring(7,4);

            return
string.Format($"{convertedPhoneNumber.countryCode}{convertedPhoneNumber.providerCode}/
{convertedPhoneNumber.firstPart}-{convertedPhoneNumber.secondPart}");
        }
```

```
    }
}
```

1.4.2. Entry.cs

```
using System;

namespace PhoneApp
{
    public class Entry
    {
        public string Name { get; private set; }
        public string PhoneNumber { get; private set; }
        public string Email { get; private set; }
        public DateTime? DateOfBirth { get; set; }

        public Entry(string name, string phoneNumber)
        {
            Name = name;
            PhoneNumber = phoneNumber;
        }
        public Entry(string name, string phoneNumber, string email)
            : this(name, phoneNumber)
        {
            Email = email;
        }
        public Entry(string name, string phoneNumber, DateTime? dateOfBirth)
            : this(name, phoneNumber)
        {
            DateOfBirth = dateOfBirth;
        }
        public Entry(string name, string phoneNumber, string email, DateTime?
dateOfBirth)
            : this(name, phoneNumber, email)
        {
            DateOfBirth = dateOfBirth;
        }
    }
}
```

1.4.3. Phone.cs

```
namespace PhoneApp
{
    public class Phone
    {
        public string Model { get; private set; }
        public Phonebook Phonebook { get; private set; }

        public Phone(string model, Phonebook phonebook)
        {
            Model = model;
            Phonebook = phonebook;
        }
    }
}
```

1.4.4. Phonebook.cs

```
using System;
using System.Collections.Generic;
```

```
namespace PhoneApp
{
    public class Phonebook
    {
        private List<Entry> contacts;

        public Phonebook()
        {
            contacts = new List<Entry>();
        }
        public Phonebook(Phonebook phonebook)
        {
            contacts = new List<Entry>(phonebook.contacts);
        }
        public void addEntry(string name, string phoneNumber, string email = null,
DateTime? dateOfBirth = null)
        {
            bool isEmailNull = (email == null) ? true : false;
            bool isDateOfBirthNull = (dateOfBirth == null) ? true : false;

            if (isEmailNull && isDateOfBirthNull)
                contacts.Add(new Entry(name,
ConvertPhoneNumberTo.toHungarianFormat(phoneNumber)));
            else if (isEmailNull)
                contacts.Add(new Entry(name,
ConvertPhoneNumberTo.toHungarianFormat(phoneNumber), dateOfBirth));
            else if (isDateOfBirthNull)
                contacts.Add(new Entry(name,
ConvertPhoneNumberTo.toHungarianFormat(phoneNumber), email));
            else
                contacts.Add(new Entry(name,
ConvertPhoneNumberTo.toHungarianFormat(phoneNumber), email, dateOfBirth));
        }
        public void removeEntry(string phoneNumber)
        {
            for (int i = 0; i < contacts.Count; ++i)
                if (contacts[i].PhoneNumber ==
ConvertPhoneNumberTo.toHungarianFormat(phoneNumber))
                {
                    contacts.RemoveAt(i);
                    return;
                }
        }
        public IEnumerator<Entry> GetEnumerator()
        {
            foreach (Entry entry in contacts)
                yield return entry;
        }
    }
}
```

1.4.5. Program.cs

```
using System;

namespace PhoneApp
{
    public class Program
    {
        static void writePhonebookContentOf(Phone phone)
        {
```

```
            Console.WriteLine(phone.Model);
            foreach (Entry entry in phone.Phonebook)
                Console.WriteLine(string.Format("{0,20}\t{1,20}\t{2,20}\t{3,30}",
                    entry.Name, entry.PhoneNumber, entry.Email, entry.DateOfBirth));
        }
        static void Main(string[] args)
        {
            Phonebook phonebook = new Phonebook();
            phonebook.addEntry("Melanie B Montgomery", "06101234567");
            phonebook.addEntry("Cheryl D August", "06119998888",
"orville_mcke@gmail.com");
            phonebook.addEntry("Debbie R Delvalle", "06121211212",
"chasity2013@hotmail.com", new DateTime(1983, 10, 24));
            phonebook.addEntry("Dani R Taylor", "06135556666", "hayden1971@yahoo.com",
new DateTime(1971, 12, 10));
            phonebook.addEntry("Debbie R Delvalle", "06107839527");

            Phone samsungGalaxyS10Plus = new Phone("Samsung Galaxy S10+", phonebook);
            writePhonebookContentOf(samsungGalaxyS10Plus);

            Phonebook iPhonebook = new Phonebook(phonebook);

            Phone iPhoneXS = new Phone("iPhone XS", iPhonebook);
            writePhonebookContentOf(iPhoneXS);

            iPhonebook.addEntry("William P. Sanchez", "06102702857");

            Console.WriteLine();
            Console.WriteLine("06119998888 phone number is being removed from Samsung
Galaxy S10+...");

            phonebook.removeEntry("06119998888");

            Console.WriteLine();

            writePhonebookContentOf(samsungGalaxyS10Plus);
            writePhonebookContentOf(iPhoneXS);
        }
    }
}
```

2. Gyakorló feladatok

2.1. A feladat ismertetése (DirectoryOperationApp)

Tervezzen meg, majd implementáljon egy statikus osztályt, amely mappa műveleteket végez. Az első művelet a másolás, amelyet felhasználva az alkalmazás egy mappát képes átmásolni egy másik mappába. A művelet során a mappában található fájlok mellett az almappák és az azokban lévő fájlok másolása is történjen meg. A folyamat alatt a konzolon jelenjen meg, hogy melyik fájl honnan és hova kerül. A másolás mellett az osztály metódusai segítségével biztosítsunk lehetőséget a mappák és tartalmainak törlésére, átnevezésére, valamint tartalmainak kilistázására. Az elkészült osztály kerüljön kipróbálásra.