Одиннадцатая независимая научно-практическая конференция «Разработка ПО 2015»

22 - 24 октября, Москва

Аспектная разметка кода для быстрой навигации по проекту

М.С. Малеванный

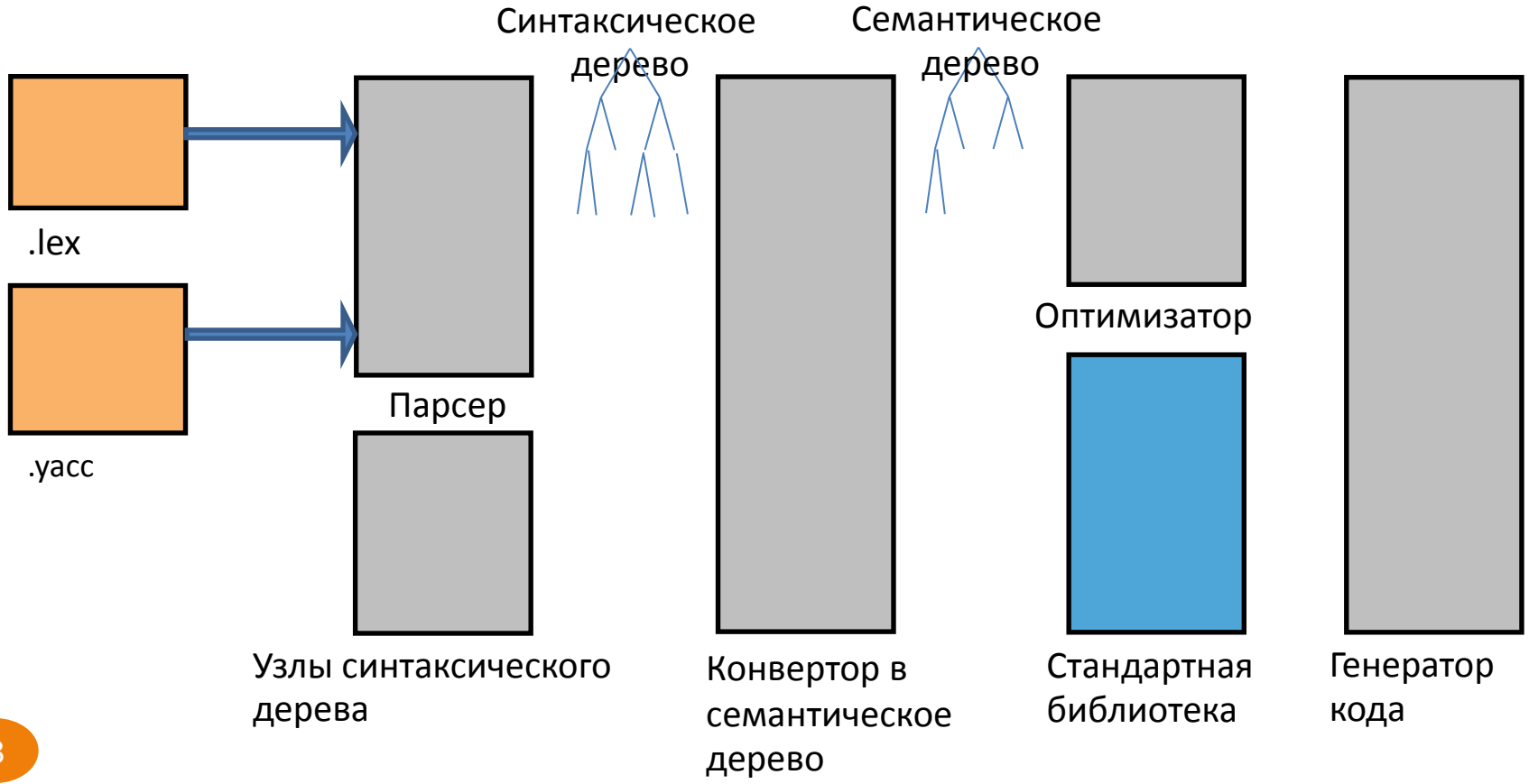
Южный федеральный университет, Институт математики, механики и компьютерных наук им. И. И. Воровича

Рабочее множество фрагментов кода

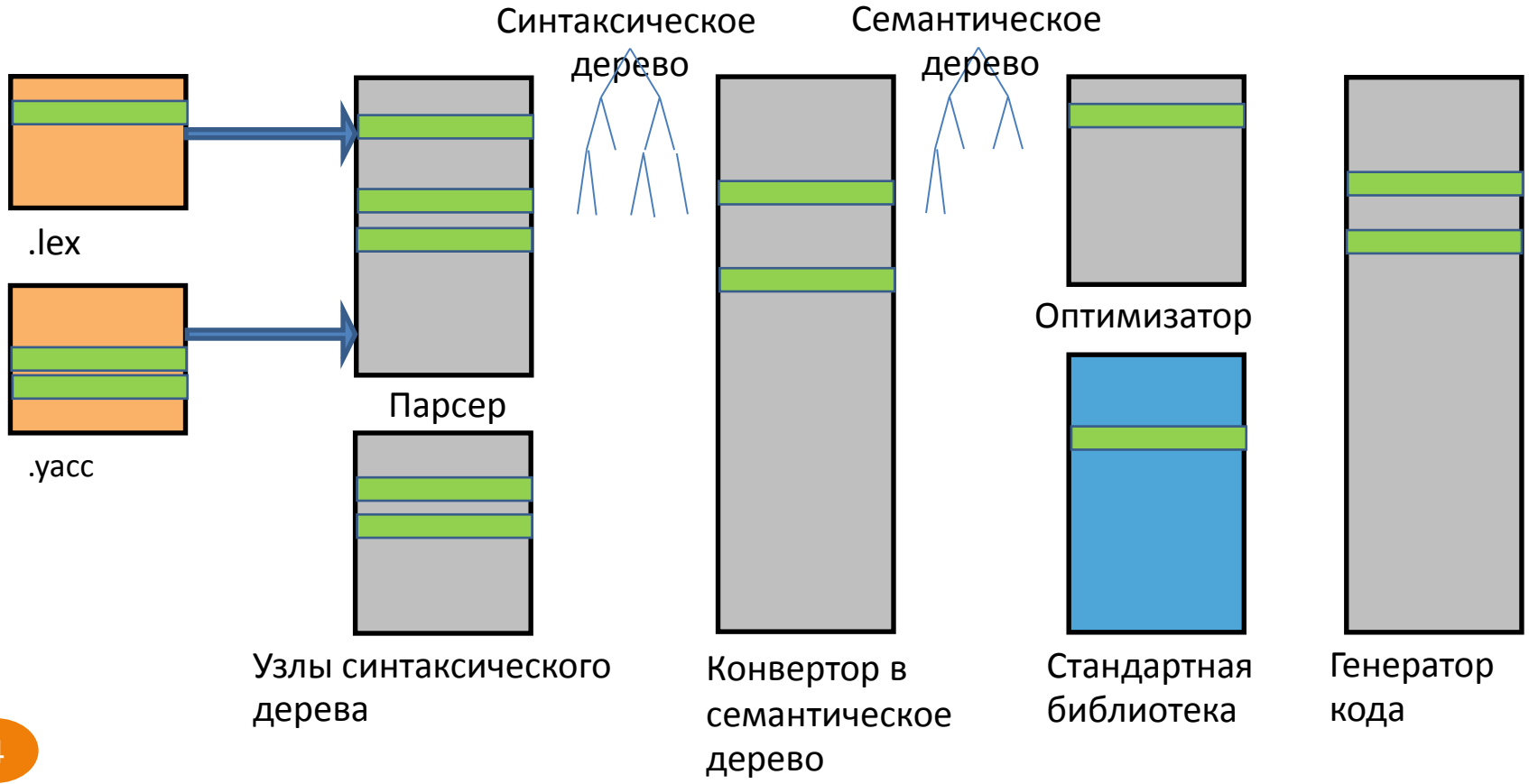
Множество фрагментов кода, связанных с решаемой задачей

