PROGRAMOZÁS 1.

1. labor

Referencia típusok (Reference types)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes#reference-types

Osztály létrehozása (Declaring Classes)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes#declaring-classes

Objektum létrehozása (Creating objects)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes#creating-objects

Adattagok (Fields)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/fields

Láthatósági szintek (Access Modifiers)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers

Tagfüggvények (Methods)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods

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

Referencia mutató (this)
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/this

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

snippetek (C# code snippets)
https://docs.microsoft.com/en-us/visualstudio/ide/visual-csharp-code-snippets?view=vs-2017

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/Es46cr8qK1tHg77xOj1rEYYBeAAefU6UordWxFBW8Wzrow?e=CDtaCB

A példaprogramok letölthetők:
https://dufoffice365-my.sharepoint.com/:f:/g/personal/katonaj_uniduna_hu/EjLmMhdtudFLsvNKcsP7oK8BA0RxUJxaHTrouGRH3rJSBA

TARTALOMJEGYZÉK

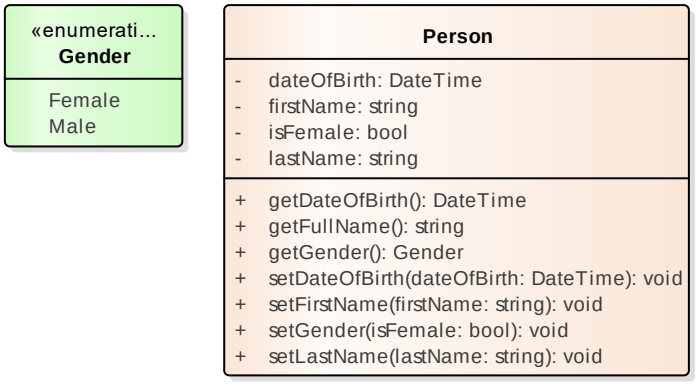
1. PersonApp: „Személy”

1.1. A feladat ismertetése

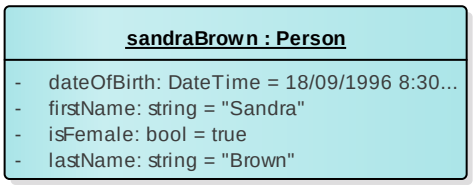
Készítsen egy *Person* osztályt, amely tárolja a személyek vezeték- és keresztnevét, születési dátumát és nemét. Az adattagokat rejtsük el más osztályok objektumai elől, azok értékeit kizárólag tagfüggvények segítségével lehessen lekérdezni és megváltoztatni. Az osztály megvalósítását követően készítsen egy objektumpéldányt is, amelyet felhasználva inicializálja az osztály adattagjait, majd írja ki a konzolra az eltárolt értékeket.

