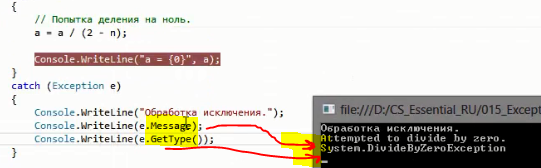
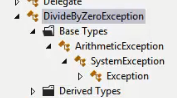
try – в переводе как попытка, пытаться что либо сделать

cath – поймать, ловить

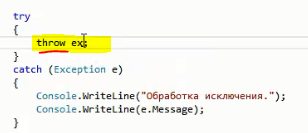


Наследование DivideByZeroException

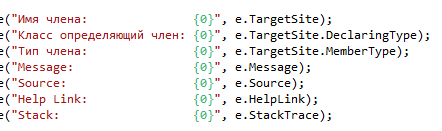


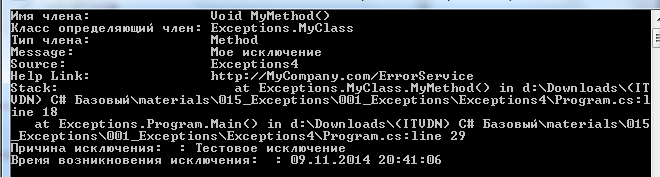
throw – бросить в переводе

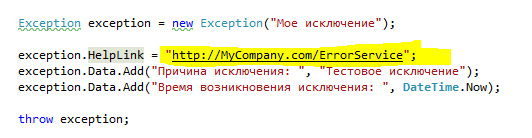




Создаем класс в котором принудительно вызываем исключение





e.TargetSite – метод который вызвал исключение

e.TargetSite.DeclaringType – класс в котором находится метод с исключением

e.TargetSite.MemberType – что вызвало исключение, какой член (в данном случае член класса - метод)

e.Message – то что мы запишем в сообщении об исключении

e.Source – номер исключения

e.HelpLink – то что пользователь укажет (ссылку по исключению)

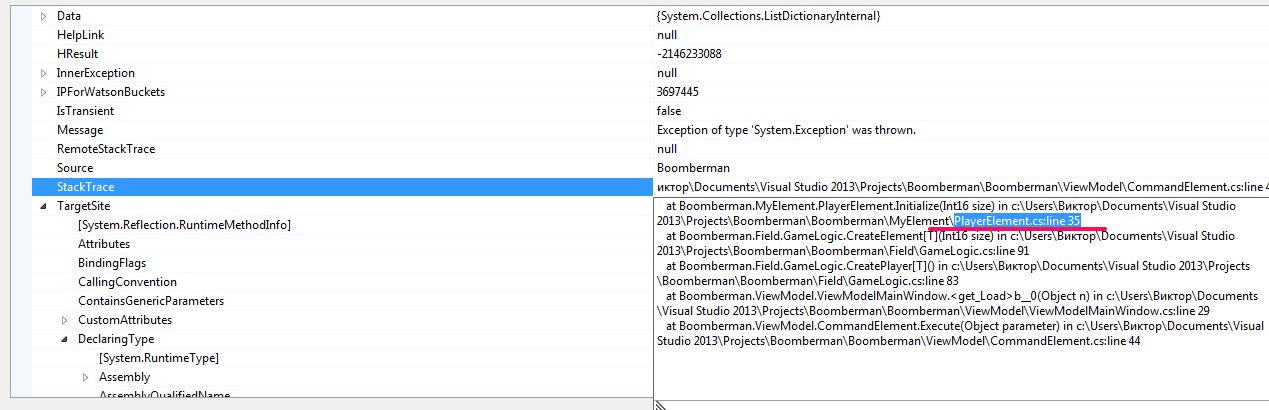
e.StackTrace – информация об исключении (на какой строке объявлен экземпляр исключения и на какой строке вызвано исключение)

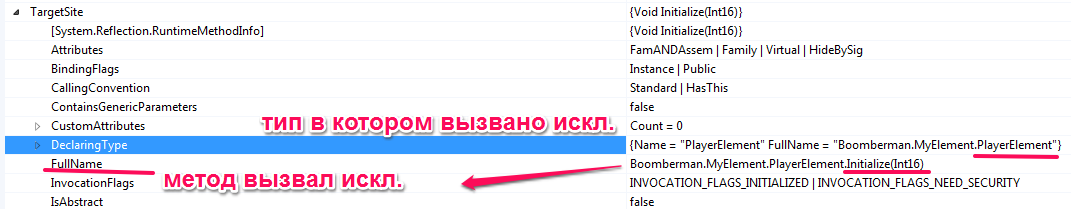
foreach (**DictionaryEntry** de in e.Data)

Console.WriteLine("{0} : {1}", de.Key, de.Value); - дополнительная информация от пользователя

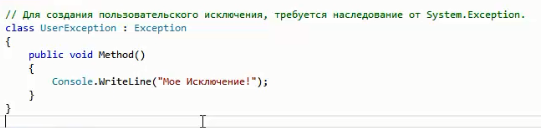
Словарь 

Номер строки и тип (класс) вызванного исключения





Пользовательские исключения

Должны наследоваться System.Exception. 

