PROGRAMOZÁS 1.

4. labor

Öröklődés (Inheritance)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/inheritance

Polimorfizmus (Polymorphism)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/polymorphism

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

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

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

virtuális módosító (virtual)
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual

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

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

Egymásba ágyazott típusok (Nested Types)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/nested-types

Felsorolás típus (enum)
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/enum

String-interpoláció (string interpolation)
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/EhQTB0Jv2ZNHlSwZHokoVB8BiZMg2cCW2kbIxnrBpraU_g?e=xTEsiW

A példaprogramok letölthetők:
https://dufoffice365-my.sharepoint.com/:f:/g/personal/katonaj_uniduna_hu/EtdjKbQDO3NJidu8UF1MJi0BTG4DqBblGcFx29wz16GtlA?e=ufA4r4

TARTALOMJEGYZÉK

1. AnimalHospitalApp: „Állatkórház” 3
1.1. A feladat ismertetése 3
1.2. A megvalósítandó alkalmazás egy lehetséges UML tervezete 4
1.2.1. Osztálydiagram 4
1.2.2. Objektumdiagram 5
1.3. A feladat megoldásának részletezése 5
1.4. A feladat megoldása 6
1.4.1. Animal.cs 6
1.4.2. Owner.cs 7
1.4.3. Dog.cs 7
1.4.4. Cat.cs 8
1.4.5. Duck.cs 9
1.4.6. Snake.cs 10
1.4.7. Program.cs 11
2. Gyakorló feladatok 14
2.1. A feladat ismertetése (AnimalHospitalApp) 14
2.2. A feladat ismertetése (ShapeApp) 14

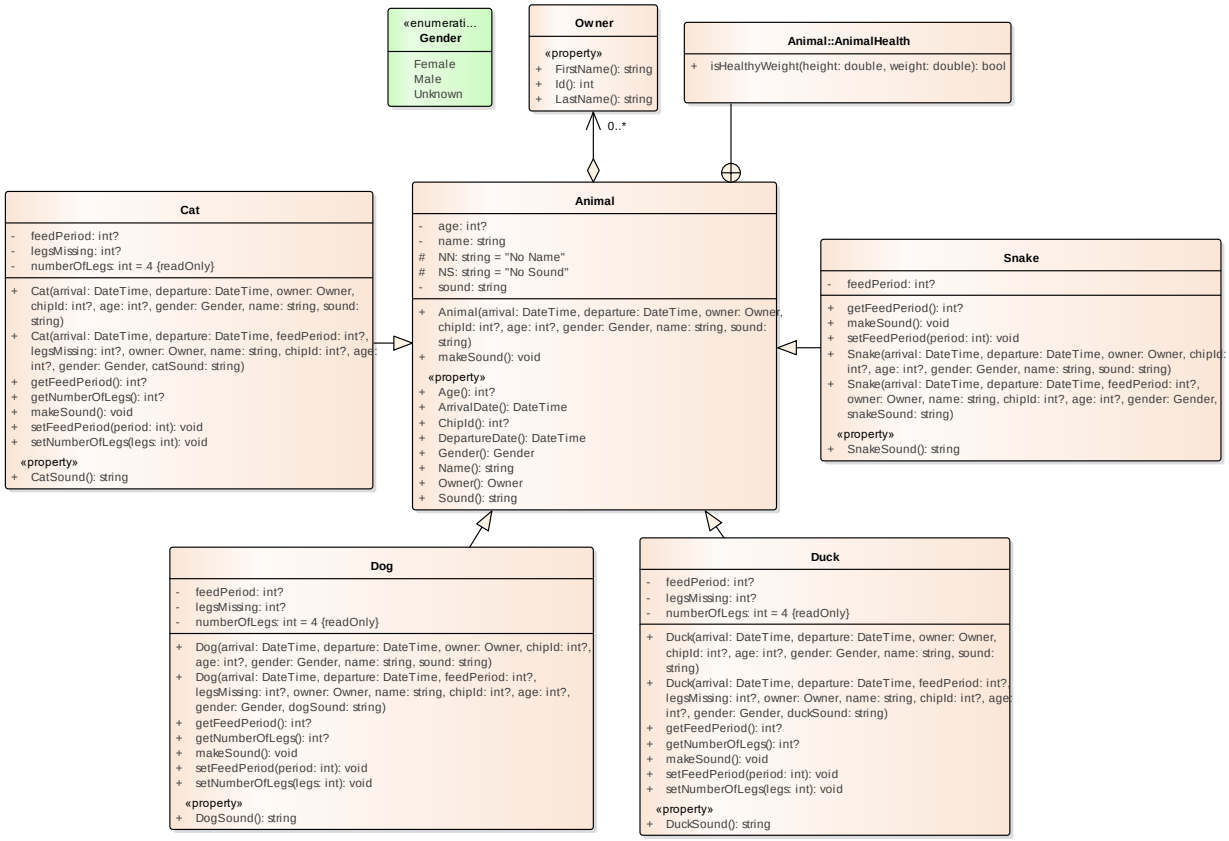
1. AnimalHospitalApp: „Állatkórház”

