

T10. Išimtys (Exceptions)

1 ak. val.

Temos klausimai

1. Sisteminės išimtys
2. Išimčių gaudymo blokai
3. Vartotojo kuriamos išimčių klasės
4. Reikšmių kontrolė jų suteikimo metu



Bazinė išimčių klasė (Exception class)

Išimčių valdytojas

- Išimtys yra objektai, todėl joms apdoroti taikomi visi objektinio programavimo mechanizmai.
- 4 baziniai žodžiai: **try**, **catch**, **throw**, **finally**. Išimtis programa aptinka vykdydama bloką **try**. Betarpiškai už šio bloko rašomas **catch** blokas, kuriame esantys sakiniai nusako, kas turi būti atliekama esant išimčiai. Vienam **try** blokui gali būti užrašyta vienas arba keli **catch** blokai.
- Pati išimtis generuojama (ir pereinama į išimties apdorojimo bloką) vykdant **throw** sakinį. Šio tipo sakiniai rašomi **try** bloke. **Catch** blokų turi būti tiek, kiek skirtingų tipų išimčių apibrėžta **throw** sakiniuose.

Sintaksė

```
try
{
    // Kodas, kuriame gali būti išimtis
    throw[reiškinys]
}
[ catch(išimties aprašas) {
    // Kodas, vykdomas catch eilutėje atsiradus nurodyto tipo
    // išimtimis
}
[catch(išimties aprašas) {
    // Kodas, apdorojantis kito tipo išimtis
} ]
. . .
]
```

Išimčių **Exception class**

Išimčių klasė **Exception class** turi:

- 4 užklotus konstruktorius.
- 8 savybes (properties).
- 8 metodus.
- 1 įvykį.

Išimčių **Exception class** aprašas

Išimčių klasė **Exception class** aprašai:

1. Exception Pavadinimas = new Exception ();

Sukuriamas naujas išimčių objektas.

**2. Exception Pavadinimas1 = new Exception
(eilutė);**

Sukuriamas naujas išimčių objektas su klaidos pranešimu **eilutė**.

**3. Exception Pavadinimas3 = new Exception
(eilutė, Pavadinimas2);**

Sukuriamas naujas išimčių objektas su klaidos pranešimu **eilutė** ir kitu išimčių objektu **Pavadinimas2**, kuris ir yra naujojo išimčių objekto priežastis.

Dažniausiai naudojamos išimčių klasės

Exception class savybės

Savybė	Aprašas
Data	Papildoma programuotojo informacija, kuri pateikiama raktas/reikšmė poromis. Numatytoji – tuščia.
HelpLink	Paima arba priskiria puslapio adresą, kur daugiau paaiškinimų.
InnerException	Informacija apie ankstesnę išimtį, kuri sukėlė dabartinę.
Message	Pranešimas apie klaidą – konstruktoriaus parametras.
Source	Paima arba priskiria asamblėjos ar objekto, kuris išmetė išimtį, vardą.
StackTrace	Kvietimų seka, kuri atvedė iki išimties.
TargetSite	Detalės apie metodą, kuris išmetė išimtį.

Pavyzdys_1 1/11

Duota informacija apie įrenginį: *pavadinimas, veikimo charakteristika*. Taip pat žinoma maksimali leistina veikimo charakteristikos reikšmė. Didinant charakteristikos reikšmę pastoviu žingsniu 10 kartų, atspausdinkite galimas charakteristikos reikšmes. Nustatykite, prie kokios charakteristikos reikšmės viršijama maksimali leistina reikšmė.

```
public class Įrenginys
{
    // didžiausia leistina įrenginio charakteristikos reikšmė
    private const int maxReikšmė = 170;
    public string pav { get; set; } // įrenginio pavadinimas
    public int savybė { get; set; } // įrenginio charakteristika
    private bool tinka = true; // ar padidinta charakteristika tinka

    public Įrenginys() { } // Konstruktorius
    // Konstruktorius su parametrais
    // pav - įrenginio pavadinimas
    // pr - pradinė charakteristika
    public Įrenginys(string pav, int pr)
    {
        this.pav = pav;
        this.savybė = pr;
    }

    ...
}
```

```
public class Įrenginys
{
    // tęsinys
    // Duomenų teisingumo tikrinimas
    public void Tikrinimas(int pokytis)
    {
        if (tinka == false)
            Console.WriteLine("Tolimesnis didinimas negalimas!!!");
        else
        {
            savybė = savybė + pokytis;
            if (savybė > maxReikšmė)
            {
                Console.WriteLine("{0} viršijo max reikšmę {1}", pav,
                                   maxReikšmė);
                tinka = false;
            }
            else Console.WriteLine("Galima charakteristika {0}",
                                   savybė);
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Įrenginys naujas = new Įrenginys("Alfa", 20);
        for (int i = 0; i < 10; i++)
            naujas.Tikrinimas(45); // kitimo žingsnis 45
    }
}
```