- Фрагменты разбросаны по проекту
- Навигация по ним требует времени и усилий
- Не является признаком плохого проектирования

Архитектура компилятора

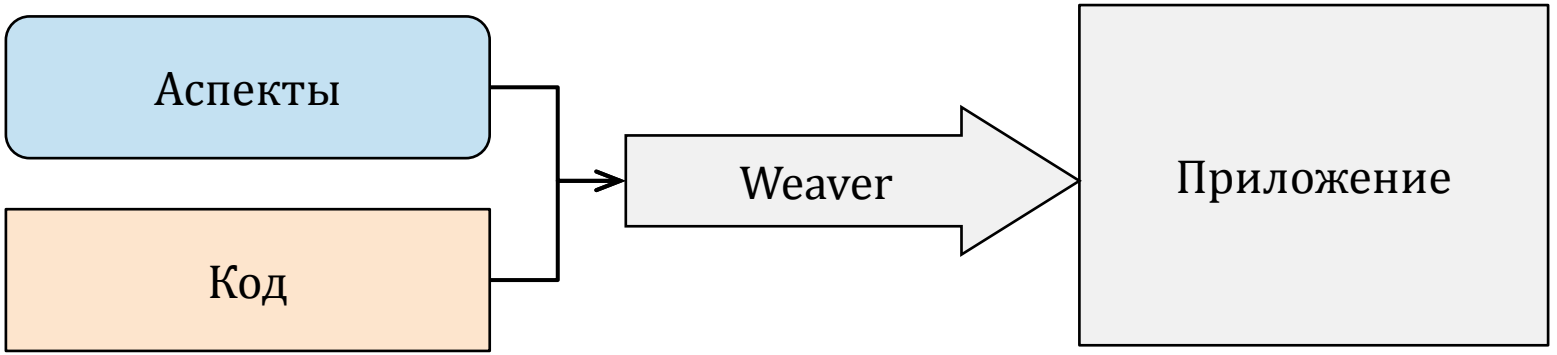


Сквозная функциональность



Аспект

- В Аспектно-Ориентированном Программировании: Модуль, реализующий сквозную функциональность