1.1. A feladat ismertetése

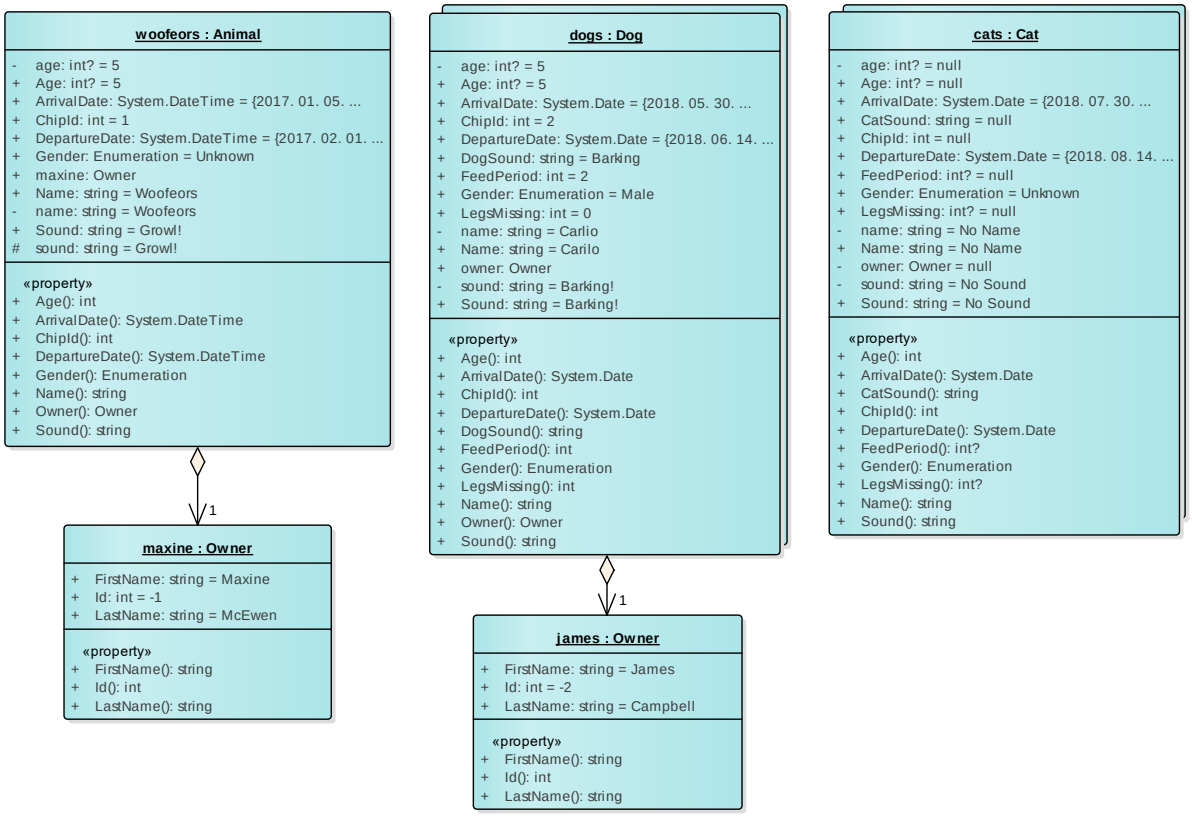
Tervezzen meg és készítsen el egy olyan pilot alkalmazást, amely egy állatkórház adatainak feldolgozását segítheti. A feladat, hogy egyfajta tárolást biztosítson a beteg állatokról a felhasználók számára. A kórház minden esetben nyilván szeretné tartani az állatok beérkezésének és távozásának időpontját. Opcionálisan tárolásra kerülhet az állat tulajdonosának neve, azonosító száma, életkora, neme {Female, Male és Unknown}, neve, valamint az olyan állatkórházban, ahol ez megvalósítható, hangfelvételt is rögzítenek az állatok hangjáról, amelyet opcionálisan tárolnak is későbbi feldolgozás céljából. Az állatokat különböző ütemben étkeztetik, ezért az egyes fajok esetében tárolásra kerül, hogy egy nap hányszor szükséges az etetés. Sajnálatos módon nem minden lábsérülés esetén tudják egy állat lábát megmenteni, így tároljuk el a ténylegesen meglévő lábmennyiséget is. A pilot alkalmazás egyelőre csak néhány faj adatainak felvételére biztosítson lehetőséget; kutya, macska, kacsa és kígyó. Az állatorvosok a gondozott állatok tápláltságának állapotát is meghatározzák oly módon, hogy veszik az adott állat testsúlyának és magasságának vagy hosszának a hányadosát. Ha ez a számított érték 0.18 és 0.27 közé esik, akkor őt kellően tápláltnak nyilvánítják.

