5. Paveldėjimas

Siekiami studijų rezultatai:

- pagrindiniai objektinio programavimo principai;
- objektinio programavimo principų taikymas programų kūrimui;
- uždavinio sprendimo algoritmo sudarymas ar žinomo algoritmo pritaikymas;
- sudaryto algoritmo realizavimas C# programavimo kalba;
- programų derinimas ir testavimas.

Susipažinsite su:

- bazinėmis ir išvestinėmis klasėmis;

5.1 Bazinės ir išvestinės klasės

Užduotis.
Klientas nusprendė išplėsti verslą ir pradėti registruoti ne tik šunis, bet ir kates. Išanalizavus reikalavimus, paaiškėjo, jog dauguma programos logikos skirtos dirbti su šunimis, tiks ir katėms. Skirsis kačių skiepų galiojimo trukmė – kates reikia skiepyti 2 kartus per metus.
Atsirado papildomų pageidavimų – reikia galimybės pažymėti agresyvius šunis (šio požymio nereikia katėms), taip pat galimybės rikiuoti gyvūnus pagal skirtingus požymius.

Pradiniai duomenys ir rezultatai.
Pasikeitė duomenų formatas. Pirmas stulpelis dabar saugo gyvūno tipą — įrašytas žodis DOG arba CAT. Šuns informacijos eilutė pasipildė požymiu „Agresyvus“ – duomenų faile įrašytas žodis „True“ arba „False“.

Pasiruošimas.

- Kopijuokite ir pridėkite į projektą visus *.cs* failus iš ankstesnio projekto

Pirmas žingsnis. Abstrakti klasė.

- Sukurkite bazinę **abstrakčią** klasę **Animal**, kurioje patalpinamos kačių ir šunų bendrinės savybės bei metodai.
- Bazinė klasė **Animal** turės šias katėms ir šunims bendras savybes:
  - `int ID`
  - `string Name`
  - `string Breed`
  - `DateTime BirthDate`
  - `Gender Gender`
  - `DateTime LastVaccinationDate`

- Bazinė klasė **Animal** taip pat turės išvestinę savybę **Age**. Išvestinę savybę **RequiresVaccination** paskelbkite abstrakčia.
- Bazinė klasė **Animal** turės konstruktorių, kuris inicializuos tik bendrus (aukščiau nurodytus) laukus.
- Kadangi tiek kačių, tiek šunų objektų palyginimas nesiskirs, **Animal** klasė turės perdengtus palyginimo metodus ir operatorius (kuriuos darėte ankstesniame darbe).

- Pertvarkymus atlikite vienu iš šių būdų:
  - kopijuoti ankstesniame darbe sukurtą **Dog.cs** failą į **Animal.cs**, įjungti jį į projektą ir pertvarkyti;
  - naujai sukurti **Animal** klasę, o iš **Dog** klasės perkelti metodus pertvarkymui;
  - naujai sukurti ir užrašyti **Animal** klasės laukus ir metodus.

```
namespace
```

```
{
    public abstract class Animal
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public string Breed { get; set; }
        public DateTime BirthDate { get; set; }
        public Gender Gender { get; set; }
        public DateTime LastVaccinationDate { get; set; }
        public int Age
        {
            get
            {
                DateTime today = DateTime.Today;
                int age = today.Year - this.BirthDate.Year;
                if (this.BirthDate.Date > today.AddYears(-age))
                {
                    age--;
                }
                return age;
            }
        }
        public abstract bool RequiresVaccination { get; }

        public Animal(int id, string name, string breed, DateTime birthDate, Gender gen
der)
        {
            this.ID = id;
            this.Name = name;
            this.Breed = breed;
            this.BirthDate = birthDate;
            this.Gender = gender;
        }

        public override bool Equals(object other)
        {
            return this.ID == ((Animal)other).ID;
        }

        public override int GetHashCode()
        {
            return this.ID.GetHashCode();
        }

        public int CompareTo(Animal other)
        {
            int result = this.Breed.CompareTo(other.Breed);
            if (result == 0)
            {
                return this.Gender.CompareTo(other.Gender);
            }
            return result;
        }
    }
}
```

Antras žingsnis. Pertvarkymas.