Внутреннее и внешнее исключение

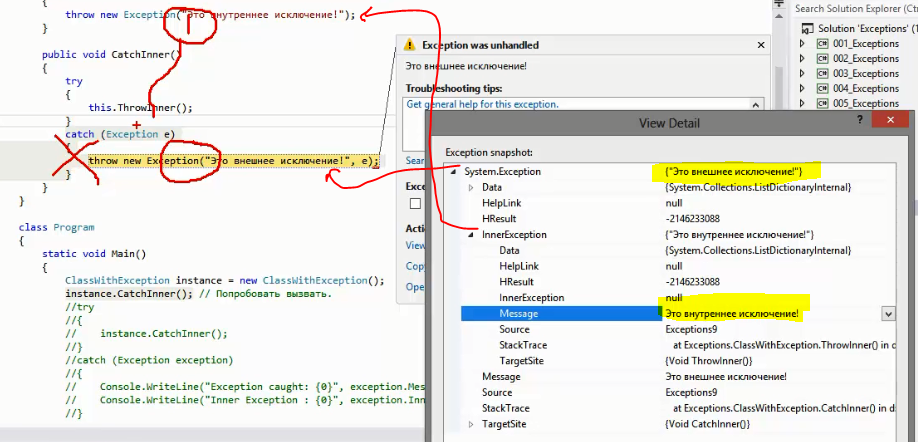
Внутреннее исключение вкладывается во внешнее исключение

СИСТЕМА ВЛОЖЕННЫХ ИСКЛЮЧЕНИЙ

Во внешнем исключении при создании по слабой ссылке экземпляра класса Exception в конструктор передаем сообщение и ВНУТРЕННЕЕ ИСКЛЮЧЕНИЕ

Если не было внутреннего исключения внешнее не отработало б

Причина внутреннее исключение, следствие внешнее исключение

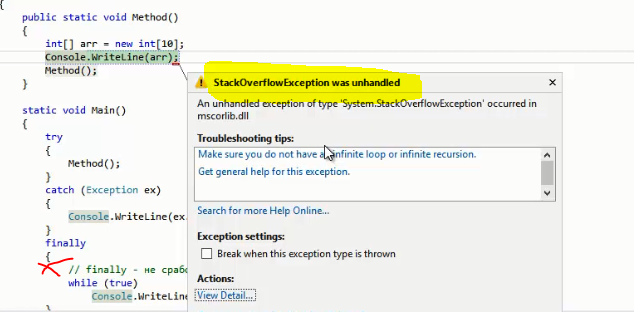


StackOverflowException – исключение которое не отрабатывает

Вызывается при переполнении стека, в данном случае создается адрес возврата и ссылка указывающая на массив который строится на куче

Начиная с платформы .NET Framework версии 2.0 объект StackOverflowException нельзя перехватывать с помощью блока try-catch и соответствующий процесс по умолчанию завершается.

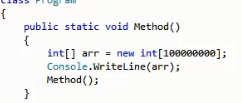
В самом эксепшене StackOverflowException нет ничего особенного. Если его выкинуть  
самому, то оно без проблем поймается. Просто CLR его выкидывает по-особому.



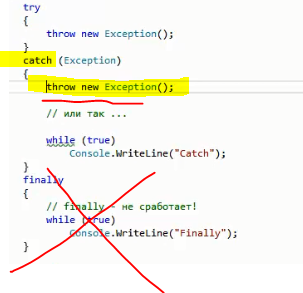
OutOfMemoryException – вызывается при переполнении кучи

Зайдет и в catch и finally

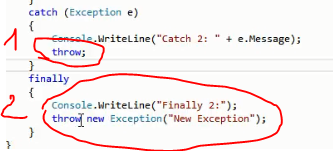
При вызове рекурсивного метода



Finally – не отработает в случае если в блоке catch бросается исключение и оно ни где в дальнейшем не будет отработано (программа завершиться ошибкой и до блока Finally не дойдет )



Если пробрасываем два исключения то пробросится из блока finally



Блоки catch ОБЯЗАНЫ СЛЕДОВАТЬ ДРУГ ЗА ДРУГОМ в обратном порядке наследованию (от производного класса к базовому)

Иначе будет ошибка на этапе компиляции

