

T06. Polimorfizmas

2 ak. val.

Temos klausimai

1. Statinis ir dinaminis polimorfizmas
2. Virtualūs metodai
3. Dinaminis susietumas
4. Abstrakti (**abstract**) klasė
5. **sealed** klasė
6. **Object** klasė

Polimorfizmo apibrėžimas

- Trečiasis OP bruožas (inkapsuliacija, paveldėjimas).
- **Polimorfizmas** – tai galimybė to paties tipo objektui skirtingai elgtis skirtingame kontekste.
- Galima skirti tokius polimorfizmo tipus:
 - operatorių užklojimas,
 - metodų užklojimas,
 - abstrakčiųjų metodų naudojimas.

Statinis polimorfizmas

Operatorių ir metodų užklojimas – tai **statinis polimorfizmas**.

Kokią prasmę suteikti operatoriui ar metodui, programą kompiliuojanti ir vykdanči sistema nusprendžia **kompiliavimo metu** (angl. compile time).

Prisiminkime: operatorių užklojimas

```
int a, b, c;  
a = 103;  
b = 406;  
c = a + b;           // c = 509
```

```
string sa, sb, sc;  
sa = „Erdvė“;  
sb = „laivis“;  
sc = sa + sb;       // sc = "Erdvėlaivis"
```

Prisiminkime: metodų užklojimas

```
public class klase
{
    private double[] A = new double [100];
    private int N { get; set; }

    double Max(int i1, int i2)
    {
        double max = A[i1];
        //...
        return max;
    }
    double Max(double d1, double d2)
    {
        double max = -999999.9;
        //...
        return max;
    }
}
```

Dinaminis polimorfizmas

- Polimorfizmas leidžia visą klasių šeimą, turinčią vieną bazinę klasę, apdoroti lyg visi objektai būtų bazinės klasės objektai.
- Tokią klasių šeimą lengva plėsti, papildant naujomis klasėmis, nes nereikia rūpintis naująja klase.
- Virtualūs bazinės ir išvestinių klasių metodai bei bazinės klasės objektų nuorodos sudaro **dinaminį polimorfizmą**.
- Kurį metodą (bazinės ar išvestinės klasės) vykdyti, programą kompiliuojanti ir vykdanti sistema nusprendžia programos **vykdymo metu** (angl. run time). Tai vadinama diniminiu susietumu arba užvėlintu susiejimu.
- Dažnai **dinaminis polimorfizmas** vadinamas tiesiog **polimorfizmu**.

Programos su paveldėjimu pavyzdys

Sukurkite programą (bazinę ir išvestines klases), kuri leistų aprašyti naminius gyvūnus bei jų elgiasį, pvz., skleidžiamus garsus.

Bazinė klasė: **NamGyvunas**

Išvestinės klasės: **Suo**

Katinas

Programos su paveldėjimu pavyzdys

```
class NamGyvunas : Object    // bazinė klasė
{
    public string vardas { get; private set; }

    public NamGyvunas(string vrd)    // Konstruktorius
    {
        this.vardas = vrd;
    }
    private string Balsas()
    {
        return "?????";
    }
    public void skleidziaGarsus()
    {
        Console.WriteLine("{0} skleidžia garsą: {1}",
                           vardas, Balsas());
    }
}
```

Programos su paveldėjimu pavyzdys

```
class Suo : NamGyvunas          // išvestinė klasė
{
    public Suo(string vardas) : base(vardas)
    {
    }
    private string Balsas()
    {
        return "Au! Au! Au!";
    }
}
```

Programos su paveldėjimu pavyzdys

```
class Katinas : NamGyvunas    // išvestinė klasė
{
    public Katinas(string vardas) : base(vardas)
    {
    }
    private string Balsas()
    {
        return "Miau! Miau! Miau!";
    }
}
```

Programos su paveldėjimu pavyzdys

...

```
NamGyvunas pikis = new NamGyvunas("Pikis");  
pikis.SkleidziaGarsus();
```

```
Suo dikas = new Suo("Dikas");  
dikas.SkleidziaGarsus();
```

```
Katinas murklys = new Katinas("Murklys");  
murklys.SkleidziaGarsus();
```

...

