

Лекция 18. Обработка ошибок

Курс «Программирование»

КИТ, 2 семестр

Щукин Александр

Валентинович

Типы ошибок

1. Синтаксические
2. Логические
3. Ошибки времени выполнения (run-time error),
исключительные ситуации (exception)
4. Структурированная обработка исключений –
методика, предназначения для обработки
исключений, которые могут возникать в процессе
выполнения программного кода

Подходы к обработке ошибок

1. С использованием операторов условия
2. Не структурированный: использование локальных констант, констант API-интерфейса, COM-интерфейсов...
3. Структурированная обработка исключений – structured exception handling.
 - Унифицированный подход.
 - Объектно-ориентированный подход

Сущности

1. Тип класса
2. Член класса – метод (функция), который способен генерировать (throw) в вызывающем коде экземпляр класса исключения при некоторых обстоятельствах
3. Блок кода на вызывающей стороне, ответственный за обращение к члену (функции), в котором может произойти исключение
4. Блок кода на вызывающей стороне, который будет обрабатывать (перехватывать – catch) исключение в случае его возникновения

Keywords: try, catch, throw, finally

Типы исключений

- Исключения, которые генерируются самой платформой .Net называются исключениями уровня системы. Считаются неустраняемыми ошибками.
- Наследуются от `System.SystemException`
 - `ArgumentOutOfRangeException`,
 - `IndexOutOfRangeException`
 - `StackOverflowException`
- Исключения уровня приложения
 - Пользовательские классы
 - Рекомендуется наследовать от `System.ApplicationException`

Пример использования Try

```
int x, y;  
int res;  
x = 10;  
y = 0;  
res = x / y;  
  
Console.WriteLine(res);  
  
int x, y;  
int res;  
x = 10;  
y = 0;  
try  
{  
    res = x / y;  
}  
catch  
{  
    Console.WriteLine("Error Handler works!");  
    res = 0;  
}  
  
Console.WriteLine(res);
```

Exception ex

```
int x, y;  
int res;  
x = 10;  
y = 0;  
try  
{  
    res = x / y;  
}  
catch (Exception ex)  
{  
    Console.WriteLine("Error Handler works!");  
    Console.WriteLine(ex.Message);  
    res = 0;  
}
```

Используйте GetType()

FormatException ex

```
s = Console.ReadLine();  
try  
{  
    z = ushort.Parse(s);  
}  
catch (System.FormatException ex)  
{  
    Console.WriteLine("Введено не число {0}", ex.Message);  
    Console.WriteLine(ex.GetType());  
}
```


Finally

```
void ReadFile(int index)
{
    // To run this code, substitute a valid path from your local machine
    string path = @"c:\users\public\test.txt";
    System.IO.StreamReader file = new System.IO.StreamReader(path);
    char[] buffer = new char[10];
    try
    {
        file.ReadBlock(buffer, index, buffer.Length);
    }
    catch (System.IO.IOException e)
    {
        Console.WriteLine("Error reading from {0}. Message = {1}", path, e.Message);
    }

    finally
    {
        if (file != null)
        {
            file.Close();
        }
    }
    // Do something with buffer...
```

Распространение (проброс) ошибки

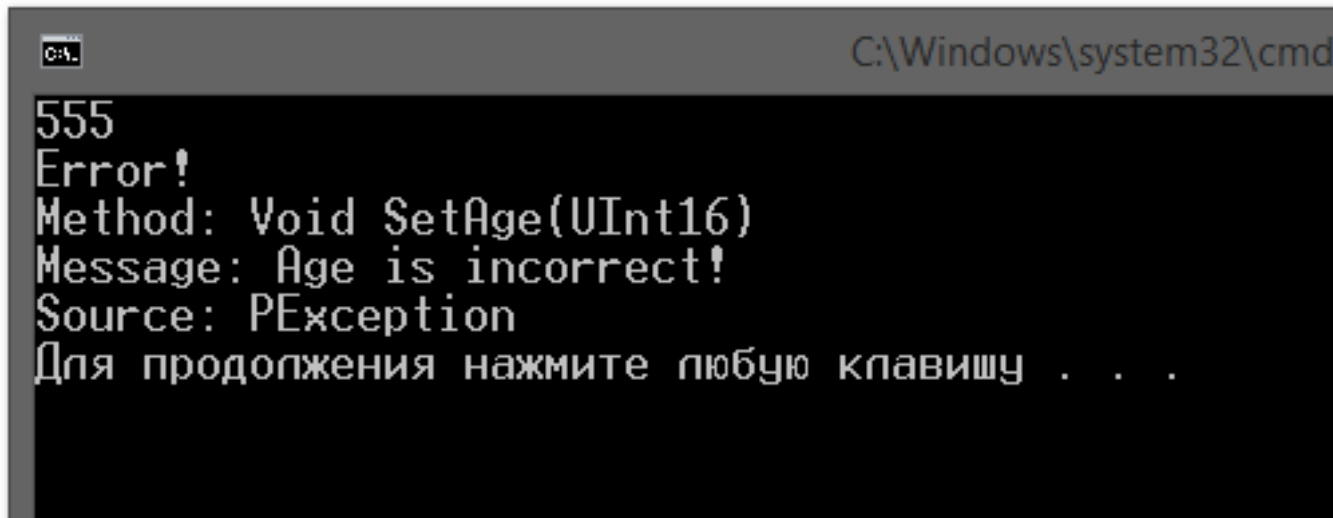
1. Включен обработчик или нет
2. Распространение ошибки вызывающей стороне
3. Возникновение ошибки в обработчике ошибки

Класс System.Exception

- Все исключения наследуются от базового класса System.Exception
 - Конструкторы
 - GetBaseException
 - GetObjectData
 - Data (коллекция пар в IDictionary)
 - HelpLink (URL)
 - InnerException (предыдущее исключение)
 - Message (сообщение)
 - Source (имя сборки или объекта)
 - ...

Пример генерации ошибки

- Структура SHuman1
- Структура SHuman2



```
C:\Windows\system32\cmd
555
Error!
Method: Void SetAge(UInt16)
Message: Age is incorrect!
Source: PException
Для продолжения нажмите любую клавишу . . .
```

- Генерация ошибки – веские основания для этого...