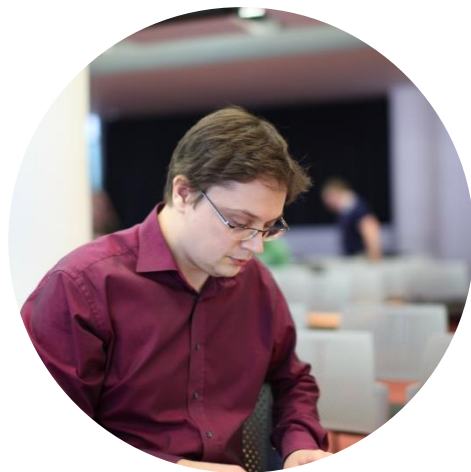





Исключительные ситуации



- WEB/WPF/WinForms/... стеки
- C/C++, C++/CLI когда необходимо
- Книга:  <https://github.com/sidristij/dotnetbook>



ОСНОВЫ ОСНОВ

try-catch|when-finally

```
try {  
    // 1  
} catch (ArgumentOutOfRangeException exception)  
{  
    // 2  
} catch (IOException exception)  
{  
    // 3  
} catch  
{  
    // 4  
} finally {  
    // 5  
}
```

try-**catch** | when-finally

```
try {  
    // 1  
} catch (ArgumentOutOfRangeException exception)  
{  
    // 2  
} catch (IOException exception)  
{  
    // 3  
} catch  
{  
    // 4  
} finally {  
    // 5  
}
```

try-catch|when-finally

```
try {  
    //...  
}  
catch (SomeException exception)  
{  
    switch(exception.ErrorCode)  
    {  
        case ErrorCode.NetworkDown:  
            // ...  
            break;  
        case ErrorCode.CacheDown:  
            // ...  
            break;  
        default:  
            throw;  
    }  
}
```

```
try {  
    //...  
}  
catch (SomeException exception) when (exception.ErrorCode == ErrorCode.NetworkDown)  
{  
    // ...  
}  
catch (SomeException exception) when (exception.ErrorCode == ErrorCode.CacheDown)  
{  
    // ...  
}
```

try-catch|when-**finally**

```
try {  
    // 1  
} catch (ArgumentOutOfRangeException exception)  
{  
    // 2  
} catch (IOException exception)  
{  
    // 3  
} catch  
{  
    // 4  
} finally {  
    // 5  
}
```

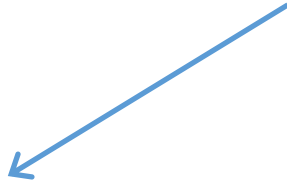
Архитектурные вопросы

Архитектура исключительной ситуации:

1. По теоретической возможности перехвата
2. По вопросам переиспользования
3. По отношению к единой группе поведенческих ситуаций
4. По источнику ошибки

