ОБРАБОТКА ОШИБОК. SYSTEM.OBJECT.

C# - ЛЕКЦИЯ 5

НА ПРОШЛОМ ЗАНЯТИИ

- Конструкции языка
- Контроль переполнения: checked/uncheked
- Массивы
- Список
- Словарь

СЕГОДНЯ В ПРОГРАММЕ

- Обработка ошибок
- System.Object

ОБРАБОТКА ОШИБОК

ОБРАБОТКА ОШИБОК

- Коды возврата
- Исключения

КОДЫ ВОЗРАТА (ЯЗЫК С)

```c
void fn(T1 arg1, T2 arg2)
{
    //  ...
}

int fn(T1 arg1, T2 arg2)
{
    int errorCode = 0;
    //  ...
    return errorCode;   //  0 - Ok, != 0 - Error
}
```

КОДЫ ВОЗРАТА (ЯЗЫК С)

```
TResult fn(T1 arg1, T2 arg2)
{
    //  ...
}

int fn(T1 arg1, T2 arg2, TResult* result)
{
    int errorCode = 0;
    //  ...
    return errorCode;   //  0 - Ok, != 0 - Error
}
```

```
int readToDos(ToDoItem * items, const char * filePath, size_t& limit, size_
{
    int result;
    FILE * stream = fopen(filePath, "rb");

    if (stream != null)
    {
        if (offset > 0)
            result = fseek(stream, sizeof(ToDoItem) * offset, SEEK_SET);
        if (result == 0)
            limit = fread(items, sizeof(ToDoItem), limit, stream);

        fclose(stream);
    }
    else
    {
        result = 1;
        limit = 0;
    }

    return result;
}
```

КОДЫ ВОЗРАТА - ПРОБЛЕМЫ

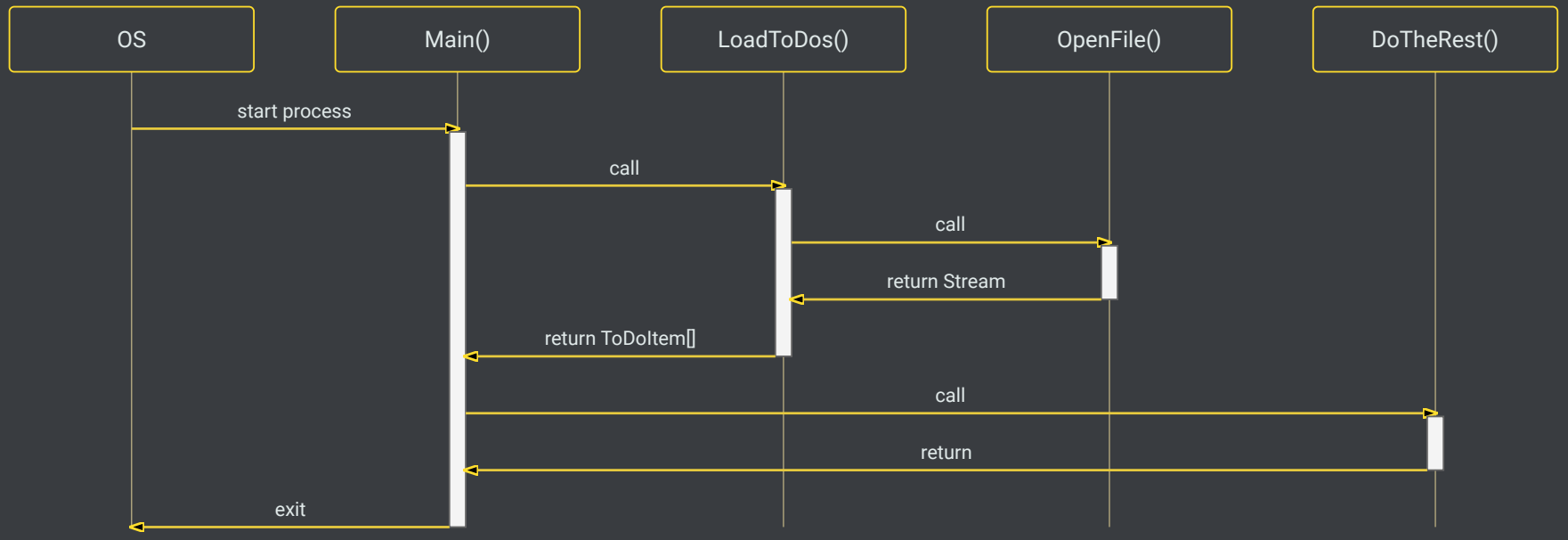
- Возвращаемое значение функции перестаёт быть результатом
- Ничто не обязывает обрабатывать ошибки
- Коды ошибок требуют документации