1.2. A megvalósítandó alkalmazás egy lehetséges UML tervezete

1.2.1. Osztálydiagram



1.2.2. Objektumdiagram



1.3. A feladat megoldásának részletezése

A terveket felhasználva elkezdtük a megvalósítást. Összesen hét osztályt és egy felsorolás típust készítettünk; *Animal, Owner, Dog, Cat, Duck, Snake* osztályok és *Gender* felsorolás típus.

Az *Animal* egy ősosztály, amely az egyes fajok általánosításaként jött létre. Annak érdekében, hogy a feladatleírásban megfogalmazott kritériumoknak megfeleljen a konstruktor paraméterlistájában a szükséges helyeken alapértelmezett értékeket alkalmaztunk. A tulajdonságok több helyen is megvalósításra kerültek, így biztosítva az ellenőrzött írást és olvasást. Mivel alapvetően minden állat képes valamiféle hang kiadására, ezért készült egy *makeSound* azonosítójú virtuális metódus, amely megjeleníti az adott állat nevét és a rá jellemző kommunikációs hangot, illetve egy specializáció során könnyen felüldefiniálható. Az *Animal* tartalmaz egy beágyazott osztályt is, amelynek metódusa visszaadja, hogy az adott állat egészséges testsúly/magasság vagy hossz aránnyal rendelkezik-e. Végül megállapítható, hogy az *Animal* osztály aggregációs kapcsolatot mutat az *Owner* osztállyal. A kompozíció azért nem teljesülhet, mert léteznek a kórházban olyan állatok, amelyeknek nincs a bőrük alá ültetett azonosító chip és a gazdájukat sem ismerik.

Az *Owner* auto-tulajdonságokat használ a tulajdonos adatok tárolására.

A *Cat, Dog, Duck* és *Snake* osztályok az *Animal* osztály kiterjesztései. A fajokat leíró osztályok két konstruktort biztosítanak. Az első konstruktor esetében elég csak az adott faj egyedének beérkezésének és távozásának dátumát megadni. A nagyobb részletességet igénylő egyedek adatrögzítése a második konstruktorral történik, ahol minden adat (érkezési- és távozási idő, tulajdonos, azonosítószám, név, hang, etetési periódus, hiányzó lábak száma, életkor és nem) megadása kötelező. Minden faj esetében meghatározásra kerülhet a lábak száma (a fajra jellemző lábmennyiség és az adminisztráció során rögzített hiányzó lábak mennyiségének különbsége). A *makeSound* metódus megjeleníti az adott egyed nevét és ha rendelkezésre áll hangfelvétel, akkor az egyedre jellemző „beszédhang”-ot is.

A *Gender* felsorolás típus lényegében a két ismert nemen (*Female*, *Male*) kívül egy harmadikat (*Unknown*) is tárol, mivel előfordulhat, hogy egy adott egyed esetében a nem meghatározása nem lehetséges.

A *Program* osztályban található *Main* függvény tartalmazza a példányokat. Az állatkórház adatainak feltöltése változatos adatokkal történt. Egy fajból több példány is létrejött. A példányosítás során minden konstruktor meghívódott, végül a rögzített adatok táblázatos formában jelentek meg.