По теоретической возможности перехвата

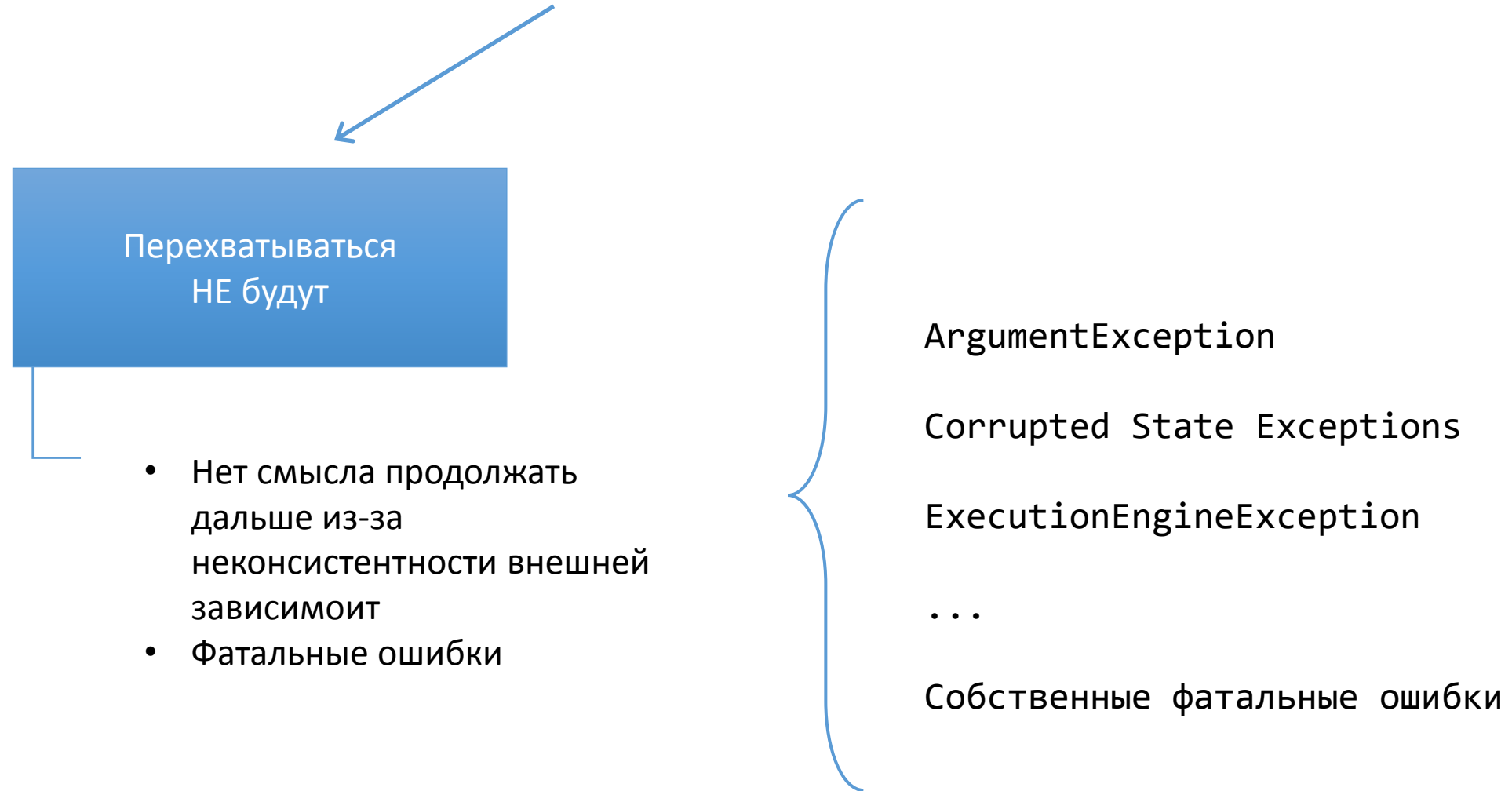
По теоретической возможности перехвата



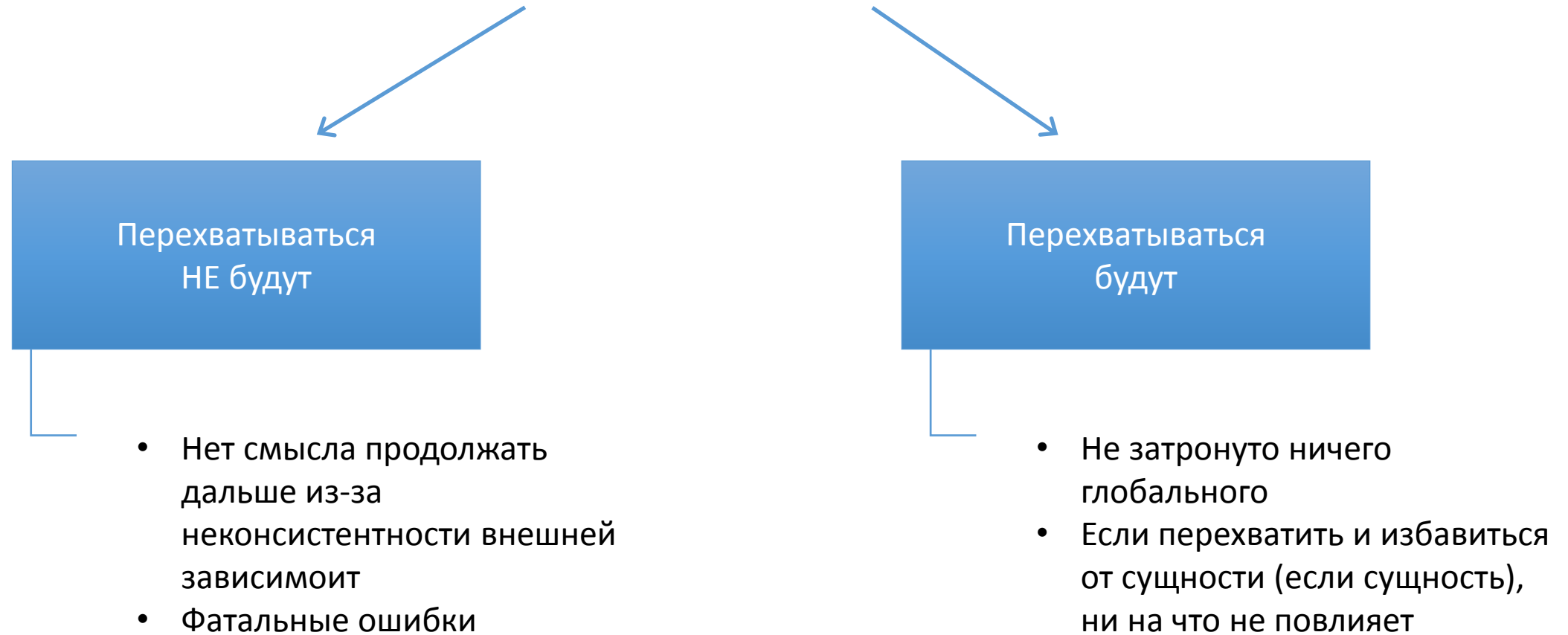
Перехватываться
НЕ будут

- Нет смысла продолжать дальше из-за неконсистентности внешней зависимости
- Фатальные ошибки

По теоретической возможности перехвата



По теоретической возможности перехвата



По вопросам переиспользования

1. Common исключения

1. `InvalidArgumentException`

По вопросам переиспользования

1. Common исключения

1. `InvalidArgumentException`
2. `InvalidDataException?`

По вопросам переиспользования

Microsoft .NET

5 instantiations of InvalidDataException

System (5)

sys\System\IO\compression\Inflater.cs (5)

595 throw new InvalidDataException();
602 throw new InvalidDataException();
613 throw new InvalidDataException();
624 throw new InvalidDataException();
650 throw new InvalidDataException();

16 references to InvalidDataException

System (6)

net\System\Net\HttpWebResponse.cs (6)

763 if(e is System.IO.InvalidDataException || e is InvalidOperationException)
785 if(e is System.IO.InvalidDataException || e is InvalidOperationException)
814 if(e is System.IO.InvalidDataException || e is InvalidOperationException)
869 if(e is System.IO.InvalidDataException || e is InvalidOperationException)
891 if(e is System.IO.InvalidDataException || e is InvalidOperationException)
919 if(e is System.IO.InvalidDataException || e is InvalidOperationException)