Ekrane matysite:

Problema: Visi gyvūnai – objektai,
nepriklausomai nuo jų tipo
skleidžia tuos pačius garsus

```
Pikis skleidžia garsą: ?????  
Dikas skleidžia garsą: ?????  
Murklys skleidžia garsą: ?????
```

Galimas problemos sprendimas

Kiekvieną išvestinę klasę (**Suo** ir **Katinas**) papildyti (**užkloti**) metodu:

```
public void skleidziaGarsus()  
{  
    // ...  
}
```

Programos su paveldėjimu pavyzdys

```
class Suo : NamGyvunas           // išvestinė klasė
{
    public Suo(string vardas) : base(vardas)
    {
    }
    private string Balsas()
    {
        return "Au! Au! Au!";
    }

    public void skleidziaGarsus()
    {
        Console.WriteLine("{0} skleidžia garsą: {1}",
                           vardas, Balsas());
    }
}
```

warning CS0108:
'Paveldejimas.Suo.SkleidziaGarsus()' hides inherited member 'Paveldejimas.NamGyvunas.SkleidziaGarsus()'. Use the new keyword if hiding was intended.

Programos su paveldėjimu pavyzdys

```
class Katinas : NamGyvunas    // išvestinė klasė
{
    public Katinas(string vardas) : base(vardas)
    {
    }
    private string Balsas()
    {
        return "Miau! Miau! Miau!";
    }
    public void skleidziaGarsus()
    {
        Console.WriteLine("{0} skleidžia garsą: {1}",
                           vardas, Balsas());
    }
}
```

warning CS0108:
'Paveldejimas.Katinas.SkleidziaGarsus()' hides inherited member
'Paveldejimas.NamGyvunas.SkleidziaGarsus()'. Use the new keyword if
hiding was intended.

Programos su paveldėjimu pavyzdys

...

```
NamGyvunas pikis = new NamGyvunas("Pikis");  
pikis.SkleidziaGarsus();
```

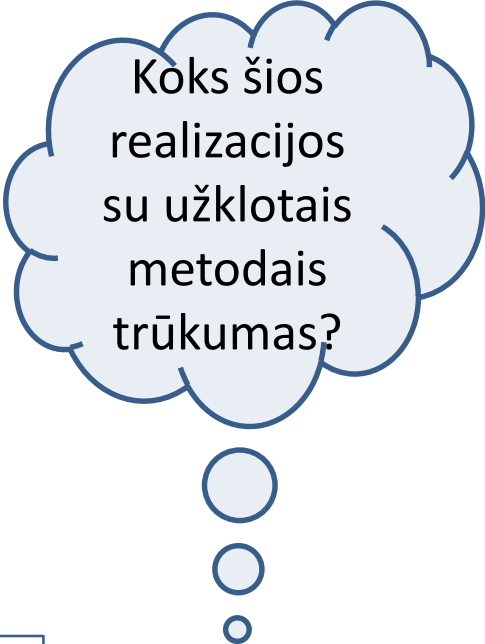
```
Suo dikas = new Suo("Dikas");  
dikas.SkleidziaGarsus();
```

```
Katinas murklys = new Katinas("Murklys");  
murklys.SkleidziaGarsus();
```

...

Ekrane matysite:

```
Pikis skleidžia garsą: ??????  
Dikas skleidžia garsą: Au! Au! Au!  
Murklys skleidžia garsą: Miau! Miau! Miau!
```



Koks šios
realizacijos
su užklotais
metodais
trūkumas?

Geresnis problemos sprendimas

Pasinaudoti **virtualumo** galimybe, kuri leidžia programos vykdymo metu pasirinkti tinkamus metodus (**dinaminis polimorfizmas**).

Virtualaus metodo paskelbimas

Bazinės klasės metodas yra virtualus, jei klasės apraše metodo antraštėje yra žodis **virtual** (netikras, menamas).

Išvestinės klasės metodas, pilnai sutampantis pagal metodo antraštę su bazinės klasės virtualiu metodu, taip pat yra virtualus (tik čia vietoje **virtual** reikia rašyti **override**).

Prisiminkite dažnai savo sukurtose klasėse naudojamą užklotą virtualų klasės **Object** metodą **ToString()**.

Virtualiais **negali būti** klasių konstruktoriai.

Virtualių metodų poreikis

Bazinė klasė – **Figūra**.

Išvestinės klasės: **Apskritimas**, **Stačiakampis**:

Visoms klasėms reikia metodo **Pieštifigūrą()**.