- Sukuriame arba pertvarkome šuns **Dog** klasę taip, kad joje liktų tik specifinės savybės ir metodai.
- Sukuriame/pertvarkome šuns klasės **konstruktorių** taip, kad bendrinės savybės būtų inicializuojamos kviečiant bazinės klasės konstruktorių, o naujai pridėta specifinė savybė (*Agressive*) būtų inicializuojama šuns klasėje (žr. konstruktoriaus aprašą).
- Savybės **LastVaccinationDate** realizacija liks ta pati.

```
namespace
{
    public class Dog : Animal
    {
        private const int VaccinationDuration = 1;

        public bool Aggresive { get; set; }

        public Dog(int id, string name, string breed, DateTime birthDate,
            Gender gender, bool aggresive) : base(id, name, breed, birthDate, gender)
        {
            this.Aggresive = aggresive;
        }

        public override bool RequiresVaccination
        {
            get
            {
                if (LastVaccinationDate.Equals(DateTime.MinValue))
                {
                    return true;
                }
                return LastVaccinationDate.AddYears(VaccinationDuration)
                         .CompareTo(DateTime.Now) < 0;
            }
        }
    }
}
```

Trečias žingsnis.

- Pagal šuns klasės pavyzdį sukurkite klasę **Cat**.
- Realizuokite savybę **LastVaccinationDate**.

```
namespace
{
    public class Cat : Animal
    {
        private const int VaccinationDurationMonths = 6;

        public Cat(int id, string name, string breed, DateTime birthDate,
            Gender gender) : base(id, name, breed, birthDate, gender)
        {

        }

        public override bool RequiresVaccination
        {
            get
            {
                if (this.LastVaccinationDate.Equals(DateTime.MinValue))
                {
                    return true;
                }
                return LastVaccinationDate.AddMonths(VaccinationDurationMonths)
                         .CompareTo(DateTime.Now) < 0;
```

```
            }
        }
    }
}
```

Ketvirtas žingsnis. Kitų klasių pertvarkymas.

Pertvarkykite kitas klases, jog jos dirbtų nebe su **Dog**, bet su **Animal** objektais.

- Pertvarkykite **DogsContainer** klasę – pervadinkite į **AnimalsContainer**, jos viduje esantį **Dog** masyvą pakeiskite **Animal** masyvu. Atitinkamai paredaguokite šios klasės metodų parametrus, taip pat metodų ir kintamųjų vardus.
- **Vaccination** savybę **DogID** pervadinkite į **AnimalID**. Atitinkamai pakeiskite šioje klasėje naudojamų kintamųjų vardus.
- Pertvarkykite **DogsRegister** klasę – pervadinkite į **Register**. Atitinkamai paredaguokite šios klasės metodų parametrus, taip pat metodų ir kintamųjų vardus.
- Klasėje **InOutUtils** metodą **ReadDogs** pervadinkite į **ReadAnimals** ir pritaikykite skaityti naujo formato duomenis. Taip pat šioje klasėje, kur reikia, paredaguokite metodų bei kintamųjų vardus.

```
public static AnimalsContainer ReadAnimals(string fileName)
{
    AnimalsContainer animals = new AnimalsContainer();
    string[] lines = File.ReadAllLines(fileName, Encoding.UTF8);
    foreach (string line in lines)
    {
        string[] values = line.Split(';');
        string type = values[0];
        int id = int.Parse(values[1]);
        string name = values[2];
        string breed = values[3];
        DateTime birthDate = DateTime.Parse(values[4]);

        Gender gender;
        Enum.TryParse(values[5], out gender); //tries to convert value to enum

        switch(type)
        {
            case "DOG":
                bool aggresive = bool.Parse(values[6]);
                Dog dog = new Dog(id, name, breed, birthDate, gender, aggresive);
                animals.Add(dog);
                break;
            case "CAT":
                Cat cat = new Cat(id, name, breed, birthDate, gender);
                animals.Add(cat);
                break;
            default:
                break;//unknown type
        }
    }
    return animals;
}
```

- Įvykdykite programą ir pasitikrinkite gautus rezultatus.
- Atkreipkite dėmesį – nėra spausdinama naujai pridėtos savybės **Aggresive** reikšmė.

Penktas žingsnis.

- Papildykite **Register** klasę metodu, kuris skirtas agresyvių šunų kiekiui skaičiuoti.

```
public int CountAggresiveDogs()
{
    int count = 0;
    for (int i = 0; i < this.all.Count; i++)
    {
        Animal animal = this.all.Get(i);
        if (animal is Dog && (animal as Dog).Aggresive)
        {
            count++;
        }
    }
    return count;
}
```

- Panaudokite šį metodą. Atspausdinkite, kiek agresyvių šunų yra registre.

