

Хвърляне на изключения



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>



Съдържание

1. Генериране (хвърляне) на изключения
2. Избор на типа на изключението
3. Препоръки при работа с изключения
4. Създаване на потребителски изключения

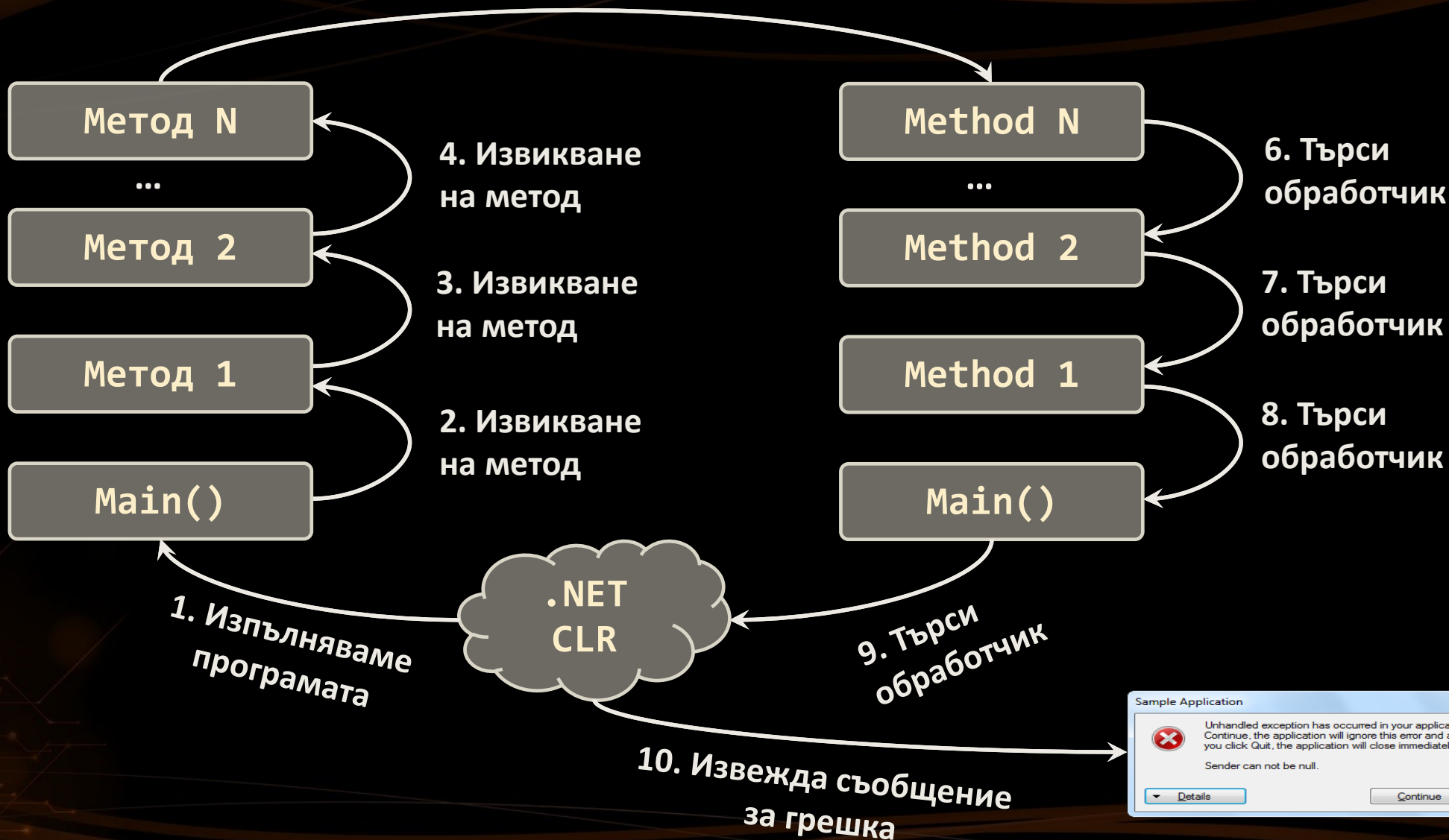


Хвърляне на изключения

- Изключенията се хвърлят (пораждат) чрез командата **throw**
- Целта е уведомяване на кода, извикал текущия програмен блок, за грешка или друга необичайна ситуация
- Когато се хвърля изключение:
 - Изпълнението на програмата спира
 - Изключението пътува през стека
 - Докато не достигне подходящ **catch** блок, който да го прихване
- Неприхванатите изключения извеждат съобщение за грешка

Как работят изключенията?