System.ServiceModel (10)

System\ServiceModel\Channels\BinaryMessageEncoder.cs (1)

642 catch (InvalidDataException)

System\ServiceModel\Channels\FramingDecoders.cs (9)

448 protected Exception CreateException(InvalidDataException innerExc
455 protected Exception CreateException(InvalidDataException innerExc
520 catch (InvalidDataException e)
819 catch (InvalidDataException e)
955 catch (InvalidDataException e)
1148 catch (InvalidDataException e)
1256 catch (InvalidDataException e)
1522 catch (InvalidDataException e)
1646 catch (InvalidDataException e)

Microsoft Reference Source .NET Framework 4.7.2

```
1 namespace System.IO {  
2  
3     using System;  
4     using System.Runtime.Serialization;  
5  
6     #if !FEATURE_NETCORE  
7         [Serializable]  
8     #endif // !FEATURE_NETCORE  
9     public sealed class InvalidDataException : SystemException  
10    {  
11        public InvalidDataException ()  
12            : base(SR.GetString(SR.GenericInvalidData)) {  
13        }  
14  
15        public InvalidDataException (String message)  
16            : base(message) {  
17        }  
18  
19        public InvalidDataException (String message, Exception innerException)  
20            : base(message, innerException) {  
21        }  
22  
23        #if !FEATURE_NETCORE  
24            internal InvalidDataException (SerializationInfo info, StreamingContext  
25            )  
26        #endif // !FEATURE_NETCORE  
27    }  
28 }  
29  
30
```