ИСКЛЮЧЕНИЯ

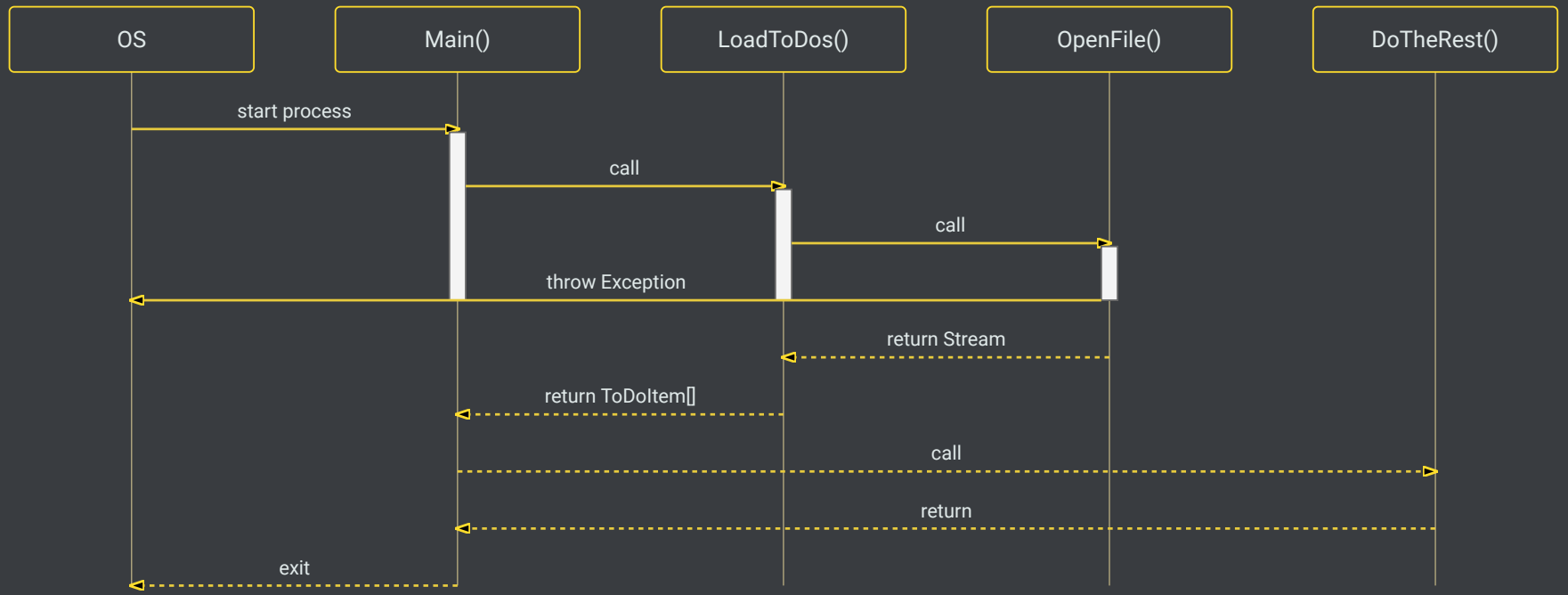
```
void fn(T1 arg1, T2 arg2)
{
    //  ... если ошибка, то генерируем исключение
}

TResult fn(T1 arg1, T2 arg2)
{
    //  ... если ошибка, то генерируем исключение
}
```