Pavyzdys_1 5/11

```
Galima charakteristika 65
Galima charakteristika 110
Galima charakteristika 155
Alfa viršijo max reikšmę 170
Tolimesnis didinimas negalimas!!!
Tolimesnis didinimas negalimas!!!
Tolimesnis didinimas negalimas!!!
Tolimesnis didinimas negalimas!!!
Tolimesnis didinimas negalimas!!!
Tolimesnis didinimas negalimas!!!
Press any key to continue . . .
```

Pavyzdys_1 6/11

```
// Pakeičiame klasės Įrenginys skaičiavimo metodą Tikrinimas į
// metodą Tikrinimas1. Panaudojame bedrąją išimtį
public void Tikrinimas1(int pokytis)
{
    if (tinka == false)
        Console.WriteLine("Tolimesnis didinimas negalimas!!!");
    else
    {
        savybė = savybė + pokytis;
        if (savybė > maxReikšmė)
        {
            tinka = false;
            throw new Exception(string.Format
                ("{0} viršijo max reikšmę {1}", pav, maxReikšmė));
        }
        else
            Console.WriteLine("Galima charakteristika {0}", savybė);
    }
}
```

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 20);
    // Išimčių gaudymas
    try
    {
        for (int i = 0; i < 5; i++)
            naujas.Tikrinimas1(45);
    }
    catch (Exception ex) // Gaudo visas išimtis
    {
        Console.WriteLine("Išimčių gaudymas");
        Console.WriteLine(" Metodas {0}", ex.TargetSite);
        Console.WriteLine(" Pranešimas {0}", ex.Message);
        Console.WriteLine(" Šaltinis {0}", ex.Source);
    }
}
```

Pavyzdys_1 8/11

```
Galima charakteristika 65
Galima charakteristika 110
Galima charakteristika 155
Išimčių gaudymas
Metodas Void Tikrinimas1(Int32)
Pranešimas Alfa viršijo max reikšmę 170
Šaltinis Išimtys
Press any key to continue . . .
```


Pavyzdys_1 9/11

```
// Pakeičiame klasės Įrenginys skaičiavimo metodą Tikrinimas į
// metodą Tikrinimas2. Panaudojame bedrąją išimtį
public void Tikrinimas2(int pokytis)
{
    if (tinka == false)
        Console.WriteLine("Tolimesnis didinimas negalimas!!!");
    else
    {
        savybė = savybė + pokytis;
        if (savybė > maxReikšmė)
        {
            tinka = false;
            Exception ex = new Exception(string.Format
                ("{0} viršijo max reikšmę {1}", pav, maxReikšmė));
            ex.HelpLink = "http://proin.ktu.lt";
            throw ex;
        }
        else
            Console.WriteLine("Galima charakteristika {0}", savybė);
    }
}
```

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 20);
    // Išimčių gaudymas
    try
    {
        for (int i = 0; i < 5; i++)
            naujas.Tikrinimas2(45);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Išimčių gaudymas");
        Console.WriteLine(" Metodas {0}", ex.TargetSite);
        Console.WriteLine(" Pranešimas {0}", ex.Message);
        Console.WriteLine(" Šaltinis {0}", ex.Source);
        Console.WriteLine(" Hipernuoroda {0}", ex.HelpLink);
    }
}
```