1.4. A feladat megoldása

1.4.1. Animal.cs

```
using System;
using System.Linq;

namespace AnimalHospitalApp
{
    public enum Gender { Female, Male, Unknown };
    public class Animal
    {
        protected const string NN = "No Name", NS = "No Sound";
        public int? ChipId { get; set; }
        public DateTime ArrivalDate { get; set; }
        public DateTime DepartureDate { get; set; }
        public Gender Gender { get; set; }
        public Owner Owner { get; set; }

        public Animal(DateTime arrival, DateTime departure, Owner owner = null, int?
chipId = null, int? age = null, Gender gender = Gender.Unknown, string name = NN,
string sound = NS)
        {
            ArrivalDate = arrival;
            DepartureDate = departure;
            Owner = owner;
            ChipId = chipId;
            this.age = age;
            Gender = gender;
            this.name = name;
            this.sound = sound;
        }

        /*public void makeSound()
        {
```

```
            Console.WriteLine($"{Name} says {Sound}");
        }*/

        public virtual void makeSound()
        {
            Console.WriteLine($"{Name} says {Sound}");
        }

        private string name;
        public string Name
        {
            get { return name; }
            set { name = value.Any(char.IsDigit) ? NN : value; }
        }

        private int? age;
        public int? Age
        {
            get { return age; }
            set { age = value < 0 ? null : value; }
        }

        private string sound;
        public string Sound
        {
            get { return sound; }
            set { sound = value.Length < 3 ? NS : value; }
        }

        public class AnimalHealth
        {
            public bool isHealthyWeight(double height, double weight)
            {
                double ratio = weight / height;
                return ratio >= 0.18 && ratio <= 0.27;
            }
        }
    }
}
```

1.4.2. Owner.cs

```
namespace AnimalHospitalApp
{
    public class Owner
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
    }
}
```

1.4.3. Dog.cs

```
using System;

namespace AnimalHospitalApp
{
    class Dog : Animal
    {
        private int? feedPeriod;
        private int? legsMissing;
        private readonly int numberOfLegs = 4;
```

```
        public string DogSound { get; set; }

        public Dog(DateTime arrival, DateTime departure, Owner owner = null, int?
chipId = null, int? age = null, Gender gender = Gender.Unknown,
            string name=NN, string sound=NS)
            : base(arrival, departure, owner, chipId, age, gender, name, sound) { }

        public Dog(DateTime arrival, DateTime departure, int? feedPeriod, int?
legsMissing, Owner owner = null, string name = NN, int? chipId = null,
            int? age = null, Gender gender = Gender.Unknown, string dogSound=NS)
            : base(arrival, departure, owner, chipId, age, gender, name, dogSound)
        {
            DogSound = dogSound;
            this.feedPeriod = feedPeriod;
            this.legsMissing = legsMissing;
        }

        public int? getFeedPeriod()
        {
            return feedPeriod != null ? feedPeriod : null;
        }

        public void setFeedPeriod(int period)
        {
            feedPeriod = period;
        }

        public int? getNumberOfLegs()
        {
            return legsMissing != null ? numberOfLegs - legsMissing : numberOfLegs;
        }

        public void setNumberOfLegs(int legs)
        {
            legsMissing = legs;
        }

        public override void makeSound()
        {
            Console.WriteLine($"{Name} says {DogSound}");
        }

        /*public new void makeSound()
        {
            Console.WriteLine($"{Name} says {Sound} and {DogSound}");
        }*/
    }
}
```

1.4.4. Cat.cs

```
using System;

namespace AnimalHospitalApp
{
    class Cat : Animal
    {
        private int? feedPeriod;
        private int? legsMissing;
        private readonly int numberOfLegs = 4;
        public string CatSound { get; set; }
```

```
        public Cat(DateTime arrival, DateTime departure, Owner owner = null, int?
chipId = null, int? age = null, Gender gender = Gender.Unknown,
            string name = NN, string sound = NS)
            : base(arrival, departure, owner, chipId, age, gender, name, sound) { }

        public Cat(DateTime arrival, DateTime departure, int? feedPeriod, int?
legsMissing, Owner owner = null, string name = NN, int? chipId = null,
            int? age = null, Gender gender = Gender.Unknown, string catSound = NS)
            : base(arrival, departure, owner, chipId, age, gender, name, catSound)
        {
            CatSound = catSound;
            this.feedPeriod = feedPeriod;
            this.legsMissing = legsMissing;
        }

        public int? getFeedPeriod()
        {
            return feedPeriod != null ? feedPeriod : null;
        }

        public void setFeedPeriod(int period)
        {
            feedPeriod = period;
        }

        public int? getNumberOfLegs()
        {
            return legsMissing != null ? numberOfLegs - legsMissing : numberOfLegs;
        }

        public void setNumberOfLegs(int legs)
        {
            legsMissing = legs;
        }

        public override void makeSound()
        {
            Console.WriteLine($"{Name} says {CatSound}");
        }

        /*public new void makeSound()
        {
            Console.WriteLine($"{Name} says {Sound} and {DogSound}");
        }*/
    }
}
```