По вопросам переиспользования

1. Common исключения

1. `InvalidArgumentException`

~~2. `System.IO.InvalidDataException`~~

3. Относятся к текущему домену

По вопросам переиспользования

1. Common исключения

1. `InvalidArgumentException`

2. Относятся к текущему домену

2. `Exception.ErrorCode` – явно лучше чем парсинг Message

```
public class ParserException
{
    public ParserError ErrorCode { get; }

    public ParserException(ParserError errorCode)
    {
        ErrorCode = errorCode;
    }
}
```

```
public class ParseException
{
    public ParserError ErrorCode { get; }

    public ParseException(ParserError errorCode)
    {
        ErrorCode = errorCode;
    }

    public override string Message
    {
        get {
            return Resources.GetResource($"{nameof(ParseException)}{Enum.GetName(typeof(ParserError), ErrorCode)}");
        }
    }
}

public enum ParserError
{
    MissingModifier,
    MissingBracket,
    // ...
}

// Usage
throw new ParseException(ParserError.MissingModifier);
```

По вопросам переиспользования

```
try {  
    //...  
}  
catch (SomeException exception)  
{  
    switch(exception.ErrorCode)  
    {  
        case ErrorCode.NetworkDown:  
            // ...  
            break;  
        case ErrorCode.CacheDown:  
            // ...  
            break;  
        default:  
            throw;  
    }  
}
```

```
try {  
    //...  
}  
catch (SomeException exception) when (exception.ErrorCode == ErrorCode.NetworkDown)  
{  
    // ...  
}  
catch (SomeException exception) when (exception.ErrorCode == ErrorCode.CacheDown)  
{  
    // ...  
}
```

По вопросам переиспользования

1. Common исключения
 1. InvalidArgumentException
 2. Относятся к текущему домену
2. Exception.ErrorCode
 1. Mixed режим

```
public abstract class ParserException
{
    public abstract ParserError ErrorCode { get; }

    public override string Message
    {
        get {
            return Resources.GetResource($"{nameof(ParserException)}{Enum.GetName(typeof(ParserError), ErrorCode)}");
        }
    }
}

public enum ParserError
{
    MissingModifier,
    MissingBracket
}

public class MissingModifierParserException : ParserException
{
    public override ParserError ErrorCode { get; } => ParserError.MissingModifier;
}

public class MissingBracketParserException : ParserException
{
    public override ParserError ErrorCode { get; } => ParserError.MissingBracket;
}

// Usage
throw new MissingModifierParserException(ParserError.MissingModifier);
```

По вопросам переиспользования

1. Common исключения
 1. InvalidArgumentException
 2. Относятся к текущему домену
2. Exception.ErrorCode
 1. Mixed режим
3. По исключению на ситуацию

По вопросам переиспользования

1. Common исключения

1. `InvalidArgumentException`

2. Относятся к текущему домену

2. `Exception.ErrorCode`

1. `Mixed` режим

3. По исключению на ситуацию

1. Поломка не сущности, а работы метода: отдельное исключение, желательно не возможное к перехвату по базовому классу вызывающей стороной

По отношению к единой группе поведенческих ситуаций

1. Объединяющая целый функционал

- Сборка приложения или группа сборок, если они образуют одну функцию

По отношению к единой группе поведенческих ситуаций

1. Объединяющая целый функционал
 - Сборка приложения или группа сборок, если они образуют одну функцию
2. По предполагаемости перехвата исключений
 - Разделяя эти две группы вы упрощаете перехват

По отношению к единой группе поведенческих ситуаций

1. Объединяющая целый функционал

- Сборка приложения или группа сборок, если они образуют одну функцию

2. По предполагаемости перехвата исключений

- Разделяя эти две группы вы упрощаете перехват
 - Argument*Exceptions
 - Prerequests Exceptions
 - Fatal Error Exceptions