Šeštas žingsnis. Palygintojo objektas.

Ankstesniuose laboratoriniuose darbuose, norėdami išrikiuoti konteineryje esančius **Dog** objektus, realizavote konteinerio metodą **Sort**, kuris lygino du objektus tarpusavyje naudodamasis **CompareTo** metodu.

- Pertvarkykite programą, sukurkite palygintojo klasę. Ši klasė turės vienintelį metodą – **Compare**, kuris mokės palyginti du objektus.

```
namespace
{
    public class AnimalsComparator
    {
        public virtual int Compare(Animal a, Animal b)
        {
            return a.CompareTo(b);
        }
    }
}
```

- Pertvarkykite **AnimalsContainer** metodą **Sort**, pakeiskite jį taip, jog priimtų palygintojo objektą per parametrus.

```
public void Sort(AnimalsComparator comparator)
{
    bool flag = true;
    while (flag)
    {
        flag = false;
        for (int i = 0; i < this.Count - 1; i++)
        {
            Animal a = this.animals[i];
            Animal b = this.animals[i + 1];
            if (comparator.Compare(a,b) > 0)
            {
                this.animals[i] = b;
                this.animals[i + 1] = a;
                flag = true;
            }
        }
    }
}
```

- Nenorime keisti ankstesnio programos veikimo. Papildykite klasę **AnimalsContainer** metodu **Sort**, be parametrų.

```
public void Sort()
{
    Sort(new AnimalsComparator());
}
```

- Įvykdykite programą ir pasitikrinkite gautus rezultatus. Viskas turėtų atrodyti taip pat.

Septintas žingsnis. Nauja palygintojo klasė.

- Sukuriame naują klasę **AnimalsComparatorByName**, kuri paveldės klasę **AnimalsComparator**.

```
namespace
{
    class AnimalsComparatorByName : AnimalsComparator
    {
        public override int Compare(Animal a, Animal b)
        {
            return a.Name.CompareTo(b.Name);
        }
    }
}
```

- Pakeiskite **Main** metode esantį kreipinį į **Sort**, paduokite naujai sukurtą palygintojo objektą.

```
allDogs.Sort(new AnimalsComparatorByName());
```

- Įvykdykite programą ir peržiūrėkite rezultatus.

savarankiško darbo užduotis

- Pakeiskite programą, padarykite jog darbo pabaigoje visų gyvūnų sąrašas būtų atspausdintas du kartus — pirmą sykį surikiuotas pagal gyvūno vardą, o jei vardas sutampa, tai pagal ID, antrą — pagal gimimo datą, o jei gimimo data sutampa, tai pagal ID.

savarankiško darbo užduotis

- Įmonė pradėjo registruoti ir jūrų kiaulytes. Jūrų kiaulytės neskiepijamos. Papildykite programą klase **GuineaPig** bei atitinkamai pakeiskite skaitymo, rašymo ir kitus metodus.

savarankiško darbo užduotis

- Sportas. Duota informacija apie krepšininkus: komandos pavadinimas, pavardė, vardas, gimimo data, žaistų rungtynių skaičius, įmestų taškų skaičius, atkovotų kamuolių skaičius, rezultatyvių perdavimų skaičius. Duota informacija apie futbolininkus: komandos pavadinimas, pavardė, vardas, gimimo data, žaistų rungtynių skaičius, įmuštų įvarčių skaičius, surinktų geltonų kortelių skaičius. Sukurkite klasę **Player** (savybės – komandos pavadinimas, vardas, pavardė, gimimo data, žaistų rungtynių skaičius, įmestų taškų (įmuštų įvarčių) skaičius), kurią paveldės klasės **Basketball** (savybės – atkovotų kamuolių skaičius, rezultatyvių perdavimų skaičius) ir **Football** (savybė – surinktų geltonų kortelių skaičius). Duota informacija apie komandas: komandos pavadinimas, miestas, komandos treneris, žaistų rungtynių skaičius. Atrinkite nurodyto miesto komandų žaidėjus, kurie žaidė visose komandos rungtynėse ir įmetė taškų (įmušė įvarčių) ne mažiau kaip vidurkis.