1.4.5. Duck.cs

```
using System;

namespace AnimalHospitalApp
{
    class Duck : Animal
    {
        private int? feedPeriod;
        private int? legsMissing;
        private readonly int numberOfLegs = 2;
        public string DuckSound { get; set; }

        public Duck(DateTime arrival, DateTime departure, Owner owner, int? chipId =
null, int? age = null, Gender gender = Gender.Unknown,
```

```
            string name = NN, string sound = NS)
            : base(arrival, departure, owner, chipId, age, gender, name, sound) { }

        public Duck(DateTime arrival, DateTime departure, int? feedPeriod, int?
legsMissing, Owner owner = null, string name = NN, int? chipId = null,
            int? age = null, Gender gender = Gender.Unknown, string duckSound = NS)
            : base(arrival, departure, owner, chipId, age, gender, name, duckSound)
        {
            DuckSound = duckSound;
            this.feedPeriod = feedPeriod;
            this.legsMissing = legsMissing;
        }

        public int? getFeedPeriod()
        {
            return feedPeriod != null ? feedPeriod : null;
        }

        public void setFeedPeriod(int period)
        {
            feedPeriod = period;
        }

        public int? getNumberOfLegs()
        {
            return legsMissing != null ? numberOfLegs - legsMissing : numberOfLegs;
        }

        public void setNumberOfLegs(int legs)
        {
            legsMissing = legs;
        }

        public override void makeSound()
        {
            Console.WriteLine($"{Name} says {DuckSound}");
        }

        /*public new void makeSound()
        {
            Console.WriteLine($"{Name} says {Sound} and {DogSound}");
        }*/
    }
}
```

1.4.6. Snake.cs

```
using System;

namespace AnimalHospitalApp
{
    class Snake : Animal
    {
        private int? feedPeriod;
        public string SnakeSound { get; set; }

        public Snake(DateTime arrival, DateTime departure, Owner owner, int? chipId =
null, int? age = null, Gender gender = Gender.Unknown,
            string name = NN, string sound = NS)
            : base(arrival, departure, owner, chipId, age, gender, name, sound) { }
```

```
        public Snake(DateTime arrival, DateTime departure, int? feedPeriod, Owner
owner = null, string name = NN, int? chipId = null,
            int? age = null, Gender gender = Gender.Unknown, string snakeSound = NS)
            : base(arrival, departure, owner, chipId, age, gender, name, snakeSound)
        {
            SnakeSound = snakeSound;
            this.feedPeriod = feedPeriod;
        }

        public int? getFeedPeriod()
        {
            return feedPeriod != null ? feedPeriod : null;
        }

        public void setFeedPeriod(int period)
        {
            feedPeriod = period;
        }

        public override void makeSound()
        {
            Console.WriteLine($"{Name} says {SnakeSound}");
        }

        /*public new void makeSound()
        {
            Console.WriteLine($"{Name} says {Sound} and {DogSound}");
        }*/
    }
}
```