1.2. A megvalósítandó alkalmazás lehetséges UML tervezetei

1.2.1. Osztálydiagram

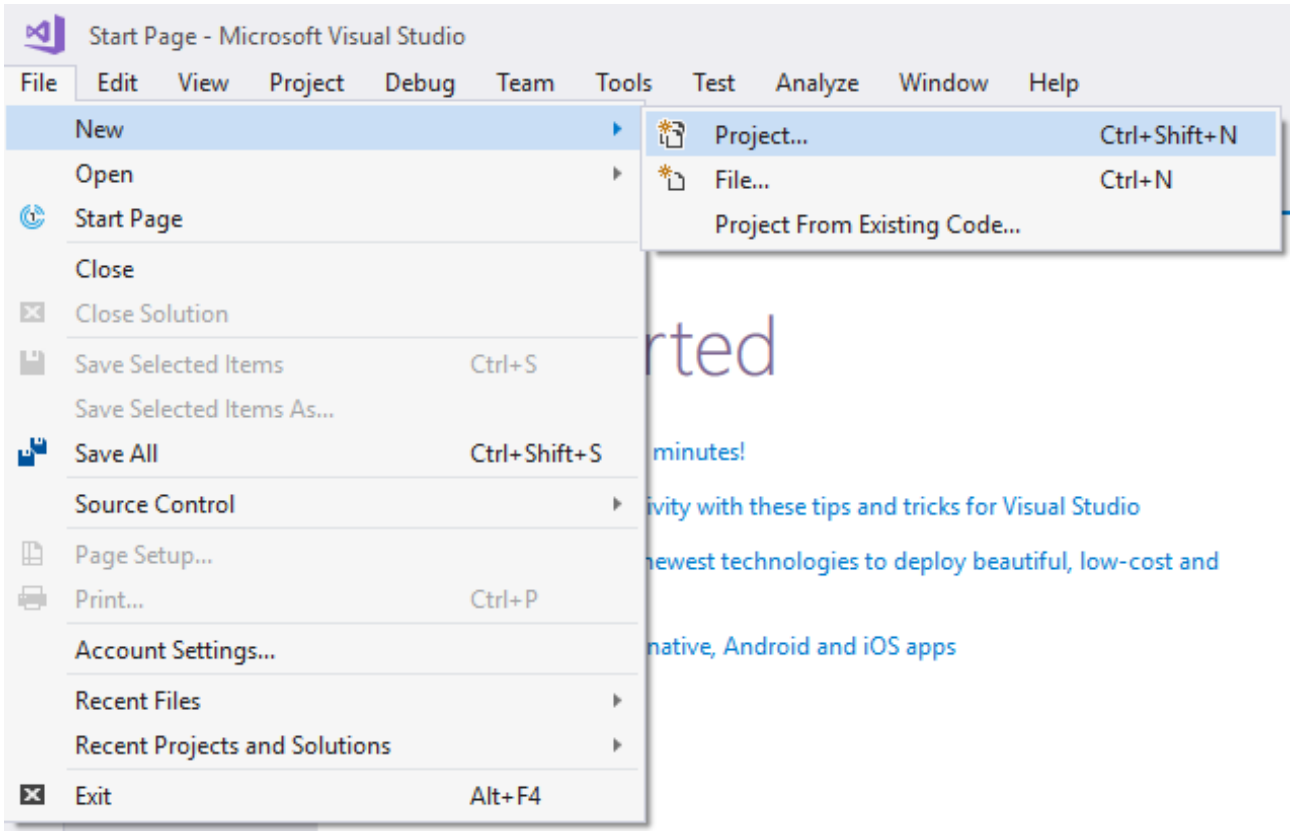


1.2.2. Objektumdiagram

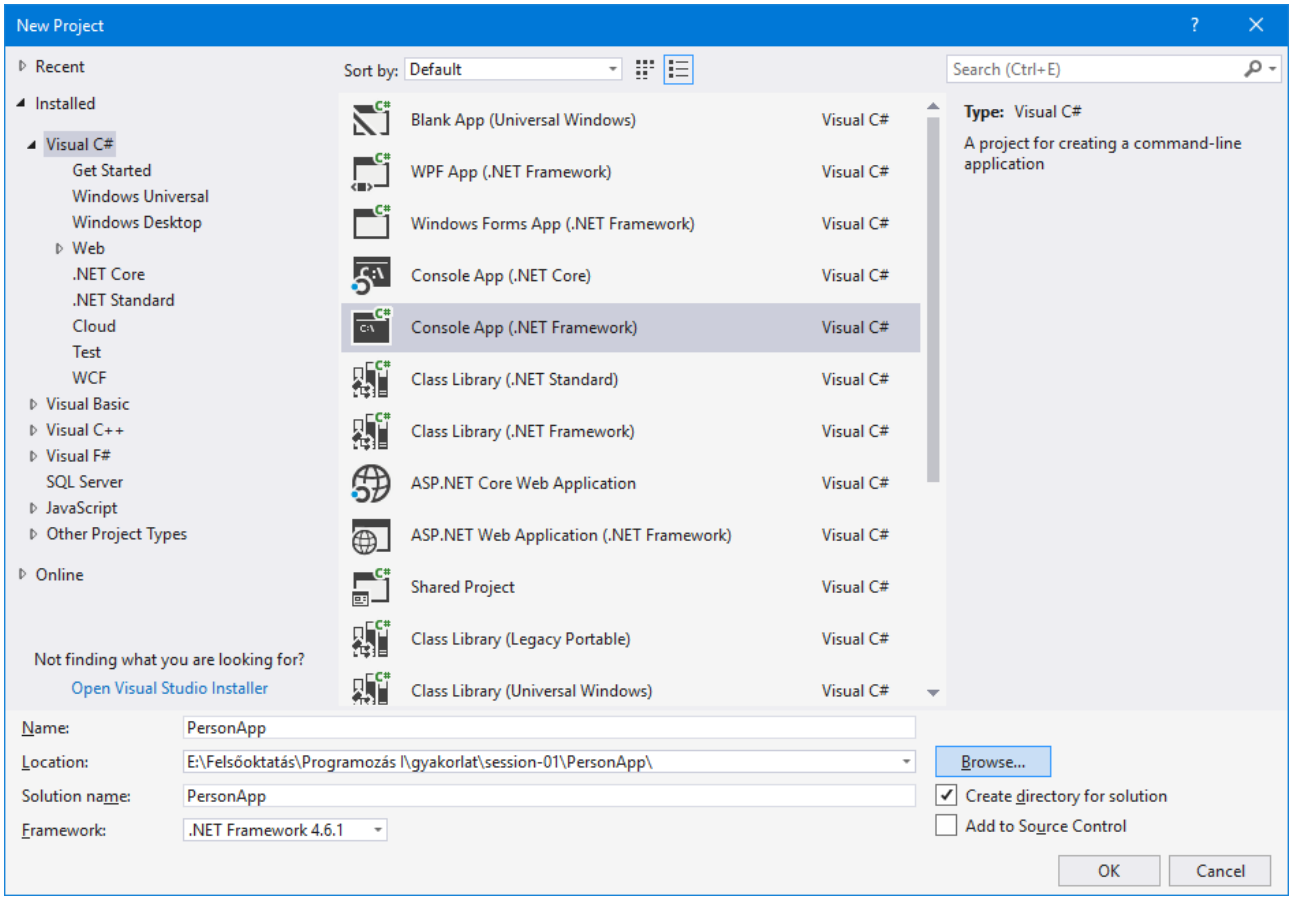