Pavyzdys_1 11/11

```
Galima charakteristika 65
Galima charakteristika 110
Galima charakteristika 155
Išimčių gaudymas
  Metodus Void Tikrinimas2(Int32)
  Pranešimas Alfa viršijo max reikšmę 170
  Šaltinis Išimtys
  Hipernuoroda http://proin.ktu.lt
Press any key to continue . . .
```

Išimčių grupės

Dvi išimčių grupės:

```
public class SystemException: Exception
{
    // įvairūs konstruktoriai
}
```

```
// Išimtys: ArgumentOutOfRangeException,  
// IndexOutOfRangeException ir daug kitų
```

```
public class ApplicationException: Exception
{
    // įvairūs konstruktoriai
}
```

Vartotojo išimtis 1

```
// Vartotojo sukurta išimčių klasė
public class ĮrenginioCharakterIšimtis : ApplicationException
{
    private string messageDetails = String.Empty;
    public DateTime ErrorTimeStamp { get; set; }
    public string CauseOfError { get; set; }
    // Konstruktorius be parametrų
    public ĮrenginioCharakterIšimtis()
    {
    }
    // Konstruktorius su parametrais
    public ĮrenginioCharakterIšimtis(string message, string cause, DateTime time)
    {
        messageDetails = message;
        CauseOfError = cause;
        ErrorTimeStamp = time;
    }

    // Užklojam Message
    public override string Message
    {
        get { return string.Format("Klaidos pranešimas: {0}", messageDetails); }
    }
}
```

Vartotojo išimtis 2

```
// Pakeičiame klasės Įrenginys skaičiavimo metodą Tikrinimas į  
// metodą Tikrinimas3. Panaudojame sukurta išimčių klasę  
public void Tikrinimas3(int pokytis)  
{  
    if (tinka == false)  
        Console.WriteLine("Tolimesnis didinimas negalimas!!!");  
    else  
    {  
        savybė = savybė + pokytis;  
        if (savybė > maxReikšmė)  
        {  
            tinka = false;  
            ĮrenginioCharakterIšimtis_ex = new  
                ĮrenginioCharakterIšimtis(string.Format  
                    ("{0} įrenginys nedirba. ", pav ),  
                    "Viršijo max reikšmę.", DateTime.Now);  
            ex.HelpLink = "http://proin.ktu.lt";  
            throw ex;  
        }  
    }  
    else  
        Console.WriteLine("Galima charakteristika {0}", savybė);  
}}
```

Išimties gaudymas 1

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 20);
    // Išimčių gaudymas
    try
    {
        for (int i = 0; i < 5; i++)
            naujas.Tikrinimas3(45);
    }
    catch (ĮrenginioCharakterIšimtis ex)
    {
        Console.WriteLine("Išimčių gaudymas");
        Console.WriteLine(" Laikas: {0}", ex.ErrorTimeStamp);
        Console.WriteLine(" Pranešimas: {0}", ex.Message);
        Console.WriteLine(" Priežastis: {0}", ex.CauseOfError);
        Console.WriteLine(" Hipernuoroda: {0}", ex.HelpLink);
    }
}
```

Išimties gaudymas 2

```
Galima charakteristika 65
Galima charakteristika 110
Galima charakteristika 155
Išimčių gaudymas
Laikas: 2016-05-23 19:06:08
Pranešimas: Klaidos pranešimas: Alfa įrenginys nedirba.
Priežastis: Viršijo max reikšmę.
Hipernuoroda: http://proin.ktu.lt
Press any key to continue . . .
```


Kelios išimtis 1

