Error handling

Exceptions

https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/exceptions-and-performance

throw / try / catch / finally

```
try
{
    int i = int.Parse("3");
}
catch(Exception e)
{
    Console.WriteLine("WrongInput");
}
finally
{
    Console.WriteLine("Garanted code");
}
```

```
var list = new List<string>() { "3", "-1", "fff" };
foreach(var s in list)
{
    try
    {
        int i = int.Parse(s);
        if (i < 0)
            throw new InvalidOperationException("Wrong situation");
    }
    catch(InvalidOperationException e)
    {
        Console.WriteLine("Wrong situation");
    }
    catch(FormatException e)
    {
        Console.WriteLine("Wrong Input");
    }
    catch(Exception e) {}
    catch {}
}
```

```
internal static int MyMethod(string s)
{
    try
    {
        int t = int.Parse(s);
        if (t < 0)
            throw new ArgumentOutOfRangeException(nameof(s));
        return t;
    }
    catch(FormatException e)
    {
        throw e; // CLR считает, что исключение возникло тут
    }
    catch(ArgumentOutOfRangeException _)
    {
        throw; // CLR не меняет информацию о начальной точке исключения.
    }
}

static void Main(string[] args)
{
    var list = new List<string>() { "3", "-1", "fff" };
```

```
    foreach(var s in list)
    {
        try
        {
            MyMethod(s);
        }
        catch(Exception e) {  Console.WriteLine(e);  }
    }
}
```

Класс Exception

```
publice class Exception
```

- Message - Gets a message that describes the current exception.
- InnerException Gets the Exception instance that caused the current exception.
- Data - Gets a collection of key/value pairs that provide additional user-defined information about the exception.
- HelpLink - Gets or sets a link to the help file associated with this exception.
- HResult - Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.
- Source - Gets or sets the name of the application or the object that causes the error.
- StackTrace - Gets a string representation of the immediate frames on the call stack.
- TargetSite Gets the method that throws the current exception.

```
static void Main(string[] args)
{
    try
    {
        throw new MyException {Status = Status.Forbidden};
    }
    catch(MyException e) when (e.Status == Status.Forbidden)
    {
        Console.WriteLine(e.ToString());
    }
}

internal class MyException: InvalidOperationException
{
    internal Status Status {get;set;}
    public override string ToString()
    {
        return $"{Status} Application error, {base.ToString()}";
    }
}

internal enum Status
{
```

```
    Forbidden = 1,
    Conflicted = 2,
    RetryFailed = 3
}
```

```
private static int TestParseWithCatch(string s)
{
    try
    {
        return int.Parse(s);
    }
    catch(FormatException e)
    {
        return 0;
    }
}

private static int TestParseWithOut(string s)
{
    if (! int.TryParse(s, out int i))
        i = 0;
    return i;
}

static void Main(string[] args)
{
    int count = 10000;
    double t = Benchmark(count, () => {int i = TestParseWithCatch("f"); });
```

```
    double t2 = Benchmark(count, () => {int i = TestParseWithOut("f"); });
    Console.WriteLine(t);                   // 0,01185658
    Console.WriteLine(t2.ToString("F8"));   // 0,00004262
}
```

```
internal static Result MyMethod(SomeParams p)
{
    if (1==1)
        throw new MyException();
    return new Result {};
}

internal static Result MyMethod2(SomeParams p, out ErrorData error)
{
    error = null;
    return new Result {};
}

internal static Tuple<Result, ErrorData> MyMethod3(SomeParams p)
{
    return new Tuple<Result, ErrorData>(new Result{}, null);
}

internal class Result {}
internal class SomeParams {}
internal class ErrorData {}
internal class MyException: Exception {}
```

Рихтер об исключениях:

> В сообществе программистов вопросы быстродействия, связанные с обработкой исключений, обсуждаются очень часто и активно. Некоторые пользователи считают эту процедуру настолько медленной, что отказываются к ней прибегать. Но я утверждаю, что для объектно-ориентированной платформы обработка исключений обязательна. Собственно, чем ее можно заменить? Неужели вы предпочтете, чтобы методы возвращали значение true или false, чтобы сообщать об успехе или неудаче своей работы? Или воспользуетесь кодом ошибок типа enum? В этом случае вы столкнетесь с худшими сторонами обоих решений. CLR и код библиотек классов будут генерировать исключения, а ваш код начнет возвращать код ошибок. В итоге вам все равно придется вернуться к необходимости обработки исключений.

> Трудно сравнивать быстродействие механизма обработки исключений и более привычных средств уведомления об исключениях (возвращения значения HRESULT, специальных кодов и т. п.). Если вы напишете код, который сам будет проверять значение, возвращаемое каждым вызванным методом, фильтровать и передавать его коду, вызвавшему метод, то быстродействие приложения серьезно снизится. Даже если оставить быстродействие в стороне, объем дополнительного кодирования и потенциальная возможность ошибок окажутся неподъемными. В такой обстановке обработка исключений выглядит намного лучшей альтернативой.

Особенности:

- Finally может не выполниться
  - Экстренное завершение приложения
  - Некоторые тяжелые ошибки в духе StackOverflowException
- CLR запрещает аварийно завершать потоки во время выполнения кода блоков catch и finally
- Если надо бросить исключение
  - Сначала поискать не подойдет ли одно из стандартных msdn
  - Свои исключения надо делать:
    - нет соотвествующего исключения в фреймворке
    - хочется обработать это исключение уникальным образом отличным от дефолтных
- При исключении в catch / finally будет поведение аналогичное, как если исключение будет брошено после блока finally, вся информация о первом исключении теряется (кроме throw;)

Debug / Trace

Logging frameworks