1.3. A feladat megoldásának lépései

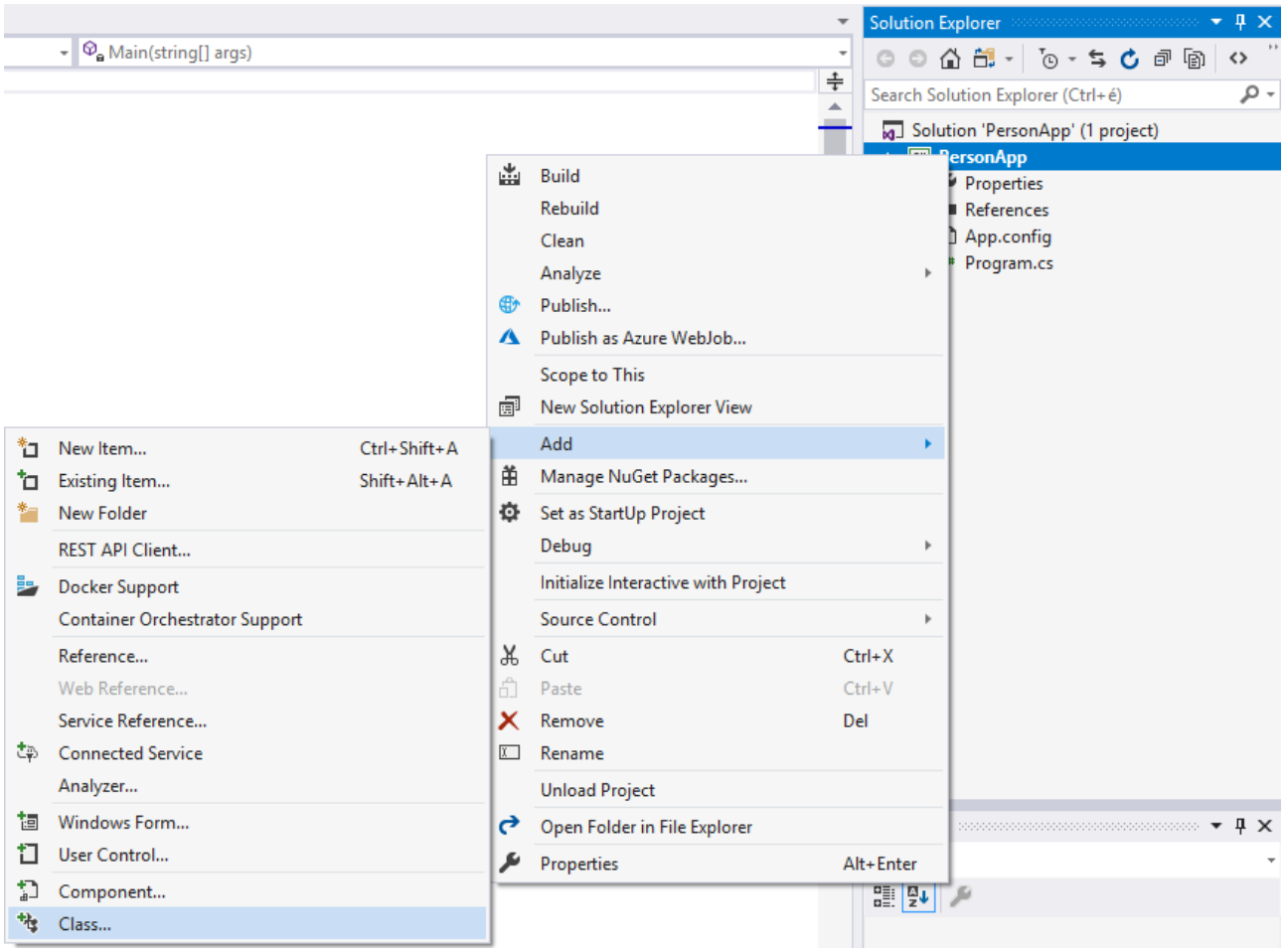
Indítsuk el a *Visual Studio* alkalmazásunkat, majd válasszuk ki a *File->New->Project* menüpontot és kattintsunk rá.