1.4.7. Program.cs

```
using System;

namespace AnimalHospitalApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random();

            Owner maxine = new Owner
            {
                Id = -001,
                FirstName = "Maxine",
                LastName = "McEwen"
            };
            Animal woofeors = new Animal(new DateTime(2017, 01, 05), new
DateTime(2017, 02, 01), maxine, 001)
            {
                Name = "Woofeors",
                Age = 5,
                Sound = "Growl!"
            };
            Animal.AnimalHealth getHealth = new Animal.AnimalHealth();

            Owner james = new Owner()
            {
                Id = -002,
                FirstName = "James",
```

```
                LastName = "Campbell"
            };
            Animal[] dogs = new Dog[2];
            dogs[0] = new Dog(new DateTime(2018, 05, 30), new DateTime(2018, 06, 14),
2, 0, james, "Carilo", 002, 5, Gender.Male, "Barking");

            Owner jones = new Owner()
            {
                Id = -003,
                FirstName = "Jones",
                LastName = "Brenda"
            };
            dogs[1] = new Dog(new DateTime(2016, 03, 10), new DateTime(2016, 04, 02),
2, 1, jones, "Derilla", 003, 6, Gender.Female, "Grrrr!");

            Owner ellis = new Owner()
            {
                Id = -004,
                FirstName = "Ellis",
                LastName = "Craig"
            };

            Owner noName = new Owner()
            {
                Id = 000,
                FirstName = "No",
                LastName = "Name"
            };

            Cat[] cats = new Cat[2];
            cats[0] = new Cat(new DateTime(2018, 07, 30), new DateTime(2018, 08, 14),
2, 0, noName);
            cats[1] = new Cat(new DateTime(2018, 09, 09), new DateTime(2018, 09, 21),
1, 1, ellis, "Salida", 004, 2, Gender.Female);

            Console.WriteLine("ANIMAL");
            woofeors.makeSound();

            Console.WriteLine("DOGS");
            foreach (Animal dog in dogs)
                dog.makeSound();

            //if(dog is Animal) dog.makeSound();
            //if(dog is Dog) dog.makeSound();


            Console.WriteLine("CATS");
            foreach (Animal cat in cats)
                cat.makeSound();

            Console.WriteLine();

            const string defaultTitleFormat = "{0,25} {1,15} {2,15} {3,10} {4,5}
{5,10} {6,10}",
                extendedTitleFormat = defaultTitleFormat + "{7,10} {8,10}";

            Console.WriteLine(string.Format(extendedTitleFormat, "Owner", "Arrival",
"Departure", "Name", "Age", "Gender", "Healthy?", "Legs", "Feeds"));
            Console.WriteLine(string.Format(defaultTitleFormat,
woofeors.Owner.FirstName + " " + woofeors.Owner.LastName,
woofeors.ArrivalDate.ToString("yyyy/MM/dd"),
```

```
                woofeors.DepartureDate.ToString("yyyy/MM/dd"), woofeors.Name,
woofeors.Age, woofeors.Gender, getHealth.isHealthyWeight(60.3, 15.2)));

            foreach (Dog dog in dogs)
                Console.WriteLine(string.Format(extendedTitleFormat,
dog.Owner.FirstName + " " + dog.Owner.LastName,
dog.ArrivalDate.ToString("yyyy/MM/dd"),
                    dog.DepartureDate.ToString("yyyy/MM/dd"), dog.Name, dog.Age,
dog.Gender, getHealth.isHealthyWeight(rnd.NextDouble() * 100, rnd.NextDouble() * 100),
                    dog.getNumberOfLegs(), dog.getFeedPeriod()));

            foreach (Cat cat in cats)
                Console.WriteLine(string.Format(extendedTitleFormat,
cat.Owner.FirstName + " " + cat.Owner.LastName,
cat.ArrivalDate.ToString("yyyy/MM/dd"),
                    cat.DepartureDate.ToString("yyyy/MM/dd"), cat.Name, cat.Age,
cat.Gender, getHealth.isHealthyWeight(rnd.NextDouble() * 100, rnd.NextDouble() * 100),
                    cat.getNumberOfLegs(), cat.getFeedPeriod()));
        }
    }
}
```

2. Gyakorló feladatok

2.1. A feladat ismertetése (AnimalHospitalApp)

Bővítsük tovább osztályokkal és funkciókkal az órai alkalmazást. Bővítsük az *Owner* osztályt és tároljuk el a tulajdonos néhány további adatát. A korábbi órákhoz hasonlóan implementálhatunk és alkalmazhatunk saját konverziós osztályokat is. Az órai alkalmazással ellentétben az adatokat a lehető legtöbb helyen véletlen generáljuk és ne kézzel adjuk meg. A létrehozott állapotokat mentsük ki egy szöveges fájlba.

2.2. A feladat ismertetése (ShapeApp)

Tervezzünk meg és implementáljunk egy olyan alkalmazást, amely kiszámolja a körök, ellipszisek, háromszögek, négyzetek és téglalapok kerületét és területét. A megoldás során használjunk öröklődést és polimorfizmust!