5. Хвърляне на изключение



Използване на командата Throw

- **Хвърляне** на изключение със съобщение за грешка:

```
throw new ArgumentException("Invalid amount!");
```

- Изключението може да приема съобщение и причина:

```
try
{
    ...
}
catch (SQLException sqlEx)
{
    throw new InvalidOperationException("Cannot save invoice.", sqlEx);
}
```

- Бележка: ако и оригиналното изключение не бъде подадено като параметър, ще загубим първоначалната причина за изключението

Повторно хвърляне на изключение

- Прихванатите изключения може да бъдат хвърлени наново:

```
try
{
    Int32.Parse(str);
}
catch (FormatException fe)
{
    Console.WriteLine("Parse failed!");
    throw fe; // Re-throw the caught exception
}
```

```
catch (FormatException)
{
    throw; // Re-throws the last caught exception
}
```

Хвърляне на изключения – пример

```
public static double Sqrt(double value)
{
    if (value < 0)
        throw new System.ArgumentOutOfRangeException("value",
            "Sqrt for negative numbers is undefined!");
    return Math.Sqrt(value);
}

static void Main()
{
    try
    {
        Sqrt(-1);
    }
    catch (ArgumentOutOfRangeException ex)
    {
        Console.Error.WriteLine("Error: " + ex.Message);
        throw;
    }
}
```



Избиране на типа на изключението

- Когато се подаде невалидна стойност в параметър на метод:
 - **ArgumentException, ArgumentNullException, ArgumentOutOfRangeException**
- Когато заявената операция не се поддържа
 - **NotSupportedException**
- Когато методът все още не е реализиран
 - **NotImplementedException**
- Когато няма друг подходящ стандартен клас изключения
 - Създайте ваш собствен клас (наследяващ **Exception**)

Препоръки при работа с изключения

- **catch** блоковете трябва да започват с изключенията, които са най-ниско в йерархията (т.е. с най-специфичните)
 - И да продължават с по-общите изключения
 - В противен случай ще има грешка при компилация
- Всеки **catch** трябва да обработва само тези изключения, които очаква
 - Ако метод не е компетентен да обработи дадено изключение, той би трябвало да го остави неприхванато
 - Прихващането на всички изключения, независимо от какъв тип са, е популярна лоша практика (анти-шаблон)!

Препоръки при работа с изключения(2)

- Когато генерирате изключение, винаги подавайте на конструктора достатъчно говорящо **пояснително съобщение**
 - Когато хвърляте изключение, винаги подавайте добро описание на проблема, който го е предизвикал
 - Съобщението на изключението трябва да обяснява какво е породило проблема и как той може да бъде решен
 - Добро: „Размерът трябва да е число в диапазона [1...15]" 
 - Добро: „Невалидно състояние. Извикайте първо Initialize()"
 - Лошо: „Неочакван проблем"
 - Лошо: „Невалиден аргумент"
- 

Препоръки при работа с изключения(3)

- Изключенията може да намалят производителността на приложението
 - Затова ги хвърляйте само в ситуации, които са наистина необичайни и трябва да бъдат обработени
 - Не хвърляйте изключения при нормалната работа на програма
 - CLR може да хвърли изключения по всяко време, няма как да бъде предвидено това
 - Например **System.OutOfMemoryException**

Създаване на потребителски изключения

- Потребителските изключения наследяват някой от класовете изключения (най-често **System.Exception**)

```
public class TankException : Exception
{
    public TankException(string msg)
        : base(msg)
    {
    }
}
```

- Те се хвърлят като всяко друго изключение

```
throw new TankException("Not enough fuel to travel");
```


Обобщение

- Изключенията се хвърлят (пораждат) чрез командата **throw**
- Когато се хвърля изключение:
 - Изпълнението на програмата спира
 - Изключението пътува през стека, докато не бъде прихванато от **catch** блок
 - Всеки **catch** трябва да обработва само тези изключения, които очаква
- Прихванато изключение може да бъде хвърлено наново
- Неприхванатите изключения извеждат съобщение за грешка



Хвърляне на изключения



Въпроси?



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni
Foundation