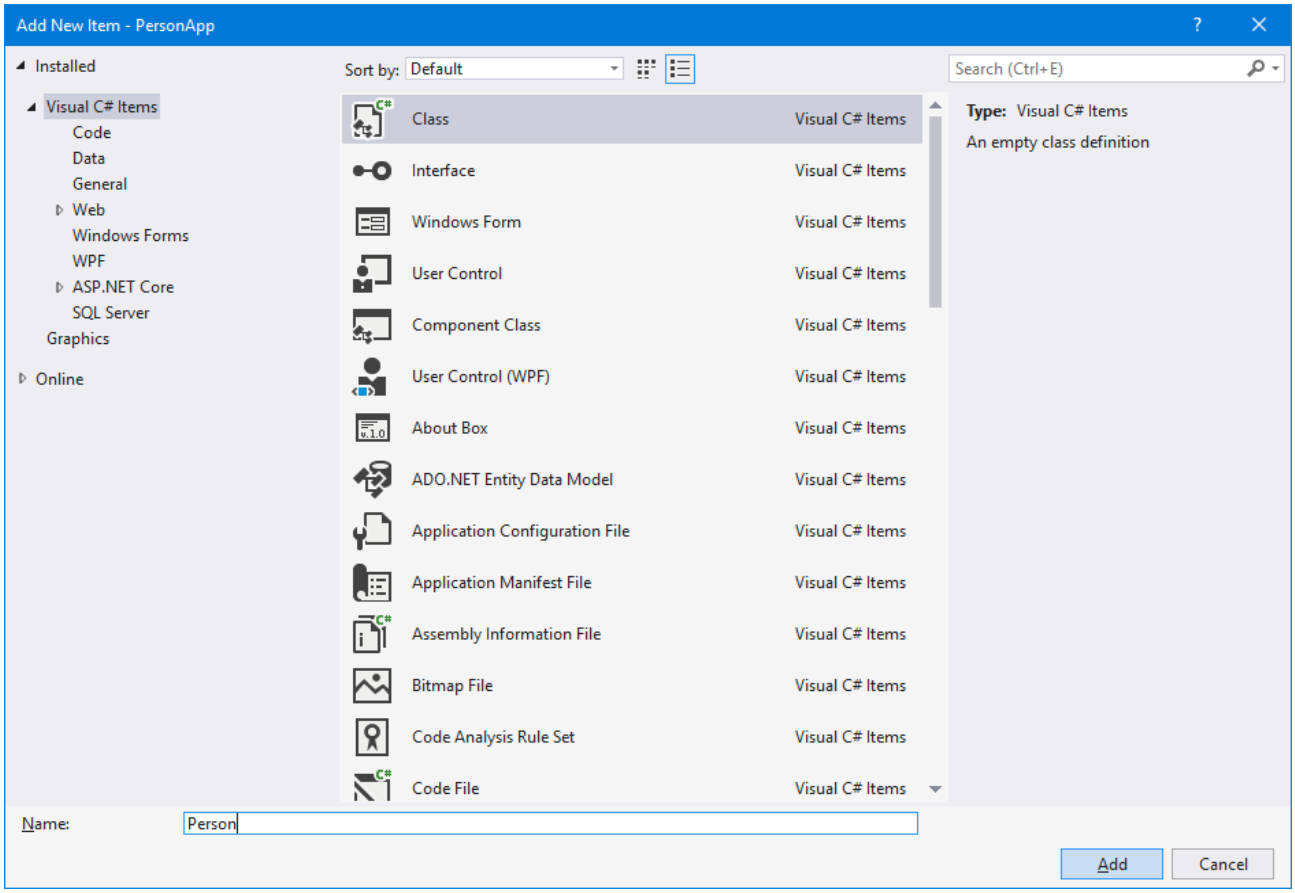
A kattintást követően a *New Project* felugró ablakban válasszuk ki a *Console App (.NET Framework)*-t, majd a projekt és a megoldás neve is legyen *PersonApp*. A mentési helyet válasszuk ki nekünk megfelelően, valamint a *Framework* verziót hagyjuk alapértelmezetten (ebben az esetben 4.6.1). Az adatok megadását követően kattintsunk az OK gombra.