Vadinasi, jis turėtų būti klasėje **Figūra**, tačiau kiekvienai figūrai jis turi būti skirtingas.

Sakykim, dar vėliau sukuriama klasė **Trikampis**.

Metodas **Pieštifigūrą()** turi būti virtualus.

Jei metodas *virtualus*, jis sako kompiliatoriui "Nežinau, kaip reikia įgyvendinti. Palauk, kol bus naudojamas programoje ir paimk įgyvendinimą iš objekto."

Tai *užvėlinto* ar *dinaminio susietumo* technika.

Virtualaus metodo panaudojimo paveldėjime pavyzdys (1/5)

```
class NamGyvunas : Object // bazinė klasė
{
    public string vardas { get; private set; }
    public NamGyvunas(string vrd) // Konstruktorius
    {
        this.vardas = vrd;
    }
    public virtual string Balsas()
    {
        return "?????";
    }
    public void skleidziaGarsus()
    {
        Console.WriteLine("{0} skleidžia garsą: {1}\n",
                           vardas, Balsas());
    }
    public override string ToString()
    {
        return string.Format("{0}", vardas);
    }
}
```

Virtualus metodas

Užklotas klasės
Object
virtualus metodas

Virtualaus metodo panaudojimo paveldėjime pavyzdys (2/5)

```
class Suo : NamGyvunas // išvestinė klasė Suo
{
    public Suo(string vardas) : base(vardas)
    {
    }
    public override string Balsas()
    {
        return "Au! Au! Au!";
    }
    public override string ToString()
    {
        return string.Format("{0}", base.ToString());
    }
}
```

Užklotas klasės
NamGyvunas
virtualus metodus

Užklotas klasės
Object
virtualus metodus

Virtualaus metodo panaudojimo paveldėjime pavyzdys (3/5)

```
class Katinas : NamGyvunas    // išvestinė klasė Katinas
{
    public Katinas(string vardas) : base(vardas)
    {
    }
    public override string Balsas()
    {
        return "Miau! Miau! Miau!";
    }
    public override string ToString()
    {
        return string.Format("{0}", base.ToString());
    }
}
```

Užklotas klasės
NamGyvunas
virtualus metodas

Užklotas klasės
Object
virtualus metodas

Virtualaus metodo panaudojimo paveldėjime pavyzdys (4/5)

```
...  
NamGyvunas pikis = new NamGyvunas("Pikis");  
Console.WriteLine("Naminio gyvūno vardas: {0}", pikis.ToString());  
pikis.SkleidziaGarsus();
```

```
Suo dikas = new Suo("Dikas");  
Console.WriteLine("Šuns vardas: {0}", dikas.ToString());  
dikas.SkleidziaGarsus();
```

```
Katinas murklys = new Katinas("Murklys");  
Console.WriteLine("Katino vardas: {0}", murklys.ToString());  
murklys.SkleidziaGarsus();
```

```
...
```

Virtualaus metodo panaudojimo paveldėjime pavyzdys (5/5)

Ekrane matysite:

Naminio gyvūno vardas: Píkis
Píkis skleidžia garsą: ?????

Šuns vardas: Dikas
Dikas skleidžia garsą: Au! Au! Au!

Katino vardas: Murkllys
Murkllys skleidžia garsą: Miau! Miau! Miau!

Priskyrimo veiksmas paveldėjime

Jeigu klasė **A** yra bazinė (*base*) klasei **B**, arba kitaip tariant, klasė **B** yra išvestinė (*derived*) klasės **A** atžvilgiu, tuomet klasės **B** objektą galima priskirti klasės **A** tipo kintamajam, **bet ne atvirkščiai**. Pvz.:

// **pikis** – bazinės klasės objektas, **dikas** – išvestinės klasės objektas

pikis = **dikas**;

**Console.WriteLine("Naminio gyvūno vardas: {0}",
pikis.ToString());**

pikis.SkleidziaGarsus();

Kas bus
atspausdinta
konsolėje?

// **dikas** = **pikis**; // **Klaida: toks priskyrimas negalimas!**

Iškyla **problema**: ne visų išvestinės klasės kintamųjų reikšmės bus priskirtos bazinės klasės kintamiesiems.