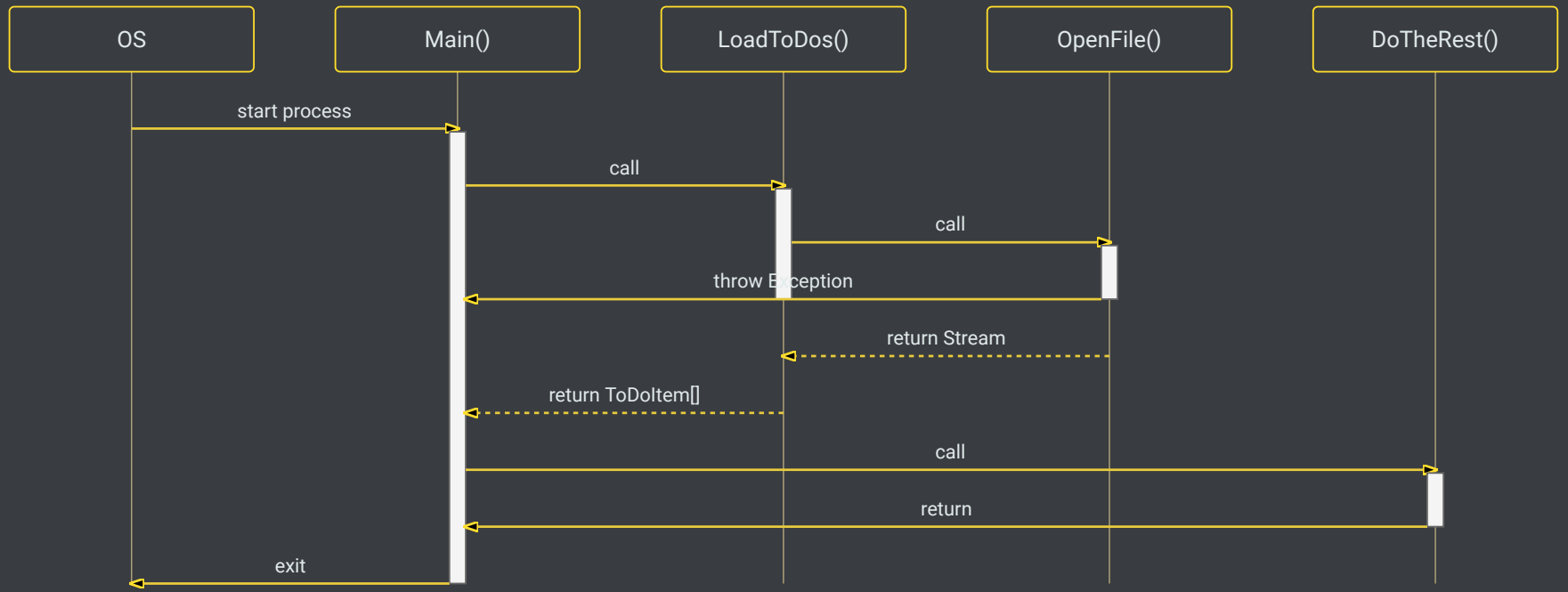
НОРМАЛЬНЫЙ ПОТОК ИСПОЛНЕНИЯ



ПОТОК ИСПОЛНЕНИЯ ПРИ ИСКЛЮЧЕНИИ



ПОТОК ИСПОЛНЕНИЯ ПРИ ОБРАБОТКЕ ИСКЛЮЧЕНИЙ



РАБОТА С ИСКЛЮЧЕНИЯМИ

- Генерация исключений
- Обработка исключений

ГЕНЕРАЦИЯ ИСКЛЮЧЕНИЙ. СИНТАКСИС

```
throw [<выражение>];
```

- опционально только в блоке catch
- выражение должно иметь тип, приводимый к System.Exception

ТИП SYSTEM.EXCEPTION

Свойства

```
public class Exception //  ...
{
    public virtual string Message { get; }
    public virtual string StackTrace { get; }
    public Exception InnerException { get; }
    public int HResult { get; protected set; }
    public MethodBase TargetSite { get; }
    public virtual IDictionary Data { get; }
    public virtual string HelpLink { get; set; }
}
```