A kattintást követően legenerálódik a projektünk. A létrejött projektben (*PersonApp*) hozzunk létre egy osztályt *Person* néven. Ehhez a *Solution Explorer*-ben kattintsunk jobb egér gombbal a projekt nevére, majd válasszuk ki az *Add* menüpont *Class* almenüpontját.



Kattintást követően a felugró ablakba írjuk be az osztály azonosítóját PascalCase típust használva. Az osztály nevének megadását követően kattintsunk az *Add* gombra.



A kattintást követően az osztály legenerálódik. Az UML terveket is felhasználva elkezdhetjük a megvalósítást. Az osztály négy darab magán (*private*) láthatósági szinttel ellátott adattagot tartalmaz; *firstName*, *lastName*, *dateOfBirth*, *isFemale*, amelyeknek típusa rendre; *string*, *string*, *DateTime* és *bool*, ahol a *DateTime* nem egy primitív típus, mint a *string* vagy a *bool*, hanem egy struktúra, értéktípus. Az osztály a négy adattagon felül további hét darab, nyilvános láthatósági szinttel rendelkező metódust tartalmaz:

- **void setFirstName(string firstName)**: A paraméterben kapott *firstName* értéket állítja be az osztály *firstName* adattagjának értékeként. A *this* mutató segítségével kiküszöbölhetjük a takarás problémáját.
- **void setLastName(string lastName)**: A paraméterben kapott *lastName* értéket állítja be az osztály *lastName* adattagjának értékeként. A *this* mutató segítségével itt is kiküszöbölhetjük a takarás problémáját.
- **string getFullName()**: A *firstName* és *lastName* adattagokban szereplő értékeket adja vissza egy *string* típusként, *string* konkatenációt alkalmazva.

- **void setGender(bool isFemale)**: A paraméterben kapott *isFemale* értéket állítja be az osztály *isFemale* adattagjának értékeként. A *this* mutató segítségével itt is kiküszöbölhetjük a takarás problémáját.
- **Gender getGender()**: Attól függően, hogy az *isFemale* logikai típusú adattag milyen értékű (*true* esetén nő, ellenkező esetben férfi), a bevezetés a programozásba című tárgyban megismert, ternális (három operandosú) kifejezést alkalmazva *Gender*-ként visszaadja a nem értékét. A *string* típus használata is helyes működést eredményez, azonban a *string*-ek helyett célravezetőbb a felsorolás típus használata. A fordító ugyanis egy *string* elírást/elgépelést nem tud ellenőrizni, ellenben, ha felsorolás típust használunk, akkor csak érvényes értéket fogad el.
- **void setDateOfBirth(DateTime dateOfBirth)**: A paraméterben kapott *dateOfBirth* értéket állítja be az osztály *dateOfBirth* adattagjának értékeként. A *this* mutató segítségével itt is kiküszöbölhetjük a takarás problémáját.
- **DateTime getDateOfBirth()**: A *dateOfBirth* adattag tárolt értékét adja vissza *DateTime* típusként.

A *Program* osztályban található a *Main* függvény, amely az alkalmazás belépési pontja. Ebben a metódusban az osztály egy példányát készítjük el. Mivel az osztályban nem készítettünk konstruktort, így a fordító létrehoz egy alapértelmezetett. Az objektum azonosítója *sandraBrown*, amelynek segítségével a nyilvános interfészen keresztül beállítjuk az adattagok értékeit. A *setter* függvények paraméterei az objektumdiagramon szereplő értékeket veszik fel. A *DateTime* struktúra konstruktorának 11 változata létezik, mi most egyet ezek közül felhasználva átadjuk az évet, hónapot, napot, órát, percet és másodpercet. A beállításokat követően az objektumon keresztül *getter* függvények segítségével megjelenítjük az objektum aktuális állapotát.

1.4. A feladat megoldása

1.4.1. Person.cs

```
using System;

namespace PersonApp
{
    public enum Gender { Female, Male }
    public class Person
    {
        private string firstName, lastName;
        private DateTime dateOfBirth;
        private bool isFemale;

        public void setFirstName(string firstName)
        {
            this.firstName = firstName;
        }

        public void setLastName(string lastName)
        {
```

```
            this.lastName = lastName;
        }

        public string getFullName()
        {
            return firstName + " " + lastName;
        }

        public void setGender(bool isFemale)
        {
            this.isFemale = isFemale;
        }

        public Gender getGender()
        {
            return isFemale ? Gender.Female : Gender.Male;
        }

        public void setDateOfBirth(DateTime dateOfBirth)
        {
            this.dateOfBirth = dateOfBirth;
        }

        public DateTime getDateOfBirth()
        {
            return dateOfBirth;
        }
    }
}
```

