Dėstytojas

Edvinas Kesminas

Inheritance and Virtual methods

Data

Šiandien išmoksite

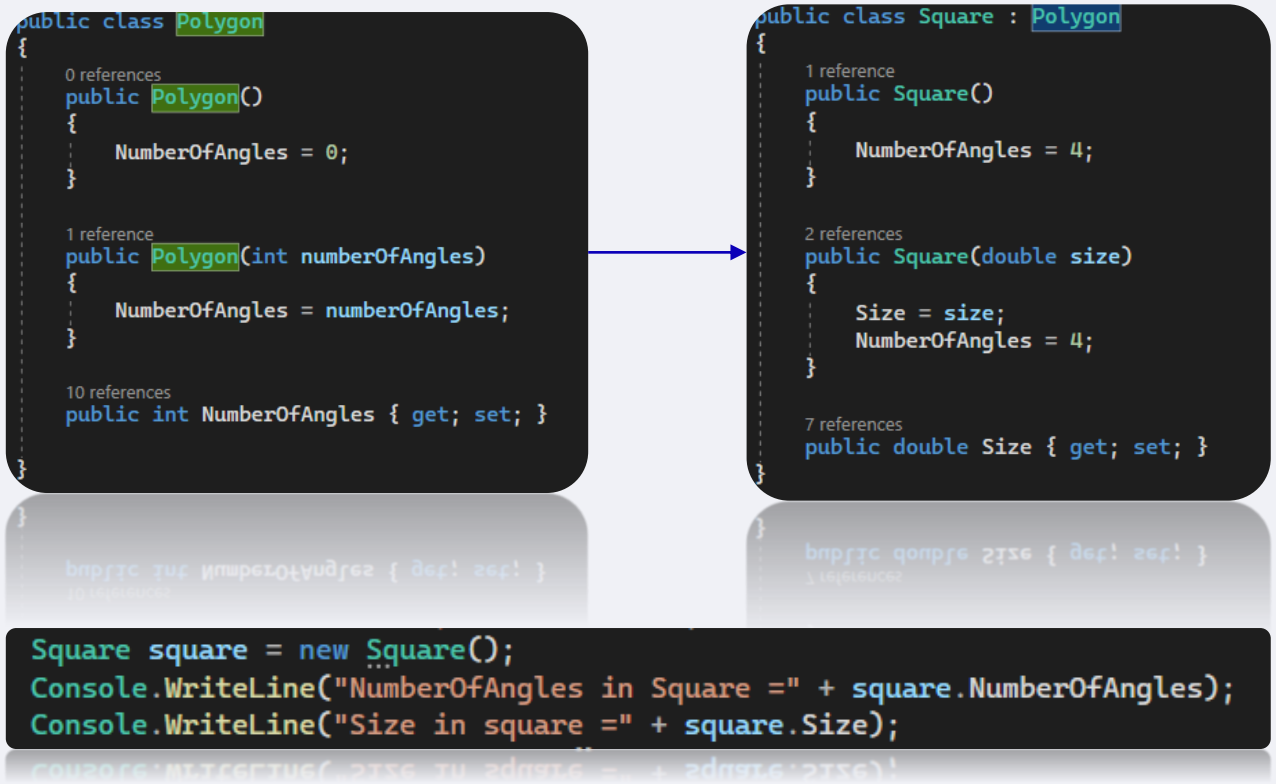
01 Inheritance

02 Virtual Methods

Inheritance

- Paveldėjimas yra svarbi objektinės programavimo paradigmos funkcija, leidžianti kurti hierarchijas ir dalintis savybėmis ir veiksmais tarp klasių
- Paveldėjimas yra vienas iš „Pillars of OOP“
- Paveldėjimas gali potencialiai padėti sumažinti dublikuoto kodo kiekį ir sukurti artimesnius klasių aprašus realiam gyvenimui