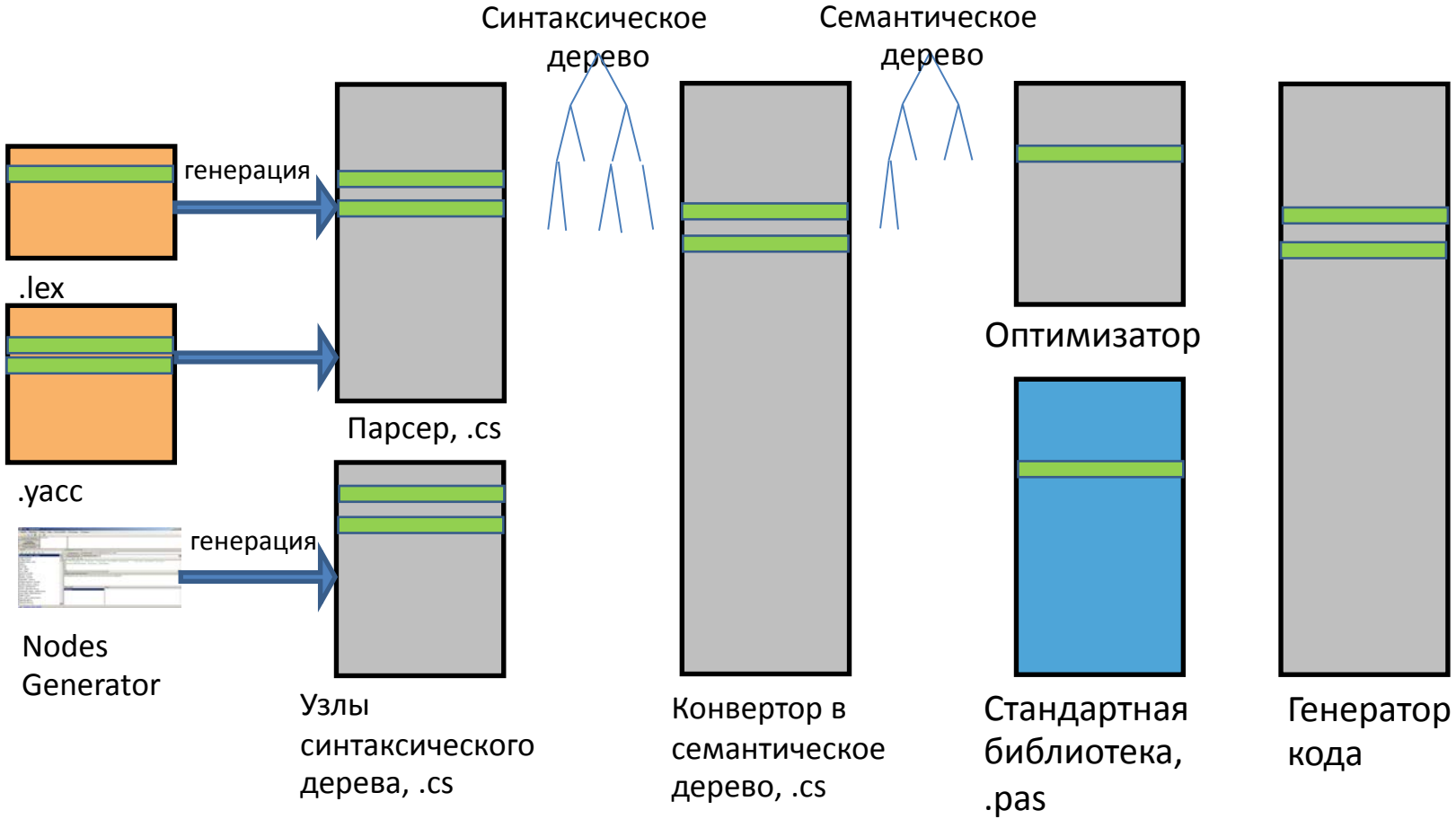
Аспект

В общем смысле:

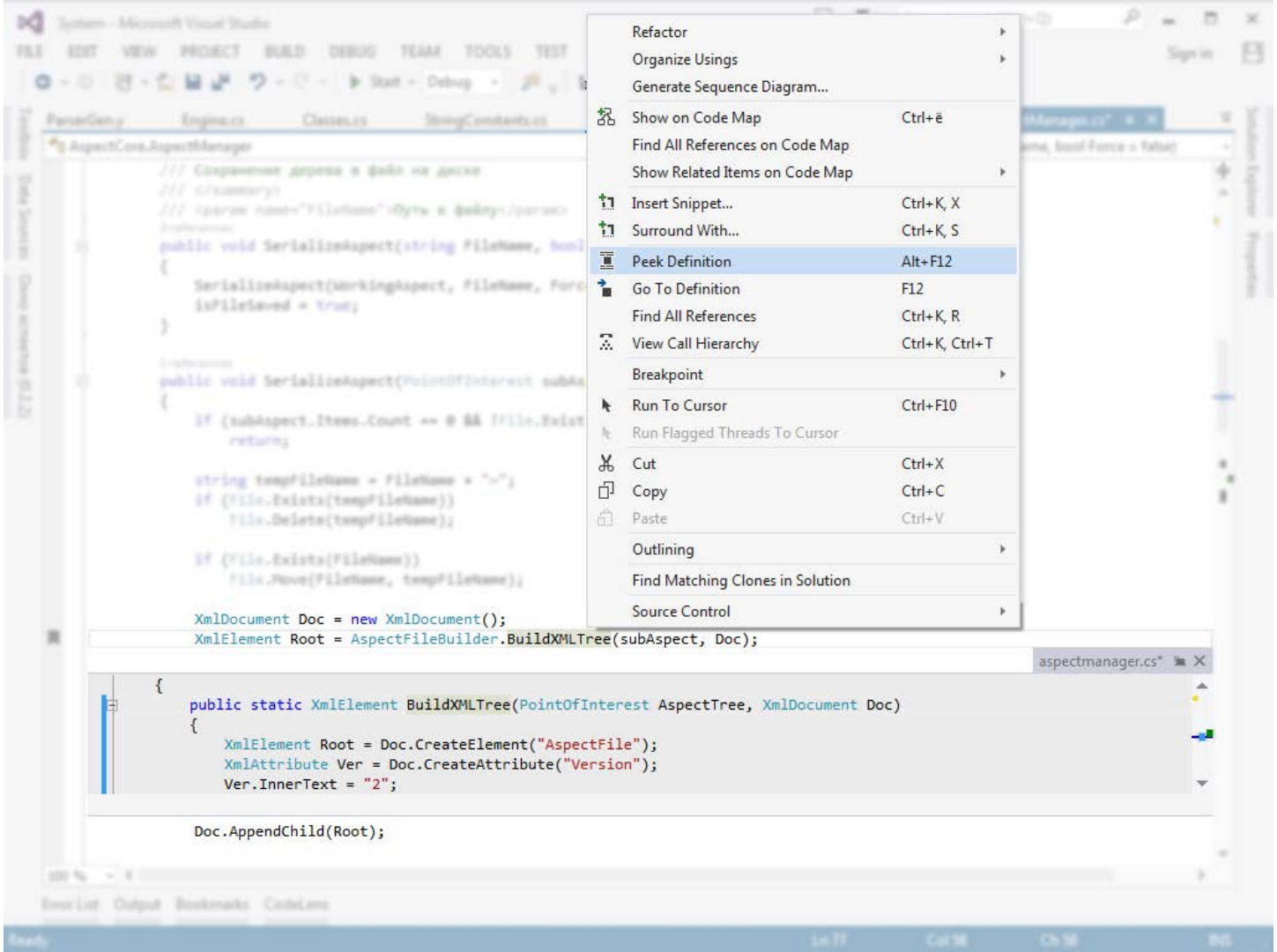
Множество логически связанных фрагментов кода