1.4.2. Program.cs

```
namespace PersonApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Person sandraBrown = new Person();
            sandraBrown.setFirstName("Sandra");
            sandraBrown.setLastName("Brown");
            sandraBrown.setGender(true);
            sandraBrown.setDateOfBirth(new DateTime(1996, 9, 18, 8, 30, 52));

            Console.WriteLine(sandraBrown.getFullName());
            Console.WriteLine(sandraBrown.getGender());
            Console.WriteLine(sandraBrown.getDateOfBirth());
        }
    }
}
```

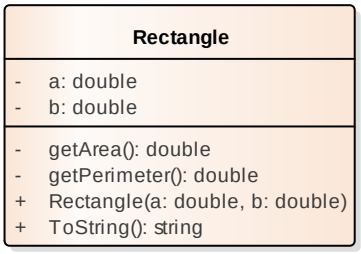
2. RectangleApp: „Téglalap kerülete és területe”

2.1. A feladat ismertetése

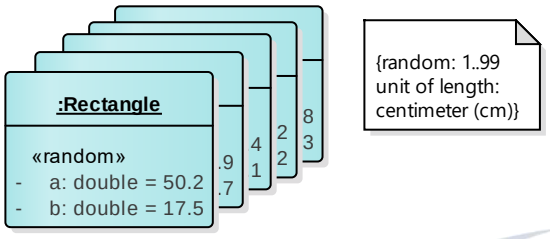
Készítsen egy *Rectangle* osztályt, amely tárolja egy téglalap két oldalhosszának méreteit, lebegőpontos típusú számként. Az adattagokat rejtsük el más osztályok objektumai elől, értékeiket kizárólag tagfüggvények segítségével lehessen lekérdezni és megváltoztatni. Az első objektum állapot létrehozásához paraméteres konstruktort használjunk. Az objektum legyen képes a tárolt adatokat felhasználva kiszámolni a téglalap kerületét és területét, valamint megjeleníteni a két oldalhossz értékét és a számított kerület, terület eredményeket. Az osztály megvalósítását követően készítsen öt objektumpéldányt is. Az objektum „születésekor” a téglalap két oldalának értéke véletlen szám legyen 1-100-ig.

2.2. A megvalósítandó alkalmazás lehetséges UML tervezetei

2.2.1. Osztálydiagram



2.2.2. Objektumdiagram



Az osztály létrehozását követően a terveket felhasználva elkezdhetjük a megvalósítást. Az osztály két darab magán (*private*) láthatósági szinttel ellátott adattagot tartalmaz; *a*, *b*, amelyek a téglalap oldalhosszait tárolják. Az osztály a két adattagon felül további négy darab, nyilvános és magán láthatósági szinttel rendelkező tagfüggvényeket tartalmaz:

- **Rectangle(double a, double b)**: Paraméteres konstruktor, amely a paraméterben kapott *a* és *b* értékeket állítja be az osztály *a* és *b* adattagjának értékeként. A *this* mutató segítségével kiküszöbölhetjük a takarás problémáját.
- **double getArea()**: Az *a* és *b* adattagokban tárolt értékeket felhasználva kiszámolja, majd visszaadja a téglalap területét.

- **double getPerimeter()**: Az *a* és *b* adattagokban tárolt értékeket felhasználva kiszámolja, majd visszaadja a téglalap kerületét.
- **override string ToString()**: Függvény felüldefiniálást alkalmazva megvalósítjuk a formázott kimenetet.

A *Program* osztályban található a *Main* függvény, amely az alkalmazás belépési pontja. Ebben a függvényben készítjük el az objektumtömböt, egy *for* ciklust felhasználva. A *Rectangle* osztály objektumain kívül, a véletlenszámok generálása érdekében, példányosítjuk a *Random* osztályt is. A ciklusmagon belül végezzük a tényleges példányosítást, ahol az osztály paraméterezett konstruktorát felhasználva véletlenszámokkal 1 és 99 között inicializáljuk az osztály adattagjait.