Inheritance



```
public class Polygon
{
    0 references
    public Polygon()
    {
        NumberOfAngles = 0;
    }

    1 reference
    public Polygon(int numberOfAngles)
    {
        NumberOfAngles = numberOfAngles;
    }

    10 references
    public int NumberOfAngles { get; set; }
}
```

```
public class Square : Polygon
{
    1 reference
    public Square()
    {
        NumberOfAngles = 4;
    }

    2 references
    public Square(double size)
    {
        Size = size;
        NumberOfAngles = 4;
    }

    7 references
    public double Size { get; set; }
}
```

```
Square square = new Square();
Console.WriteLine("NumberOfAngles in Square =" + square.NumberOfAngles);
Console.WriteLine("Size in square =" + square.Size);
```

- Paveldėjimas leidžia kurti hierarchijas, kuriose viena klasė paveldi savybes ir veiksmus iš kitos klasės.
- Tėvinė klasė (Polygon) yra aukštesnio lygio klasė, kurioje apibrėžiama bendra logika ir savybės, o vaikinės klasės(Square) yra žemesnio lygio klasės, kurios paveldi tėvinės klasės savybes ir gali papildomai turėti savo savybes ir veiksmus.
- Pavyzdžiui, galime turėti tėvinę klasę "Gyvūnas" su savybėmis "vardas" ir "amžius", o vaikinę klasę "Šuo" arba "Katė", kurios paveldi tėvinės klasės savybes ir gali turėti papildomas specifines savybes, pvz., "veislę"

Inheritance

- Paveldėjimas palaiko tiek lygių kiek mūsų klasių aprašams gali prireikti.
- Rekomenduojama nedaryti labai gilių paveldėjimų, nes kodo priežiūra tampa labai sudėtinga dėl klasių struktūros ir kodo atsekamumo.

```
public class Entity
{
    0 references
    public int Id { get; set; }
}

1 reference
public class Human : Entity
{
    0 references
    public string Name { get; set; }
    0 references
    public string Surname { get; set; }
    0 references
    public DateTime Birthday { get; set; }
}

0 references
public class Employee : Human
{
    0 references
    public double Salary { get; set; }
    0 references
    public DateTime EmploymentDate { get; set; }
}
```

Užduotis nr. 1

- Sukurkite bazinę klasę „Vehicle" su savybe „Speed". Parašykite vaikinę klasę „Car", kuri paveldi „Vehicle" klasę. Sukurkite objektą iš „Car" klasės ir nustatykite greitį. Išspausdinkite automobilio greitį. Papildykite programą sukurdami „Bike“ klasę ir atlikti tą patį ką atlikose su „Car“ klase.

- Sukurkite bazinę klasę „Employee" su savybėmis „Name" ir „Salary". Parašykite vaikinę klasę „Manager", kuri paveldi „Employee" klasę. Pridėkite savybę „Employees" ir nustatykite jos vertę. Išspausdinkite vadovo vardą, atlyginimą ir darbuotojų priklausančių vadovui skaičių.
- Plėskite "Employee" klasę iš ankstesnio pratimo, sukurdami vaikinę klasę „Programmer", kuri paveldi "Employee" klasę. Pridėkite savybę „ProgrammingLanguage" ir nustatykite jos vertę. Išspausdinkite programuotojo vardą, atlyginimą ir programavimo kalbą. Sukurkite „Manager“ metodą, kuris atspausdina visą informaciją apie programuotojus jo komandoje.

Užduotis nr. 2

- Sukurkite bazinę klasę „Product" su savybėmis „Name" ir „Price". Parašykite vaikinę klasę „Food", kuri paveldi "Product" klasę. Pridėkite savybę „ExpirationTime" ir nustatykite jos vertę. Išspausdinkite maisto pavadinimą, kainą ir galiojimo laiką.
- Plėskite "Product" klasę iš ankstesnio pratimo, sukurdami vaikinę klasę „Electronic", kuri paveldi "Product" klasę. Pridėkite savybę „Warranty" ir nustatykite jos vertę. Išspausdinkite elektronikos pavadinimą, kainą ir garantijos laiką.

- Aprašykite transporto priemonių nuomos sistemą. Jūsų sistema turėtų turėti klientų klases, transporto priemonių klases, skirtingų apmokėjimo būdų klases. Sukurkite trumpą scenarijų, kuriame galėtumėte kaip klientas rezervuoti sau norimą transporto priemonę ir priklausomai kokia tai transporto priemonė ji turėtų atitinkamai išmesti pranešimą kada yra panaudojami metodai „Drive()", „PrintMaxSpeed", „GetCapacity" ir „GetFuelEfficiency".