ТИП SYSTEM.EXCEPTION

Конструкторы

```
public class Exception //  ...
{
    public Exception();
    public Exception(string message);
    public Exception(string message, Exception innerException);
}
```

ГЕНЕРАЦИЯ ИСКЛЮЧЕНИЙ

```
throw new Exception("Some error message");
```

ЧАСТО ВСТРЕЧАЮЩИЕСЯ ТИПЫ ИСКЛЮЧЕНИЙ

- `Exception`
- `NullReferenceException`
- `IndexOutOfRangeException`
- `RankException`

ЧАСТО ВСТРЕЧАЮЩИЕСЯ ТИПЫ ИСКЛЮЧЕНИЙ

- `ArgumentNullException`
- `ArgumentOutOfRangeException`
- `ArgumentException`
- `InvalidOperationException`
- `NotImplementedException`

ЧАСТО ВСТРЕЧАЮЩИЕСЯ ТИПЫ ИСКЛЮЧЕНИЙ

- `NotSupportedException`
- `FormatException`
- `OverflowException`
- `TimeoutException`

ЧАСТО ВСТРЕЧАЮЩИЕСЯ ТИПЫ ИСКЛЮЧЕНИЙ

- `FileNotFoundException`
- `DirectoryNotFoundException`
- `DriveNotFoundException`
- `PathTooLongException`
- …

ОБРАБОТКА ИСКЛЮЧЕНИЙ. СИНТАКСИС

```
try
    <инструкция>
[ catch [ (<тип_исключения1> [ <имя_1> ]) ] [ when (<логическое_выражение1
    <инструкция1> ]
[ ... ]
[ finally
    <инструкцияN> ]
```

ОБРАБОТКА ИСКЛЮЧЕНИЙ. ПРИМЕРЫ

```
Console.WriteLine("Enter an integer");
try
{
    int value = int.Parse(Console.ReadLine());
}
catch (FormatException ex)
{
    Console.WriteLine("Failed to parse input");
    Console.WriteLine(ex);
}
```

ОБРАБОТКА ИСКЛЮЧЕНИЙ. ПРИМЕРЫ

```
Enter an integer
a
Failed to parse input
System.FormatException: Input string was not in a correct format.
at System.Number.StringToNumber(ReadOnlySpan`1 str,
    NumberStyles options, NumberBuffer& number,
    NumberFormatInfo info, Boolean parseDecimal)
at System.Number.ParseInt32(ReadOnlySpan`1 s, NumberStyles style,
    NumberFormatInfo info)
at System.Int32.Parse(String s)
at ConsoleApp1.Program.Main(String[] args) in
    C:\Users\a.pavlyuk\temp\ConsoleApp1\ConsoleApp1\Program.cs:line 12
```

```
class MathUtils
{
    public static long GetFibonacci(int index)
    {
        if (index < 0)
            throw new ArgumentException(
                "Expected a non-negative value", nameof(index));

        return index > 1
            ? GetFibonacci(index - 1) + GetFibonacci(index - 2)
            : 1;
    }
}
```

```
SqlConnection conn = EstablishDbConnection();
try
{
    IEnumerable<Order> orders = QueryPlacedUnprocessedOrders(conn);
    // do something else
}
finally
{
    Disconnect(conn);
}
```

```
SqlConnection conn = EstablishDbConnection();
try
{
    IEnumerable<Order> orders = QueryPlacedUnprocessedOrders(conn);
    // do something else
}
catch (Exception ex) when (ex.HResult == 42)
{
    // log exception
    throw new DbInteractionException(ex);
}
finally
{
    Disconnect(conn);
}
```

```
SqlConnection conn = EstablishDbConnection();
try
{
    IEnumerable<Order> orders = QueryPlacedUnprocessedOrders(conn);
    // do something else
}
catch
{
    // log exception
    throw new DbInteractionException(/* ?? */);
}
finally
{
    Disconnect(conn);
}
```

```
SqlConnection conn = EstablishDbConnection();
try
{
    IEnumerable<Order> orders = QueryPlacedUnprocessedOrders(conn);
    // do something else
}
catch (Exception ex)
{
    // log exception
    throw;  // не throw ex;
}
finally
{
    Disconnect(conn);
}
```

SYSTEM.OBJECT

SYSTEM.OBJECT

- Базовый тип для всех остальных типов
- А значит, все типы приводимы к System.Object
- А значит, члены System.Object - это минимум возможностей для всех типов
- object - это класс. А значит, ссылочный тип.