По отношению к единой группе поведенческих ситуаций

1. Объединяющая целый функционал

- Сборка приложения или группа сборок, если они образуют одну функцию

2. По предполагаемости перехвата исключений

- Разделяя эти две группы вы упрощаете перехват
 - `Argument*Exceptions`
 - `Prerequests Exceptions`
 - `Fatal Error Exceptions`

3. По функциональным зонам библиотеки

По отношению к единой группе поведенческих ситуаций

1. Объединяющая целый функционал
 - Сборка приложения или группа сборок, если они образуют одну функцию
2. По предполагаемости перехвата исключений
 - Разделяя эти две группы вы упрощаете перехват
 - `Argument*Exceptions`
 - `Prerequests Exceptions`
 - `Fatal Error Exceptions`
3. По функциональным зонам библиотеки
4. Собственные исключительные ситуации

По источнику ошибки

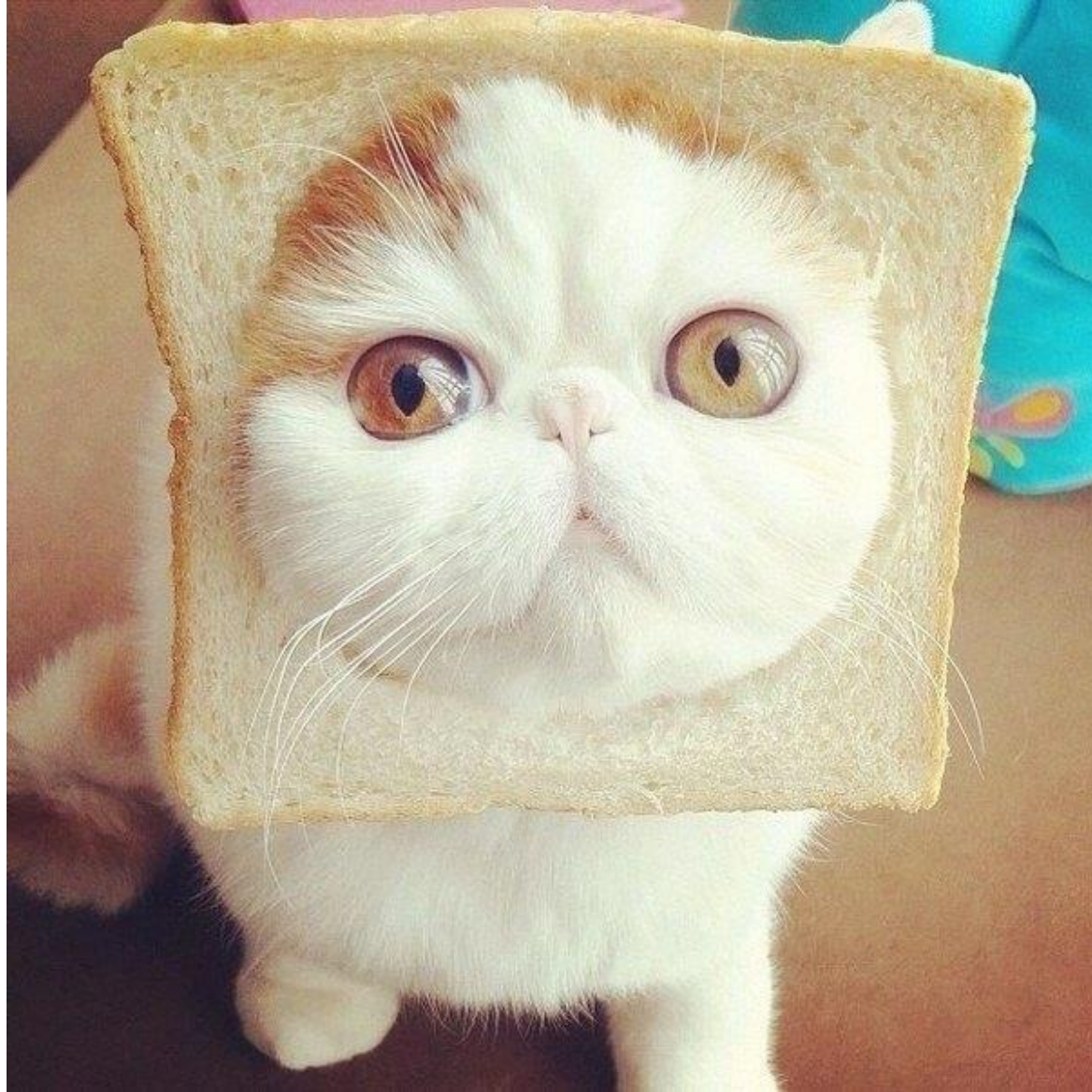
1. Вызов `unsafe` кода, который отработал с ошибкой