Аспекты
Код
Vs.
Код с аспектами

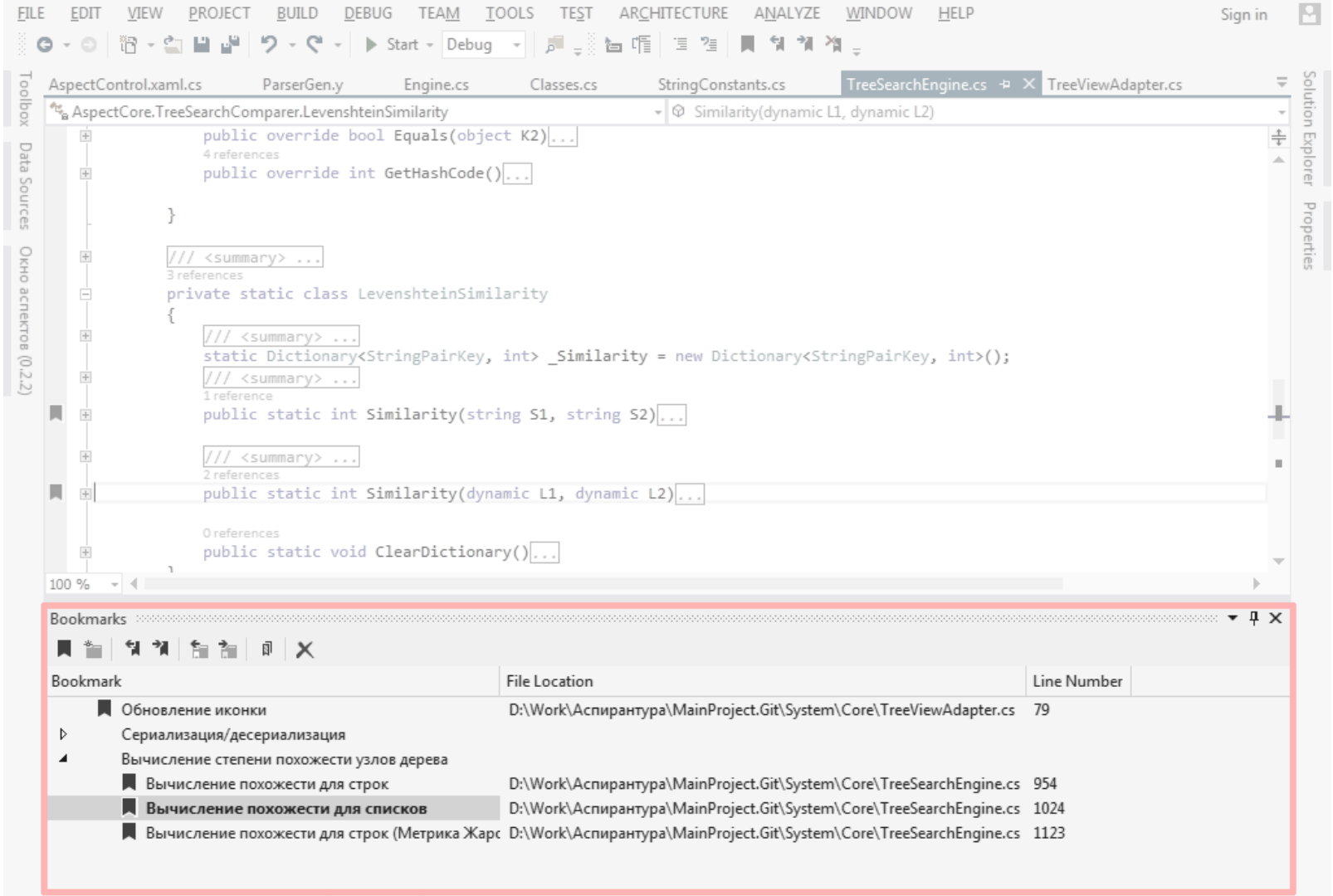
Сквозная функциональность в PascalABC.NET




Аспекты
Код
Vs.
Код с аспектами


Refactor
Organize Usings
Generate Sequence Diagram...
Show on Code Map Ctrl+ë
Find All References on Code Map
Show Related Items on Code Map
Insert Snippet... Ctrl+K, X
Surround With... Ctrl+K, S
Peek Definition Alt+F12
Go To Definition F12
Find All References Ctrl+K, R
View Call Hierarchy Ctrl+K, Ctrl+T
Breakpoint
Run To Cursor Ctrl+F10
Run Flagged Threads To Cursor
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Outlining
Find Matching Clones in Solution
Source Control
XmlDocument Doc = new XmlDocument();
XmlElement Root = AspectFileBuilder.BuildXMLTree(subAspect, Doc);
aspectmanager.cs*
{
public static XmlElement BuildXMLTree(PointOfInterest AspectTree, XmlDocument Doc)
{
XmlElement Root = Doc.CreateElement("AspectFile");
XmlAttribute Ver = Doc.CreateAttribute("Version");
Ver.InnerText = "2";
Doc.AppendChild(Root);

А с чем я работал 2 недели назад?