ПРИВОДИМОСТЬ К SYSTEM.OBJECT

```
int i = 17;
Object io1 = i;
object io2 = i; //  object - псевдоним типа System.Object

var date = DateTime.Now;
object dateObj = date;
```

ЧЛЕНЫ SYSTEM.OBJECT

```
public class Object
{
    public static bool Equals(Object objA, Object objB);
    public static bool ReferenceEquals(Object objA, Object objB);
    public virtual bool Equals(Object obj);
    public virtual int GetHashCode();
    public Type GetType();
    public virtual string ToString();
    protected Object MemberwiseClone();
}
```

object.Equals

```
Console.WriteLine(object.Equals(42, 42));
//  true

Console.WriteLine(object.Equals((object)42, (object)42));
//  true

Console.WriteLine(
    object.Equals(
        new DateTime(2019, 3, 15),
        new DateTime(2019, 3, 15)
    )
);
//  true
```

object.ReferenceEquals

```
Console.WriteLine(object.ReferenceEquals(42, 42));
//  false

var stopwatch = new Stopwatch();
var anotherOne = new Stopwatch();
Console.WriteLine(Object.ReferenceEquals(stopwatch, anotherOne));
//  false

var sameOne = stopwatch;
Console.WriteLine(Object.ReferenceEquals(stopwatch, sameOne));
//  true
```

object.ReferenceEquals

```
string str1 = "Hello, world!";
string str2 = "Hello, world!";
Console.WriteLine(Object.ReferenceEquals(str1, str2));
//  true
//  But why? 🤔
```

Интернирование строк.

Подробнее см. в статьях string.IsInterned и string.Intern

object.Equals

```
Console.WriteLine(42.Equals(42));
//  true

Console.WriteLine(((object)42).Equals((object)42));
//  true

Console.WriteLine(
    new DateTime(2019, 3, 15).Equals(new DateTime(2019, 3, 15))
);
//  true
```

object.GetHashCode

```
Console.WriteLine(42.GetHashCode());
//  42

Console.WriteLine(new DateTime(2019, 03, 17).GetHashCode());
//  2003184235

Console.WriteLine(new System.Drawing.Point(17, 42).GetHashCode());
//  539
```

object.GetType

```
Type type = "Oranges".GetType();
Console.WriteLine(type);
//  System.String
```

object.ToString

```
//  Реализация из исходного кода System.Object
public virtual String ToString()
{
    return GetType().ToString();
}
```

`object.MemberwiseClone`

- Возвращает плоскую копию объекта
- Метод имеет область видимости protected
- Подразумевается для использования в реализации ICloneable интерфейса

УПАКОВКА И РАСПАКОВКА

Boxing & unboxing

УПАКОВКА И РАСПАКОВКА

Это механизм, обеспечивающий приводимость между типами значений и типом Object.

```
//  значение i - число 7, хранится на стеке
int i = 7;

//  упаковка: значение oi - ссылка на значение 7, хранящееся в куче
object oi = i;

//  значение i изменилось, но oi осталось прежним
i = 4;

//  распаковка: значение 7 из кучи скопировано на стек, где хранится j
int j = (int)oi;
```

ДОСТОИНСТВА

- Без этого невозможна приводимость всех типов к object
- Приводимость к object позволяет создавать максимально широкие абстракции
- А также иметь набор общих методов в object

НЕДОСТАТКИ

- Понижает эффективность: память, процессор, сборка мусора
- При работе с большими коллекциями может стать узким горлышком - List`T vs ArrayList

ВОПРОСЫ