Хватање вишеструких изузетака

Окружење .NET дефинише више типова изузетака, а програми могу ухватити већину њих.
У пракси се не пише код за хватање свих изузетака помоћу блока catch.
Треба имати на уму да апликација треба да обради и изузетке на које програмер није ни мислио приликом писања кода, и зато програмер треба да буде сигуран да ће програм ухватити све реално могуће изузетке.
Пример: фамилија изузетака Exception обухвата широк опсег изузетака који се користе у разним деловима .NET окружења.

```
catch (Exception izuzetak)
{
     //...
}
```

У примеру се хватају сви могући изузеци.
Могуће је написати и следеће:

```
catch
{
     //...
}
```

Ако је потребно ухватити изузетак Exception, могуће је изоставити његов назив из процедуре catch, пошто је то подразумевани изузетак.
Али се то не препоручује; објекат изузетка који се прослеђује до процедуре за обраду изузетка може да садржи корисне информације које се тичу изузетка и до којих се не може доћи ако се користи ова верзија хватања изузетка.

Филтрирање изузетака

Могу се филтрирати изузеци који се подударају са процедурама за обраду изузетака како би смо били сигурни да се процедура за обраду одређеног изузетка покреће само када се испуне додатни услови.
Ови услови имају облик логичког израза са префиксом when.
Пример:

```
bool hvatanjeGresaka = ...;
try
{
     //...
}
catch (Exception izuzetak) when (hvatanjeGresaka == true)
{
  //izuzeci se obradjuju samo ako je promenjiva hvatanjeGresaka = true
}
```

У овом примеру се хватају сви изузеци типа Exception у зависности од вредности логичке промењиве hvatanjeGresaka; ако је вредност ове промењиве false, обрада изузетака се не дешава и користи се подразумевани механизам за обраду изузетака.
Ако hvatanjeGresaka има вредност true, покреће се код у блоку catch.

Пример: начин на који апликација пријављује необрађене изузетке

```
using System;
class Program
{
    static void Main()
    {
        try
        {
            int levo = int.Parse(Console.ReadLine());
            int desno = int.Parse(Console.ReadLine());
            int rezultat = 0;
            rezultat = levo + desno;
            Console.WriteLine(rezultat);
            Console.WriteLine($"{levo} + {desno}");
            string krajnji = rezultat.ToString();
        }
        catch (FormatException izuzetak)
        {
            Console.WriteLine(izuzetak.Message);
        }
    }
}
```

Ако би се урадио претходни пример без блокова try и catch, и ако би се као један од два улаза унело слово или текст, прекинуо би се програм и појавила би се порука о необрађеном изузетку: Unhandled Exception: System.FormatException: Input string was not in a correct format.

Циљ примера је да се спречи прекид програма и да се на адекватан начин обради ухваћени изузетак.

У коду се проблем јавља када се унети текст не може парсовати у целобројну вредност.

Процедура catch успешно хвата изузетак FormatException и исписује поруку Input string was not in a correct format

Током писања кода може се десити да су неки изузеци последица других изузетака који су раније подигнути.

Изузетак који VS пријављује је само последњи у ланцу, али је то обично ранији изузетак који истиче стварни узрок проблема.

У претходне изузетке се може ући ширењем својства InnerException, у оквиру дијалога View Detail.

Изворни изузетак је онај који има својство InnerException постављено на null.

Пример:

```
using System;
namespace Proba
{
    class Program
    {
        public static void Main(string[] args)
        {
            try
            {
                int[] a = new int[5];
                a[10] = 12;
            }
            catch (Exception izuzetak) when
(izuzetak.GetType().ToString() == "System.IndexOutOfRangeException")
            {
                MetodaKojaSeStartuje();
            }
        }
        static void MetodaKojaSeStartuje()
        {
            Console.WriteLine("Greska je pokrivena.");
        }
    }
}
```

Задаци за самосталан рад

1. Објаснити зашто се на конзоли приказује порука blok 2.

```
using System;
public class Program
{
    public static void Main()
    {
        int a = 7;
        int b = 0;
        try
        {
            MetodaKojaSeSigurnoIzvrsava();
        }
        catch (Exception izuzetak) when (a / b == 0)
        {
            Console.WriteLine("blok 1");
        }
        catch (Exception izuzetak)
        {
            Console.WriteLine("blok 2");
        }
```

```
    }

    public static void MetodaKojaSeSigurnoIzvrsava()
    {
        Type.GetType(null);
    }
}
```

2. Написати програм који ће ухватити изузетак приликом погрешног уноса типа података који није целобројан. Спречити пад апликације филтрирањем.