Virtual methods

- Virtualūs metodai yra svarbus objektinės programavimo paradigmos aspektas, leidžiantis perrašyti metodą vaikinėje klasėje, kuri paveldi tėvinės klasės metodą.
- Tai suteikia galimybę pakeisti metodų elgesį ir pritaikyti jį specifinėms vaikinės klasės reikmėms.
- Virtualūs metodai yra apibrėžti tėvinėje klasėje su raktazodžiu "virtual", o vaikinėje klasėje jie gali būti perrašyti su raktažodžiu "override".
- Tai leidžia dinamiškai pasirinkti tinkamą metodo versiją pagal objekto tipą, kuris yra saugomas tėvinės klasės rodyklėje.

```
public class Polygon
{
    0 references
    public Polygon()
    {
        NumberOfAngles = 0;
    }

    1 reference
    public Polygon(int numberOfAngles)
    {
        NumberOfAngles = numberOfAngles;
    }

    10 references
    public int NumberOfAngles { get; set; }

    4 references
    public virtual double GetPerimeter() //virt
    {
        return 0;
    }
}
```

```
public class Square : Polygon
{
    1 reference
    public Square()
    {
        NumberOfAngles = 4;
    }

    2 references
    public Square(double size)
    {
        Size = size;
        NumberOfAngles = 4;
    }

    7 references
    public double Size { get; set; }

    3 references
    public override double GetPerimeter()
    {
        return NumberOfAngles * 4;
    }
}
```

Užduotis nr. 3

- Sukurkite pagrindinę klasę pavadinimu "Transport" su virtualiu metodu "MeasureSpeed()". Įgyvendinkite šį metodą grąžinant standartinį greičio matavimo rezultatą. Sukurkite objektą iš "Transport" klasės ir iškvieskite metodo "MeasureSpeed()". Išplėskite "Transport" klasę iš ankstesnio pratimo, sukurdami paveldimą klasę pavadinimu "Car", kuri perrašo virtualų metodą "MeasureSpeed()". Įgyvendinkite šį metodą grąžinant automobilio greičio matavimo rezultatą. Sukurkite objektą iš "Car" klasės ir iškvieskite metodo "MeasureSpeed()".

- Sukurkite pagrindinę klasę pavadinimu "Employee" su virtualiu metodu "Greeting()". Įgyvendinkite šį metodą grąžinant standartinį pasisveikinimo pranešimą. Sukurkite objektą iš "Employee" klasės ir iškvieskite metodo "Greeting()".
- Išplėskite "Employee" klasę iš ankstesnio pratimo, sukurdami paveldimą klasę pavadinimu "Manager", kuri perrašo virtualų metodą "Greeting()". Įgyvendinkite šį metodą grąžinant vadovo pasisveikinimo pranešimą. Sukurkite objektą iš "Manager" klasės ir iškvieskite metodo "Greeting()".

Užduotis nr. 4

- Sukurkite pagrindinę klasę pavadinimu "Shape" su virtualiu metodu "Draw()". Įgyvendinkite šį metodą, kad būtų išvesta standartinė figūros informacija. Sukurkite išvestines klases pavadinimu "Circle" ir "Rectangle", kurios perrašo metodą "Draw()" ir išveda apskritimo ir stačiakampio informaciją atitinkamai ir juos atvaizduoja ekrane. Sukurkite objektus iš abiejų išvestinių klasių ir iškvieskite metodo "Draw()".

- Papildykite savo [Uzduotis 2.2] sprendimą ir padarykite, kad viskas vyktų naudojant virtual metodus. Papildykite programą taip, kad klientai galėtų būti kelių rūšių [„VIP“, „Standard“, „Eco“] ir kiekvienai iš šių rūšių turėtų būti priskirtos joms būdingos CalculateTotal() implementacijos. „VIP“ turėtų mažesnes kainas jei nuomoja transportą ilgiau nei 5 valandom, „Standard“ turėtų visada paprastą skaičiavimą, o „Eco“ turi turėti sumažintą tarfiją pirmoms 2 valandoms.

https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual

Naudinga informacija