Priskyrimo veiksmas paveldėjime

- Bazinės klasės kintamajam (objektui) galima priskirti anksčiau sukurtus **tos bazinės** klasės objektus.
- Bazinės klasės kintamajam (objektui) galima priskirti anksčiau sukurtus **išvestinių** klasių objektus.

Paaiškinimas:

```
NamGyvunas pikis = new NamGyvunas("Pikis");  
Suo dikas = new Suo("Dikas");
```

```
pikis = dikas;           // kiekvienas šuo yra naminis gyvūnas  
// dikas = pikis;       // Klaida: ne kiekvienas naminis gyvūnas yra šuo
```

Prisiminkime: Objekto kintamasis yra nuorodos tipo (**reference type**), t.y. objekto kintamojo reikšmė yra adresas (šešioliktainis skaičius), kuris rodo, kur atmintyje yra objekto duomenys.

Virtualaus metodo kvietimas

Vietoje bazinės klasės virtualaus (**virtual**) metodo bus kviečiamas atitinkamas (**override**) išvestinės klasės metodas, jei bazinės klasės objektui yra priskirtas išvestinės klasės objektas.

Naminio gyvūno vardas: Dikas
Dikas skleidžia garsą: Au! Au! Au!



Abstrakti klasė

Abstrakčios klasės apibrėžimas

- Klasė yra abstrakti, jei ji turi bent vieną abstraktų metodą. Naudojamas raktažodis **abstract** tiek klasei, tiek ir metodui.
- Abstraktūs metodai neturi įgyvendinimo.
- Abstraktūs metodai yra numanomai virtualūs.
- Abstrakčioje klasėje gali būti ir konkretūs metodai.
- Jei bent vienas metodas abstraktus, tai klasė abstrakti.
- Abstrakčios klasės naudojamos tik **kaip bazinės**.

Abstrakčios klasės objektai

- Programoje **negalima** kurti abstrakčiosios klasės objektų.
- Programoje galima aprašyti abstrakčiosios klasės kintamuosius (objektus). Jiems galima suteikti (priskirti) išvestinių klasių objektų duomenis.

Abstrakčių klasių paskirtis

- Abstrakčiosios klasės naudojamos tam tikros rūšies objektų pačioms bendriausioms savybėms aprašyti.
- Abstrakčiojoje klasėje galima nusakyti, kokias funkcijas gali atlikti objektas, tačiau negalima pilnai apibrėžti, kaip tos funkcijos bus atliekamos, kadangi kiekvienam porūšiui tie veiksmai gali būti skirtingi.

Abstrakčių klasių paskirtis

- Abstraktieji metodai parodo tik funkcijas, kurias galima taikyti tos rūšies objektams, tačiau funkcijos veikimas kiekvienai išvestinei klasei skiriasi.
- Tokie metodai gali būti konkretizuojami, t. y. realizuojami tik išvestinėse klasėse.
- Jei išvestinė klasė nerealizuoja abstraktaus metodo, ji tampa abstrakčiaja.

Abstrakčios savybės

Abstrakčios savybės aprašymas:

```
public abstract SavybėsTipas SavybėPavadinimas  
{  
    get; set;  
}
```

- Išvestinėse klasėse abstrakčios savybės turi būti skelbiamos **override** (kaip ir metodai) .
- Išvestinėse klasėse turi būti realizuoti abu prieigos metodai (**set** ir **get**).

Abstrakčios savybės pavyzdys

```
public abstract class BazinėKlasė  
{  
    protected int skaicius;  
    public abstract int skaičius { get; set; }  
}
```

Abstrakti savybė

```
class IšvestinėKlasė : BazinėKlasė  
{  
    public override int skaičius  
    {  
        get  
        {  
            return skaicius;  
        }  
        set  
        {  
            skaicius = value;  
        }  
    }  
}
```

Abstrakti savybė realizuota
išvestinėje klasėje

Abstrakčios savybės pavyzdys

```
...  
IšvestinėKlasė A = new IšvestinėKlasė();  
A.Skaičius = 111;  
Console.WriteLine("Skaičius: {0}", A.Skaičius.ToString());  
...
```

Ekrane matysite:

Skaičius: 111

Konstruktoriai

- Konstruktoriai negali būti abstraktūs – jie nėra paveldimi, būtų niekados neįgyvendinti.
- Statiniai (**static**) metodai negali būti abstraktūs – išvestinė klasė negali užkloti statinių metodų, būtų niekados neįgyvendinti.