По источнику ошибки

1. Вызов `unsafe` кода, который отработал с ошибкой
2. Вызов кода из внешних зависимостей

По источнику ошибки

1. Вызов `unsafe` кода, который отработал с ошибкой
2. Вызов кода из внешних зависимостей
3. Наш собственный код



Промежуточные выводы

Выводы

проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;
4. Все проверки параметров проводить под `Conditional("DEBUG")`

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;
4. Все проверки параметров проводить под `Conditional("DEBUG")`
5. Разделить домен на функциональные зоны: `CachingException`, `InternalDatabaseException`, `ParsingException`

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;
4. Все проверки параметров проводить под Conditional("DEBUG")
5. Разделить домен на функциональные зоны: CachingException, InternalDatabaseException, ParsingException
6. Частные исключения наследовать от типов функциональных зон

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;
4. Все проверки параметров проводить под Conditional(“DEBUG”)
5. Разделить домен на функциональные зоны: CachingException, InternalDatabaseException, ParsingException
6. Частные исключения наследовать от типов функциональных зон
7. Если группа частных исключений может быть объединена, объединить их еще одним базовым типом.

Выводы проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;
4. Все проверки параметров проводить под Conditional(“DEBUG”)
5. Разделить домен на функциональные зоны: CachingException, InternalDatabaseException, ParsingException
6. Частные исключения наследовать от типов функциональных зон
7. Если группа частных исключений может быть объединена, объединить их еще одним базовым типом.
8. Если предполагается что группа будет чаще перехватываться по своему базовому классу, ввести Mixed Mode с ErrorCode.

Выводы

проектирование

1. Для начала необходимо сделать базовый класс для домена. Назовем его доменным базовым классом;
2. Далее необходимо ввести дополнительный базовый класс для исключений, которые перехватывать необходимо;
3. Все исключения которые обозначают фатальные ошибки – наследовать напрямую от доменного базового класса;
4. Все проверки параметров проводить под Conditional(“DEBUG”)
5. Разделить домен на функциональные зоны: CachingException, InternalDatabaseException, ParsingException
6. Частные исключения наследовать от типов функциональных зон
7. Если группа частных исключений может быть объединена, объединить их еще одним базовым типом.
8. Если предполагается что группа будет чаще перехватываться по своему базовому классу, ввести Mixed Mode с ErrorCode.

Выводы выброс

1. Исключение из внешней зависимости

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации
 2. А потому – перехватывать

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации
 2. А потому – перехватывать
3. Исключение из внутренней зависимости, ситуация неисправима

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации
 2. А потому – перехватывать
3. Исключение из внутренней зависимости, ситуация неисправима
 1. Внешний код не в курсе ситуации

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации
 2. А потому – перехватывать
3. Исключение из внутренней зависимости, ситуация неисправима
 1. Внешний код не в курсе ситуации
 2. Исправить ситуацию вы не можете

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации
 2. А потому – перехватывать
3. Исключение из внутренней зависимости, ситуация неисправима
 1. Внешний код не в курсе ситуации
 2. Исправить ситуацию вы не можете
 3. А потому – обернуть полученное исключение собственным, поместив полученное в `InnerException`

Выводы выброс

1. Исключение из внешней зависимости
 1. Нет возможности исправить ситуацию
 2. Нет полной картины происходящего
 3. А потому – не перехватывать
2. Исключение из внутренней зависимости, ситуация исправима
 1. Внешний код не в курсе ситуации
 2. А потому – перехватывать
3. Исключение из внутренней зависимости, ситуация неисправима
 1. Внешний код не в курсе ситуации
 2. Исправить ситуацию вы не можете
 3. А потому – обернуть полученное исключение собственным, поместив полученное в `InnerException`
4. Возникло не консистентное состояние

QA