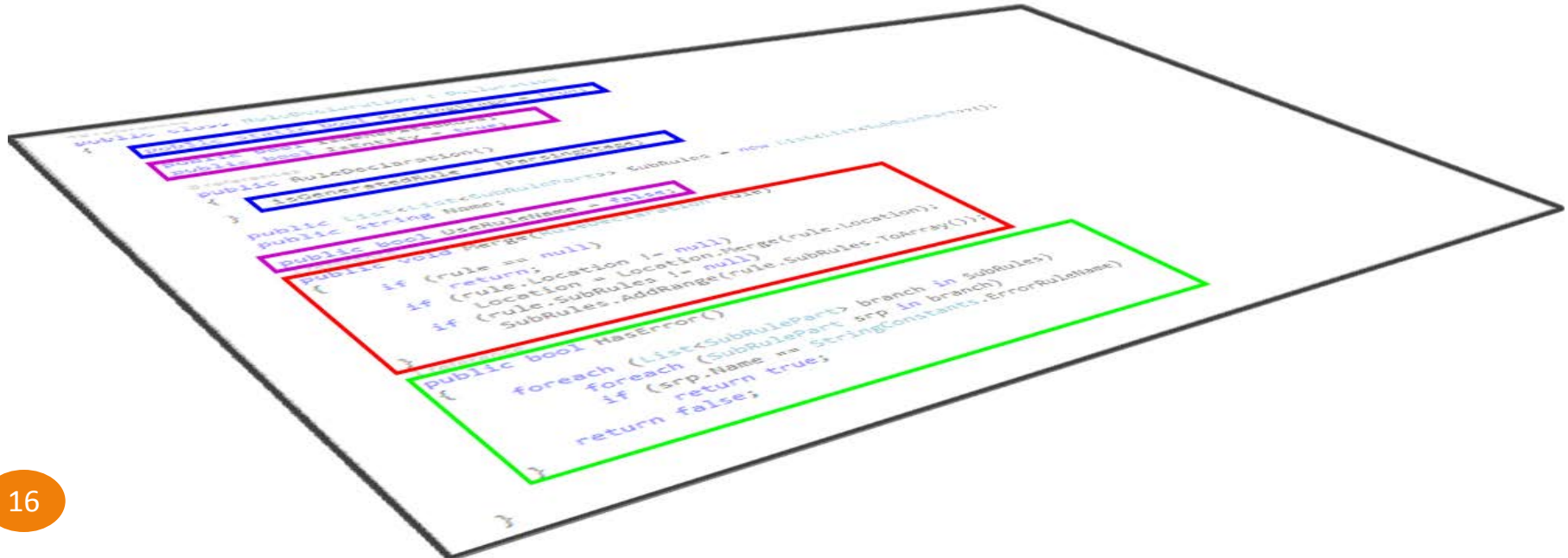

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Sign in
Start
Debug
AspectControl.xaml.cs
ParserGen.y
Engine.cs
Classes.cs
StringConstants.cs
TreeSearchEngine.cs
TreeViewAdapter.cs
AspectCore.TreeSearchComparer.LevenshteinSimilarity
Similarity(dynamic L1, dynamic L2)
public override bool Equals(object K2)...
4 references
public override int GetHashCode()...
}
/// <summary> ...
3 references
private static class LevenshteinSimilarity
{
/// <summary> ...
static Dictionary<StringPairKey, int> _Similarity = new Dictionary<StringPairKey, int>();
/// <summary> ...
1 reference
public static int Similarity(string S1, string S2)...
/// <summary> ...
2 references
public static int Similarity(dynamic L1, dynamic L2)...
0 references
public static void ClearDictionary()...
100 %
Bookmarks
Bookmark
File Location
Line Number
Обновление иконки
D:\Work\Аспирантура\MainProject.Git\System\Core\TreeViewAdapter.cs
79
Сериализация/десериализация
Вычисление степени похожести узлов дерева
Вычисление похожести для строк
D:\Work\Аспирантура\MainProject.Git\System\Core\TreeSearchEngine.cs
954
Вычисление похожести для списков
D:\Work\Аспирантура\MainProject.Git\System\Core\TreeSearchEngine.cs
1024
Вычисление похожести для строк (Метрика Жарс
D:\Work\Аспирантура\MainProject.Git\System\Core\TreeSearchEngine.cs
1123
Toolbox
Data Sources
Окно аспектов (0.2.2)
Solution Explorer
Properties