Abstrakčios klasės pavyzdys 1 (1/6)

Sukurkite abstrakčią bazinę klasę **Figura**.

Sukurkite išvestines klases:

Apskritimas


Rutulys

Cilindras (ritinys)

Kūgis. Realizuokite savarankiškai.

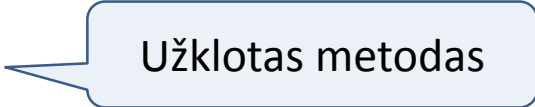
Abstrakčios klasės pavyzdys 1 (2/6)

```
public abstract class Figura : Object // abstrakti klasė
{
    public const double PI = Math.PI;
    protected double x { get; private set; }
    protected double y { get; private set; }
    public Figura()
    {
    }
    public Figura(double x, double y)
    {
        this.x = x;
        this.y = y;
    }
    public abstract double Plotas();
}
```

 Abstraktus metodas

Abstrakčios klasės pavyzdys 1 (3/6)

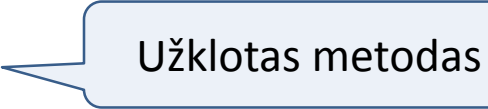
```
public class Apskritimas : Figura // išvestinė klasė
{
    public Apskritimas(double r) : base(r, 0)
    {
    }
    public override double Plotas()
    {
        return PI * x * x; // Pi * r * r
    }
}
```

A light blue rounded rectangular callout box with a pointer directed at the 'Plotas()' method in the code. It contains the text 'Užklotas metodas'.

Užklotas metodas

Abstrakčios klasės pavyzdys 1 (4/6)

```
class Rutulys : Figura      // išvestinė klasė
{
    public Rutulys(double r) : base(r, 0)
    {
    }
    public override double Plotas()
    {
        return 4 * PI * x * x;      // 4 * pi * r * r
    }
}
```

A light blue rounded rectangular callout box with a pointer directed at the 'Plotas()' method signature in the code. It contains the text 'Užklotas metodas'.

Užklotas metodas

Abstrakčios klasės pavyzdys 1 (5/6)

```
class Cilindras : Figura    // išvestinė klasė
{
    public Cilindras(double r, double h) : base(r, h)
    {
    }

    public override double Plotas()
    {
        // 2 * pi * r * r + 2 * pi * r * h
        return 2 * PI * x * x + 2 * PI * x * y;
    }
}
```

Užklotas metodas

Abstrakčios klasės pavyzdys 1 (6/6)

```
...  
double r = 1.0;    // spindulys  
double h = 10.0;   // aukštinė  
Figura apskritimas = new Apskritimas(r);  
Figura rutulys = new Rutulys(r);  
Figura cilindras = new Cilindras(r, h);  
Console.WriteLine("Apskritimo plotas = {0, 8:f3}", apskritimas.Plotas());  
Console.WriteLine("Rutulio plotas    = {0, 8:f3}", rutulys.Plotas());  
Console.WriteLine("Cilindro plotas    = {0, 8:f3}", cilindras.Plotas());  
//Figura geomFigura = new Figura(r, h); // klaida!  
...
```

Ekrane matysite:

Apskritimo plotas	=	3,142
Rutulio plotas	=	12,566
Cilindro plotas	=	69,115

Sukurkite abstrakčią bazinę klasę **Auto**.

Sukurkite dvi išvestines klases:

Keleivinis

Krovininis

Sukurkite abstrakčios klasės objektų masyvą.

```
public abstract class Auto : Object    // automobilio klasė
{
    public int Metai { get; private set; } // pagaminimo metai
    public double Rida { get; private set; } // tūkst. km
    public Auto(int metai = 0, double rida = 0.0)
    {
        this.Metai = metai;
        this.Rida = rida;
    }

    public abstract double NusidevejimoProc();

    public override string ToString()
    {
        return string.Format("Metai = {0, 4:d}; Rida = {1, 7:f3}",
                               Metai, Rida);
    }
}
```

Abstraktus metodas

```
class Keleivinis : Auto // keleivinio a. klasė
{
    public int vietuSk { get; private set; } // vietų skaičius
    public Keleivinis(int metai = 0, double rida = 0.0,
                     int vietuSk = 0) : base(metai, rida)
    {
        this.vietuSk = vietuSk;
    }
    public override double NusidevejimoProc()
    {
        return 0.1 * Rida + (2017 - Metai) + 0.2 * vietuSk;
    }
    public override string ToString()
    {
        return string.Format("{0} Vietų skaičius = {1, 2:d}.",
                              base.ToString(), vietuSk);
    }
}
```