2.3. A feladat megoldása

2.3.1. Rectangle.cs

```
namespace RectangleApp
{
    class Rectangle
    {
        private double a, b;

        public Rectangle(double a, double b)
        {
            this.a = a;
            this.b = b;
        }
        private double getPerimeter()
        {
            return 2.0 * (a + b);
        }
        private double getArea()
        {
            return a * b;
        }
        public override string ToString()
        {
            return string.Format($"a={a:0.00}cm,
b={b:0.00}cm\n\rKerület={getPerimeter():0.00}cm\n\rTerület={getArea():0.00}cm^2\n\r");
        }
    }
}
```

2.3.2. Program.cs

```
using System;

namespace RectangleApp
{
    class Program
    {
        const int N = 5;
        static void Main(string[] args)
        {
            Random rnd = new Random();
            Rectangle[] rectangles = new Rectangle[N];
```

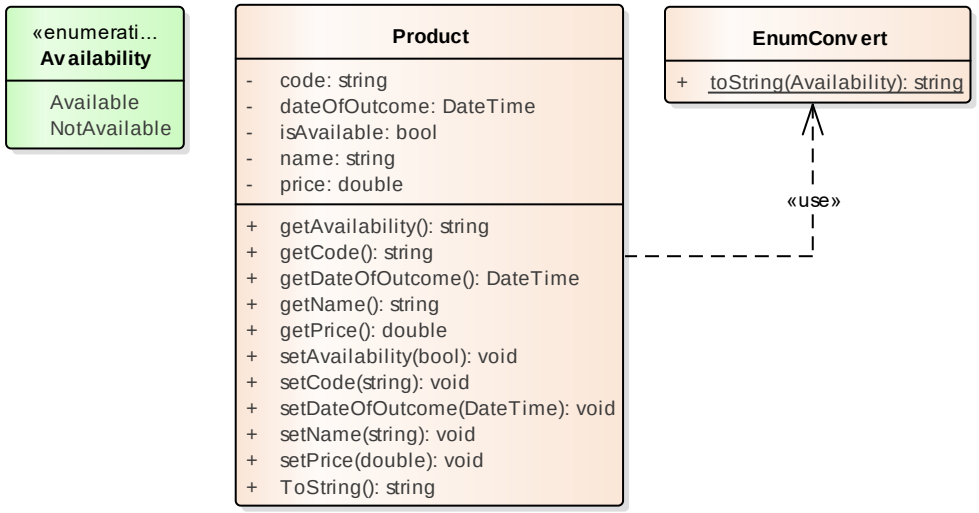
```
            for (int i = 0; i < N; ++i)
            {
                rectangles[i] = new Rectangle(1.0 + rnd.NextDouble() * 99.0,
                                              1.0 + rnd.NextDouble() * 99.0);
                Console.WriteLine(rectangles[i]);
            }
        }
    }
}
```

3. Gyakorló feladatok

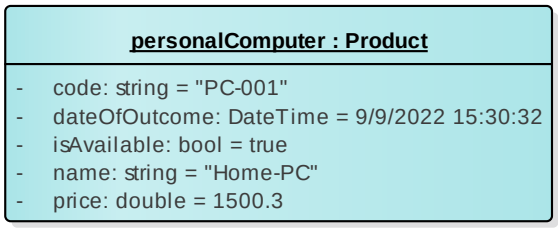
3.1. A feladat ismertetése (ProductApp)

Tervezzen meg és készítsen egy *Product* osztályt, amely tárolja egy termék kódját, nevét, megjelenésének dátumát, árát és a rendelkezésre állását. Az objektum állapotát tagfüggvények segítségével legyünk képesek megváltoztatni. A termék adatainak megjelenítését a *ToString* metódus felüldefiniálásával oldjuk meg.

3.1.1. Osztálydiagram



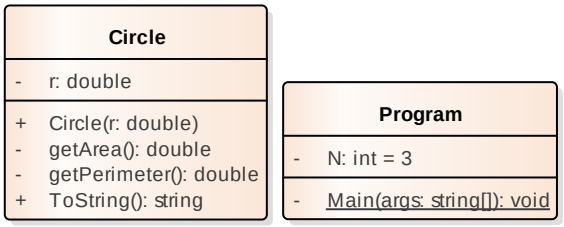
3.1.2. Objektumdiagram



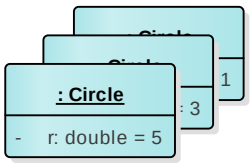
3.2. A feladat ismertetése (CircleApp)

Tervezzen meg és készítsen egy *Circle* osztályt, amely tárolja egy kör sugarát. Az adattagot rejtsük el más osztályok objektumai elől, azok értékeit kizárólag tagfüggvények segítségével lehessen lekérdezni és megváltoztatni. Az első objektum állapotok létrehozásához paraméteres konstruktort használjunk. Az objektum legyen képes a tárolt adatokat felhasználva kiszámolni a kör kerületét és területét, valamint megjeleníteni a sugár értékét és a számított kerület, terület eredményeket. Az osztály megvalósítását követően készítsen három objektumpéldányt is. Az objektum „születésekor" a kör sugarának értékét a felhasználótól kérjük be.