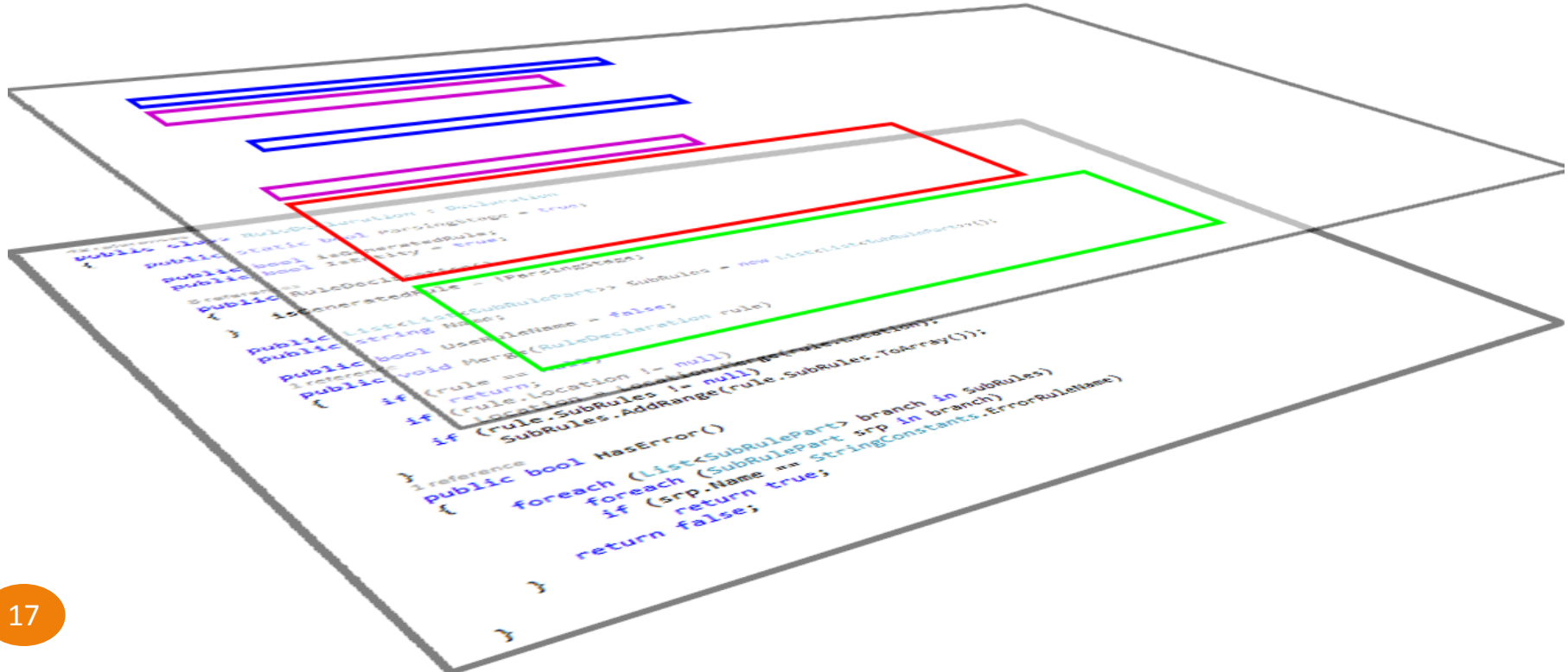
Аспектная разметка кода

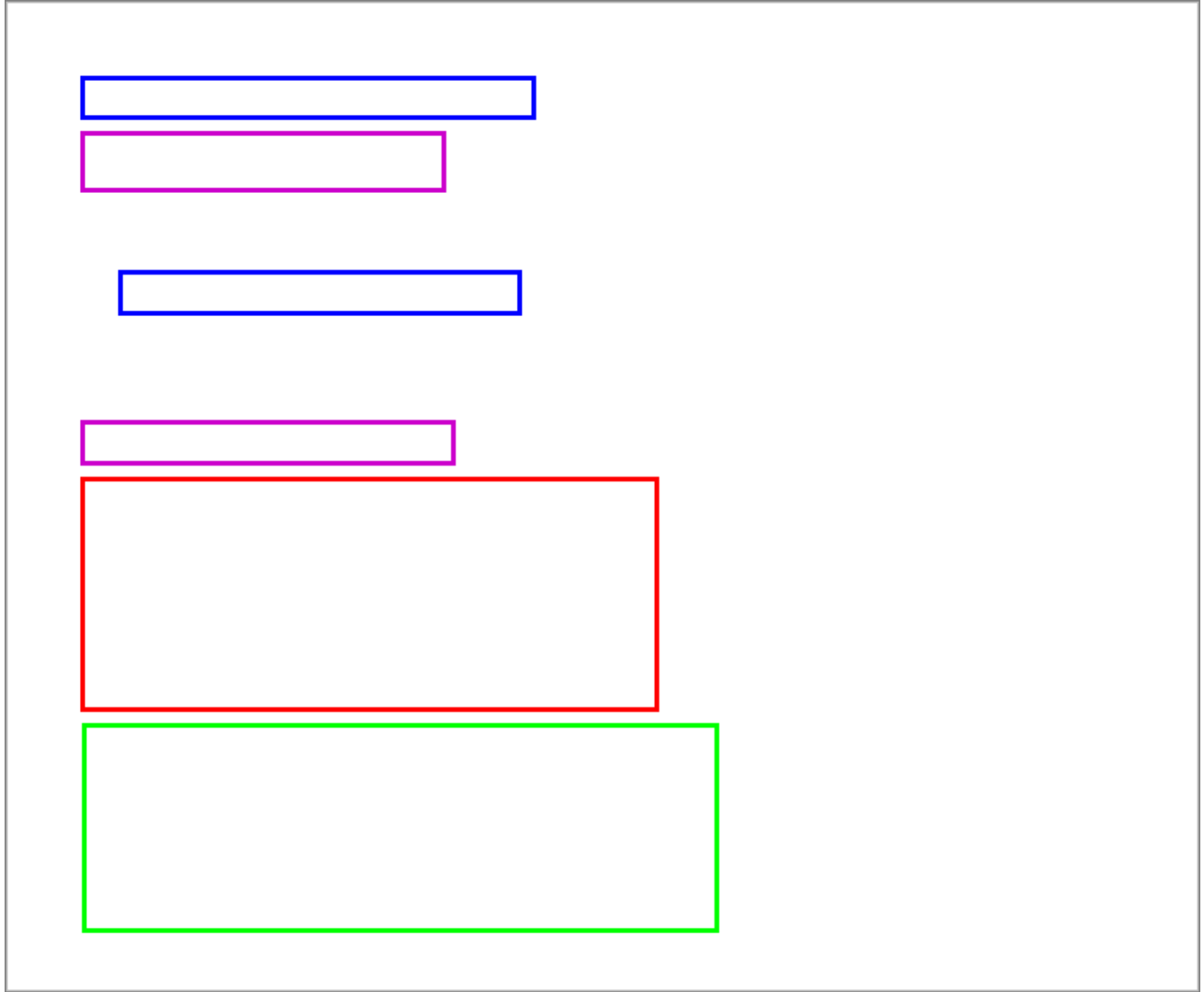
Отдельный слой метаинформации о коде

```
48 references
public class RuleDeclaration : Declaration
{
    public static bool ParsingStage = true;

    public bool isGeneratedRule;
    public bool IsEntity = true;

    8 references
    public RuleDeclaration()
    {
        isGeneratedRule = !ParsingStage;
    }

    public List<List<SubRulePart>> SubRules = new List<List<SubRulePart>>();
    public string Name;

    public bool UseRuleName = false;
    1 reference
    public void Merge(RuleDeclaration rule)
    {
        if (rule == null)
            return;
        if (rule.Location != null)
            Location = Location.Merge(rule.Location);
        if (rule.SubRules != null)
            SubRules.AddRange(rule.SubRules.ToArray());
    }
    1 reference
    public bool HasError()
    {
        foreach (List<SubRulePart> branch in SubRules)
            foreach (SubRulePart srp in branch)
            if (srp.Name == StringConstants.ErrorRuleName)
                return true;
        return false;
    }

}
```

```
48 references
public class RuleDeclaration : Declaration
{
    public static bool ParsingStage = true;

    public bool isGeneratedRule;
    public bool IsEntity = true;

    8 references
    public RuleDeclaration()
    {
        isGeneratedRule = !ParsingStage;
    }

    public List<List<SubRulePart>> SubRules = new List<List<SubRulePart>>();
    public string Name;

    public bool UseRuleName = false;
    1 reference
    public void Merge(RuleDeclaration rule)
    {
        if (rule == null)
            return;
        if (rule.Location != null)
            Location = Location.Merge(rule.Location);
        if (rule.SubRules != null)
            SubRules.AddRange(rule.SubRules.ToArray());
    }
    1 reference
    public bool HasError()
    {
        foreach (List<SubRulePart> branch in SubRules)
            foreach (SubRulePart srp in branch)
            if (srp.Name == StringConstants.ErrorRuleName)
                return true;
        return false;
    }

}
```


public bool HasError()
{
foreach (List<SubRulePart> branch in subRules)
foreach (SubRulePart srp in branch)
if (srp.Name == StringConstants.ErrorRuleName)
return true;
return false;
}
}


public bool HasError()
{
foreach (List<SubRulePart> branch in SubRules)
foreach (SubRulePart srp in branch)
if (srp.Name == StringConstants.ErrorRuleName)
return true;
return false;
}
}
}



Слияние

Работает на этапе разбора исходного текста и построения внутреннего представления

Обработка ошибок

#Флаг_парсинга

#Флаг_парсинга

#Генерация_правил

#Генерация_правил

Слияние

Работает на этапе разбора исходного текста и построения внутреннего представления