Užklotas metodas

```
class Krovininis : Auto // Krovininio a. klasė
{
    public double kelGalia { get; private set; } // tonomis
    public Krovininis(int metai = 0, double rida = 0.0,
                      double kelGalia = 0.0) : base(metai, rida)
    {
        this.kelGalia = kelGalia;
    }
    public override double NusidevejimoProc()
    {
        return 0.1 * Rida + (2017 - Metai) + 0.1 * kelGalia;
    }
    public override string ToString()
    {
        return string.Format("{0} Keliamoji galia = {1, 6:f3}.",
                              base.ToString(), kelGalia);
    }
}
```

Užklotas metodas

...

```
Auto[] FirmosAuto = new Auto[2];  
FirmosAuto[0] = new Keleivinis(2010, 15.5, 5);  
FirmosAuto[1] = new Krovininis(2007, 25.5, 5.5);  
foreach (Auto auto in FirmosAuto)  
{  
    Console.WriteLine("{0}", auto);  
    Console.WriteLine("Nusidėvėjimas = {0, 5:f2} %.",  
        auto.NusidevejimoProc());  
}
```

...

Ekrane matysite:

```
Metai = 2010; Rida = 15,500 Vietų skaičius = 5.  
Nusidėvėjimas = 8,55 %.  
Metai = 2007; Rida = 25,500 Keliamoji galia = 5,500.  
Nusidėvėjimas = 12,10 %.
```

Polimorfiniai kintamieji(1/9)

- Kintamasis, sukurtas pagal bazinę klasę ir naudojamas išvestinių klasių metodams iškviesti, vadinamas polimorfiniu.
- Polimorfinis kintamasis gali iškviesti tik tuos metodus, kurie yra bazinėje klasėje.
- Norint priskirti bazinės klasės kintamąjį išvestinės klasės kintamajam, reikia atlikti išreikštą konversiją. Konvertuotam kintamajam galima kviesti išvestinės klasės unikalius metodus, kurių nėra bazinėje klasėje.
- Konvertavimui galima naudoti operatorių **as**.
- Tipo tikrinimui naudojamas bazinis žodis **is**.
- Objektas žino savo tipą. Jį galima sužinoti naudojant metodą **GetType()**.

Polimorfiniai kintamieji(2/9)

Tekstiniam faile duota informacija apie žaidėjus, futbolistus ir krepšininkus: pavardė, vardas, pelnyti įvarčiai arba įmestų taškų vidurkis. Požymis f – futbolistas, k – krepšininkas.

Pažymėkite *rezultatyvus* futbolistus, kurie pelnė daugiau nei 7 įvarčius ir krepšininkus, kurių taškų vidurkis per rungtynes daugiau nei 15 taškų. Prieš skaičiavimus visų krepšininkų vidurkius sumažinkite 1.

```
f Jonaitis Jonas 8  
f Petraitis Petras 2  
f Juozaitis Gintaras 7  
k Petraitis Pranas 17  
k Petrutis Jonas 14
```

Polimorfiniai kintamieji(3/9)

```
abstract class Zmogus
{
    protected const int Civarciai = 7;
    protected const double Ctaskai = 15;
    public string pavarde { get; private set; }
    public string vardas { get; private set; }
    public Zmogus()
    { }
    public void DetiVarda(string pavarde, string vardas)
    {
        this.pavarde = pavarde;
        this.vardas = vardas;
    }
    public override string ToString()
    {
        return string.Format(" {0,-12} {1,-12} ", vardas, pavarde);
    }
    public abstract void DetiDuomenis(string eil); // abstraktus metodas
    public abstract bool ArRezultatyvus(); // abstraktus metodas
}
```

Polimorfiniai kintamieji(4/9)

```
class Futbolistas : Zmogus
{
    public int ivarciai { get; private set; }
    public override void DetiDuomenis(string eil)
    {
        string[] parts = eil.Split(' ');
        DetiVarda(parts[1], parts[2]);
        ivarciai = int.Parse(parts[3]);
    }
    public override bool ArRezultatyvus()
    {
        if (ivarciai > Civarčiai) return true;
        else return false;
    }
    public override string ToString()
    {
        return string.Format("Futbolistas {0} {1} ",
                               base.ToString(), ivarciai);
    }
}
```