3.2.1. Osztálydiagram



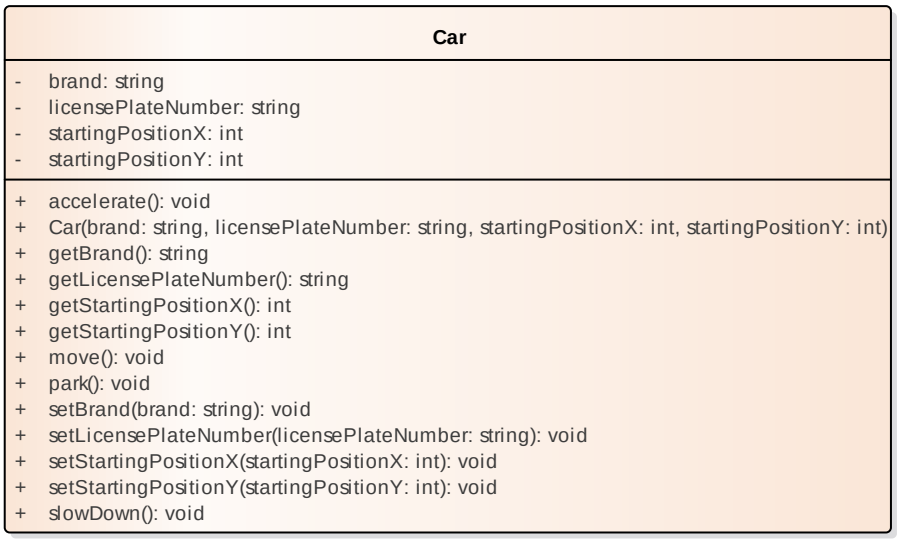
3.2.2. Objektumdiagram



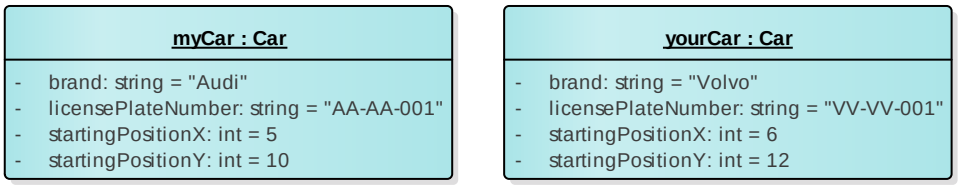
3.3. A feladat ismertetése (CarApp)

Tervezzen meg és készítsen egy *Car* osztályt, amely tárolja egy autó márkáját, rendszámát és a kiindulási pozícióját (annak x, y koordinátáját). Az adattagokat rejtsük el más osztályok objektumai elől, azok értékeit kizárólag tagfüggvények segítségével lehessen lekérdezni és megváltoztatni. Az objektum állapot létrehozásához paraméteres konstruktort használjunk. Későbbi fejlesztés céljából készítsen olyan tagfüggvényeket, amelyek az alábbi műveletek valósítják majd meg a későbbi fejlesztések során: mozgás, gyorsulás, lassulás és parkolás. Az osztály megvalósítását követően készítsen két objektumpéldányt is.

3.3.1. Osztálydiagram



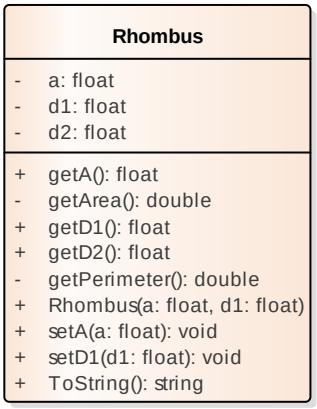
3.3.2. Objektumdiagram



3.4. A feladat ismertetése (RhombusApp)

Tervezzen meg és készítsen egy *Rhombus* osztályt, amely tárolja egy rombusz két átlójának és az oldalának a hosszát, lebegőpontos számként. Az adattagokat rejtsük el más osztályok objektumai elől, azok értékeit kizárólag tagfüggvények segítségével lehessen lekérdezni és megváltoztatni. Az első objektum állapot létrehozásához paraméteres konstruktort használjunk. Az objektum legyen képes a tárolt adatokat felhasználva kiszámolni a rombusz kerületét és területét, valamint megjeleníteni az oldalhossz értékét és a számított kerület, terület eredményeket. Az osztály megvalósítását követően készítsen egy objektumpéldányt is. Az objektum „születésekor" a rombusz oldalának értéke véletlen szám legyen 5-20 között.

3.4.1. Osztálydiagram



3.4.2. Objektumdiagram