Обработка ошибок

#Флаг_парсинга

#Флаг_парсинга

#Генерация_правил

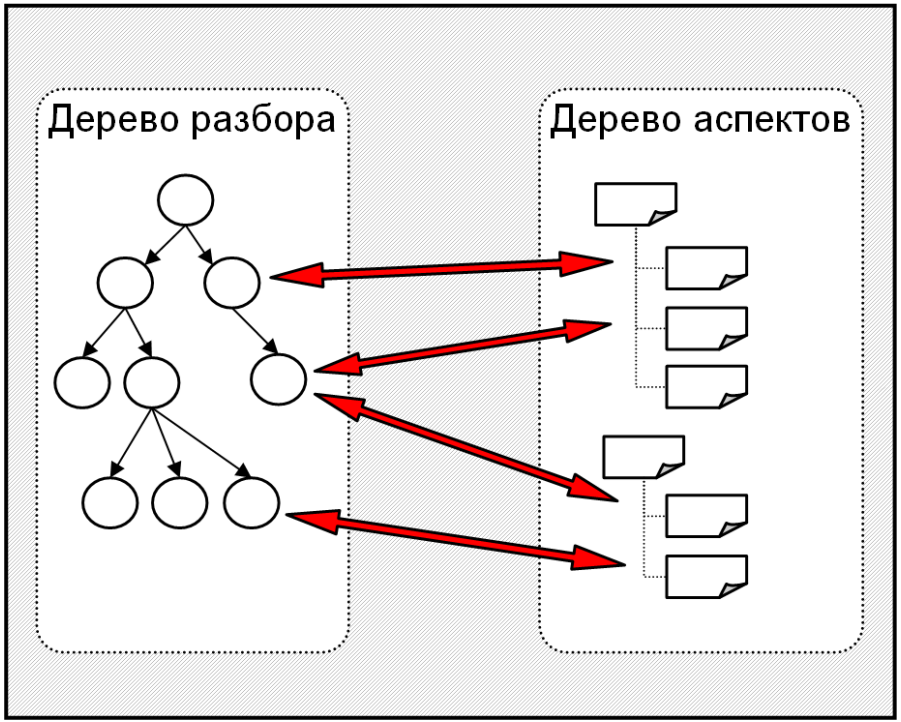
#Генерация_правил

```
48 references
public class RuleDeclaration : Declaration
{
    public static bool ParsingStage = true;

    public bool isGeneratedRule;
    public bool IsEntity = true;

    8 references
    public RuleDeclaration()
    {
        isGeneratedRule = !ParsingStage;
    }

    public List<List<SubRulePart>> SubRules = new List<List<SubRulePart>>();
    public string Name;

    public bool UseRuleName = false;
    1 reference
    public void Merge(RuleDeclaration rule)
    {
        if (rule == null)
            return;
        if (rule.Location != null)
            Location = Location.Merge(rule.Location);
        if (rule.SubRules != null)
            SubRules.AddRange(rule.SubRules.ToArray());
    }
    1 reference
    public bool HasError()
    {
        foreach (List<SubRulePart> branch in SubRules)
            foreach (SubRulePart srp in branch)
            if (srp.Name == StringConstants.ErrorRuleName)
                return true;
        return false;
    }

}
```

Поиск точки привязки

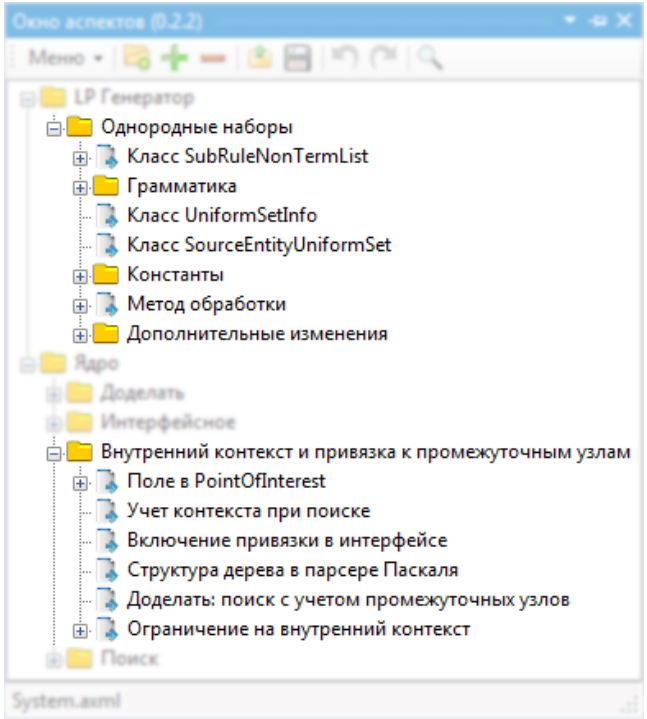
Не по номеру строки,
а по набору данных:

- Имя
- Тип
- Контекст
  - Родительские узлы
  - Дочерние узлы
  - Соседние узлы
- Текстовая строка