Polimorfiniai kintamieji(5/9)

```
class Krepsininkas : Zmogus
{
    public double taskai { get; private set; }
    public override void DetiDuomenis(string eil)
    {
        string[] parts = eil.Split(' ');
        DetiVarda(parts[1], parts[2]);
        taskai = double.Parse(parts[3]);
    }
    public void Sumazinti() {
        taskai--; }
    public override bool ArRezultatyvus()
    {
        if (taskai > Ctaskai) return true;
        else return false;
    }
    public override string ToString()
    {
        return string.Format("Krepšininkas {0} {1} ",
            base.ToString(), taskai);
    }
}
```

Polimorfiniai kintamieji(6/9)

```
class Program
```

```
{
```

```
    const int Cn = 20;
```

```
    const string CFd1 = "..\\..\\Duom.txt";
```

```
    const string CFr = "..\\..\\Rezultatai.txt";
```

```
    static void Main(string[] args)
```

```
    {
```

```
        Zmogus[] Azm = new Zmogus[Cn]; // saugomi visų žaidėjų duomenys
```

```
        if (File.Exists(CFr))
```

```
            File.Delete(CFr);
```

```
        int kiek;
```

```
        Skaityti(CFd1, Azm, out kiek);
```

```
        Spausdinti(CFr, Azm, kiek, "Duomenys", 50, "Pradinė");
```

```
        Perskaiciuoti(Azm, kiek);
```

```
        Spausdinti(CFr, Azm, kiek, "Rezultatai", 60, "Perskaičiuota");
```

```
    }
```

```
    // metodai
```

```
    // ...
```

```
}
```

Polimorfiniai kintamieji(7/9)

```
static void Skaityti(string fv, Zmogus[] An, out int kiek)
{
    string[] lines = File.ReadAllLines(fv,
                                     Encoding.GetEncoding(1257));
    kiek = 0;
    foreach (string line in lines)
    {
        if (line[0] == 'f')
            An[kiek] = new Futbolistas();
        else An[kiek] = new Krepsininkas();
        An[kiek++].DetiDuomenis(line);
    }
}
```

Polimorfiniai kintamieji(8/9)

```
static void Spausdinti(string fv, Zmogus[] Skl, int nn,
                      string antraštė, int ilgis, string tipas)
{
    using (var fr = File.AppendText(fv))
    {
        fr.WriteLine(antraštė);
        fr.WriteLine(new string('-', ilgis));
        for (int i = 0; i < nn; i++)
        {
            fr.Write(Skl[i]);
            if (tipas != "Pradinė" && Skl[i].ArRezultatyvus())
                fr.WriteLine(" rezultatyvus");
            else fr.WriteLine();
        }
        fr.WriteLine(new string('-', ilgis));
    }
}
```

Polimorfiniai kintamieji(9/9)

```
static void Perskaiciuoti(Zmogus[] Skl, int nn)
{
    for (int i = 0; i < nn; i++)
    {
        if (Skl[i] is Krepsininkas) // Tikrinamas tipas
        {
            Krepsininkas naujas = Skl[i] as Krepsininkas; //Konversija
            naujas.Sumazinti();
        }
    }
}
```

Duomenys

Futbolistas	Jonas	Jonaitis	8
Futbolistas	Petras	Petraitis	2
Futbolistas	Gintaras	Juozaitis	7
Krepšininkas	Pranas	Petraitis	17
Krepšininkas	Jonas	Petrutis	14

Rezultatai

Futbolistas	Jonas	Jonaitis	8	rezultatyvus
Futbolistas	Petras	Petraitis	2	
Futbolistas	Gintaras	Juozaitis	7	
Krepšininkas	Pranas	Petraitis	16	rezultatyvus
Krepšininkas	Jonas	Petrutis	13	

Klasių kintamieji ir nuorodos

4 priskyrimo variantai tarp bazinių, išvestinių klasių kintamųjų ir bazinių, išvestinių klasių nuorodų:

1. Bazinės klasės nuorodą visuomet priskiriame bazinės klasės kintamajam.
2. Išvestinės klasės nuorodą visuomet priskiriame išvestinės klasės kintamajam.
3. Išvestinės klasės nuorodos priskyrimas bazinės klasės kintamajam yra saugus. Tačiau ši nuoroda gali kviesti tik bazinės klasės narius
4. Tiesioginis bazinės klasės nuorodos priskyrimas išvestinė klasės kintamajam neįmanomas (kompiliavimo klaida). Reikalinga išreikšta konversija. Reikia patikrinti tipą (**is** operatorius) ir atlikti išreikštą konversiją, priešingu atveju – gausime vykdymo klaidą.

sealed klasė ir jos metodai

sealed klasė negali būti bazinė klasė.

sealed klasė negali būti abstrakti klasė.

sealed klasė negali dalyvauti paveldėjime.

Metodas gali būti paskelbtas sealed. (Išbandykite)

Metodai static taip pat yra ir sealed.

Klasė string yra sealed klasė.

sealed klasės pavyzdys

```
public sealed class KlaseA
{
    public KlaseA()        // konstruktorius be parametru
    {
        Console.WriteLine("Dirba klasės konstruktorius.");
    }
    public void Spausdinti()
    {
        Console.WriteLine("Dirba klasės metodas Spausdinti().");
    }
}

...
KlaseA obj = new KlaseA();
obj.Spausdinti();
...
```

Dirba klasės konstruktorius.
Dirba klasės metodas Spausdinti().

sealed klasė ir paveldėjimas

Pabandykime sukurti išvestinę klasę:

```
public class IšvestinėKlasė : KlaseA
{
    // ...
}
```

Kompiliavimo klaida:

(Error 1 'Paveldėjimas.IšvestinėKlasė': cannot derive from sealed type 'Paveldėjimas.KlaseA')

Object klasė

```
public class Object  
{
```

```
    // virtualūs nariai
```

```
    public virtual bool Equals(object obj);
```

```
    protected virtual void Finalize();
```

```
    public virtual int GetHashCode();
```

```
    public virtual string ToString();
```

```
    // Instance level
```

```
    public Type GetType();
```

```
    protected object MemberwiseClone();
```

```
    // Statiniai nariai
```

```
    public static bool Equals(object objA, object objB);
```

```
    public static bool ReferenceEquals(object objA,  
                                       object objB);
```

```
}
```

Object klasės metodai

Metodas	Aprašymas
Equals()	Standartiškai metodas grąžina tiesą, jei abi nuorodos rodo tą pačią vietą. Dėl to jis užklojamas. Reikia kartu užkloti ir GetHashCode().
Finalize()	Skirtas resursų grąžinimui, geriau neužkloti.
GetHashCode()	Unikalčiai apibrėžia specifinį objektą, naudojamas kaip raktas.
GetType()	Grąžina Type objektą, kuris aprašo jūsų objektą.
Memberwise Clone()	Sekli kopija – reikšmių ir nuorodų kopija.
ToString()	Grąžina kaip eilutę <namespace>.<typename>
Reference Equals()	Gauna dvi nuorodos ir grąžina tiesą, jei abi nuorodos rodo tą patį objektą arba abi nuorodos yra null.

Object klasės metodai

```
public class Asmuo
{
    // ...
}
```

```
...
Asmuo asmuoA = new Asmuo();
Console.WriteLine("ToString: {0}", asmuoA.ToString());
Console.WriteLine("Hash code: {0}", asmuoA.GetHashCode());
Console.WriteLine("Type: {0}", asmuoA.GetType());
Asmuo asmuoB = asmuoA;
if (Object.ReferenceEquals(asmuoA, asmuoB))
    Console.WriteLine("Objektų asmuoA ir asmuoB nuorodos sutampa.");
object obj = asmuoB;
if (obj.Equals(asmuoA) && asmuoB.Equals(obj))
    Console.WriteLine("Tas pats objektas.");
...
```

Ekrane matysite:

```
ToString: Paveldejimas.Asmuo
Hash code: 21083178
Type: Paveldejimas.Asmuo
Objektų asmuoA ir asmuoB nuorodos sutampa.
Tas pats objektas.
```

Šioje temoje susipažinsite:

1. Statiniu ir dinaminio polimorfizmu
2. Virtualiais metodais
3. Dinaminio susietumu
4. Abstrakčia (**abstract**) klase
5. **sealed** klase
6. **Object** klase



Klausimai?