```
// Pakeičiame klasės Įrenginys skaičiavimo metodą Tikrinimas į
// metodą Tikrinimas4. Panaudojame sukurtą išimčių klasę ir kelias išimtis
public void Tikrinimas4(int pokytis)
{
    if (tinka == false)
        Console.WriteLine("Tolimesnis didinimas negalimas!!!");
    else
    {
        if (pokytis < 0)
        {
            throw new ArgumentOutOfRangeException("Pokytis ",
                "Pokytis negali būti neigiamas");
        }
        savybė = savybė + pokytis;
        if (savybė > maxReikšmė)
        {
            tinka = false;
            IrenginioCharakterIšimtis_ex = new
                IrenginioCharakterIšimtis(string.Format
                    ("{0} įrenginys nedarba. ", pav), Viršijo_max_reikšmę, DateTime.Now);
            ex.HelpLink = "http://proin.ktu.lt";
            throw ex;
        }
        else Console.WriteLine("Galima charakteristika {0}", savybė);
    }
}
```

Kelios išimties 2

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 45);
    try
    {
        naujas.Tikrinimas4(-30);
    }
    catch(ĮrenginioCharakterIšimtis ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch(ArgumentOutOfRangeException ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Kelios išimtys 3

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 45);
    try
    {
        naujas.Tikrinimas4(-30);
    }
    catch(ĮrenginioCharakterIšimtis ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch(ArgumentOutOfRangeException ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

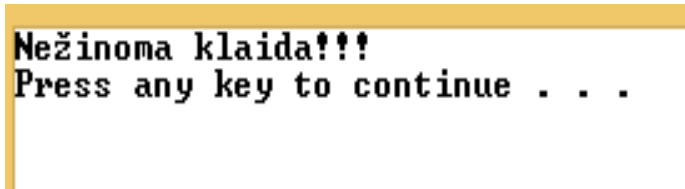
Kelios išimties 4

```
Pokytis negali būti neigiamas  
Parameter name: Pokytis  
Press any key to continue . . .
```

Vienas iš būdų sužinoti išimtis – užvesti žymeklį Visual Studio aplinkoje ant atitinkamo metodo .

Išimtis be pranešimo

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 45);
    try
    {
        naujas.Tikrinimas4(-30);
    }
    catch
    {
        Console.WriteLine("Nežinoma klaida!!!");
    }
}
```

A screenshot of a console window with a yellow header bar. It displays the output of the program: 'Nežinoma klaida!!!' followed by 'Press any key to continue . . .' on the next line.

```
Nežinoma klaida!!!
Press any key to continue . . .
```

Išimties persiuntimas

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 45);
    try
    {
        naujas.Tikrinimas4(-30);
    }
    catch(ĮrenginioCharakterIšimtis ex)
    {
        Console.WriteLine(ex.Message);
        throw;
    }
}
```

Naudojamas klasių metoduose. `Main()` metode naudoti nėra prasmės.

finally blokas 1

```
static void Main(string[] args)
{
    Įrenginys naujas = new Įrenginys("Alfa", 45);
    try
    {
        naujas.Tikrinimas4(-30);
    }
    catch (ĮrenginioCharakterIšimtis ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (ArgumentOutOfRangeException ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        Console.WriteLine("Reikia taisyti!!!");
    }
}
```

finally blokas 2

```
Pokytis negali būti neigiamas  
Parameter name: Pokytis  
Reikia taisyti!!!  
Press any key to continue . . .
```

finally blokas vykdomas visada, nepriklausomai nuo eigos **try** bloke. Vykdymas po bloko **finally** priklauso nuo eigos **try** bloke.

Rekomendacijos išimčių klasėms

1. Turėtų paveldėti klasę ApplicationException.
2. Vardas turėtų baigtis Exception.
3. Turėtų bent 3 konstruktorius:
 1. Be parametų – numatytasis pranešimas perduodamas bazinei klasei.
 2. Su string parametru – perduoti klaidos pranešimui.
 3. Su string parametru – perduoti klaidos pranešimui, ir Exception parametru dėl vidinių išimčių perdavimo.

Išimčių modelis

1. Kai try bloke pasirodo išimtis, vykdymas nutraukiamas ir einama į catch bloką, jei toks yra. Gali likti neįvykdytų sakinių.
 2. Įvykdžius catch bloką, valdymas perduodamas žemiau paskutinio catch bloko.
 3. Toks išimčių valdymas vadinamas užbaigimo modeliu.
- Kitose kalbose gali būti pratęsimo modelis, kai grąžinamas valdymas komandai, esančiai iš karto žemiau išimties iššaukimo komandos.
4. Jei išimtis neiššaukiama, catch blokai nevykdomi.
- Atitikimas catch blokui – identiška klasė arba išvestinė klasė.



Klausimai?