Примеры использования

Переключение между задачами



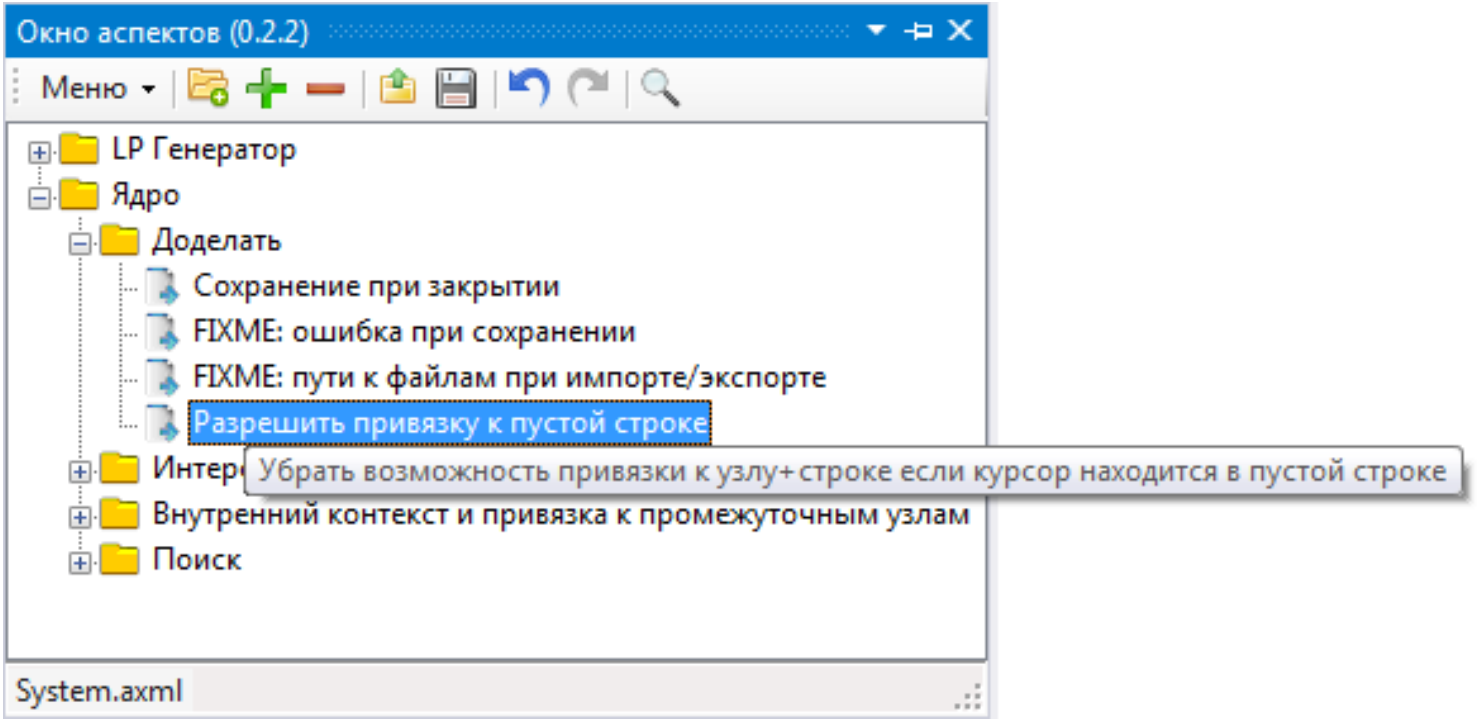
Задача 1

Рабочее множество 1

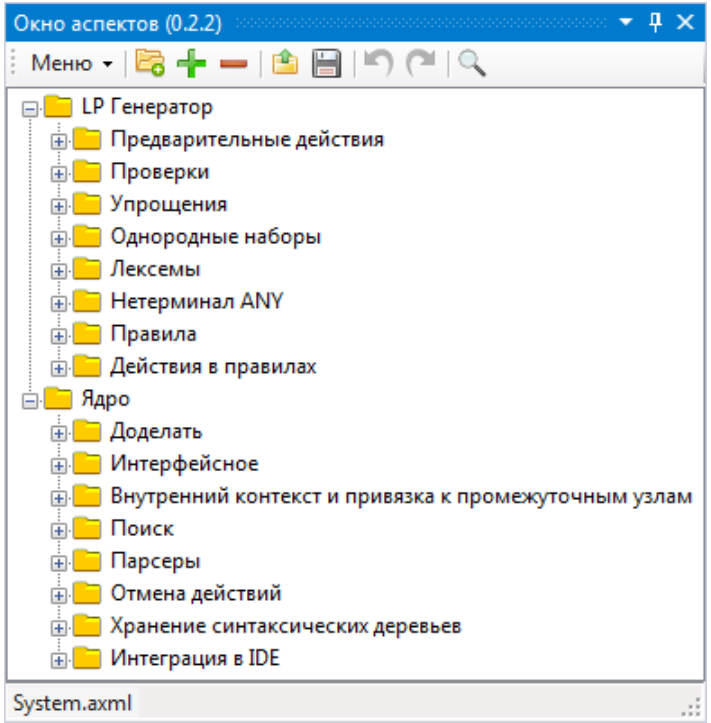
Задача 2

Рабочее множество 2

Классические закладки



Знакомство с проектом

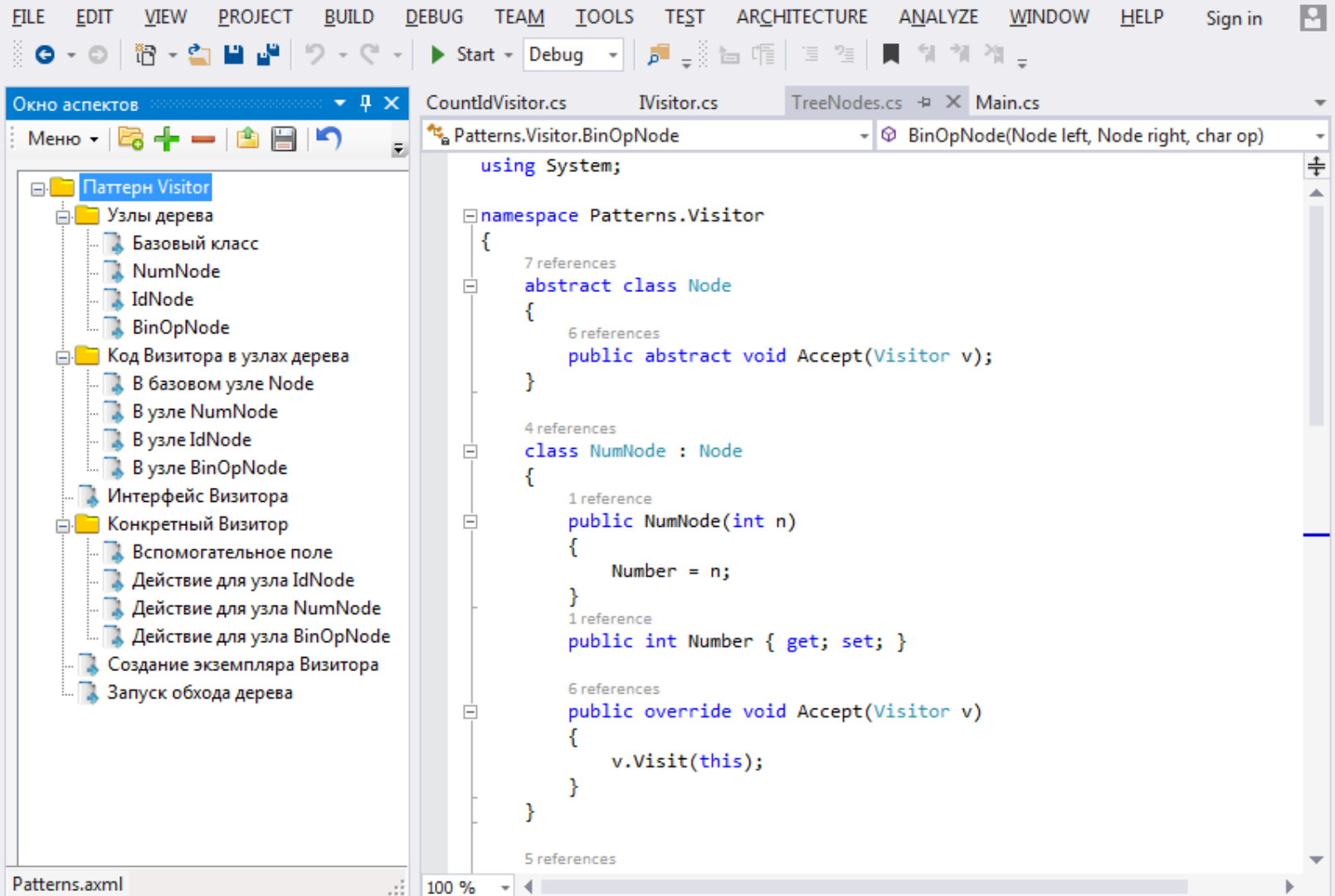


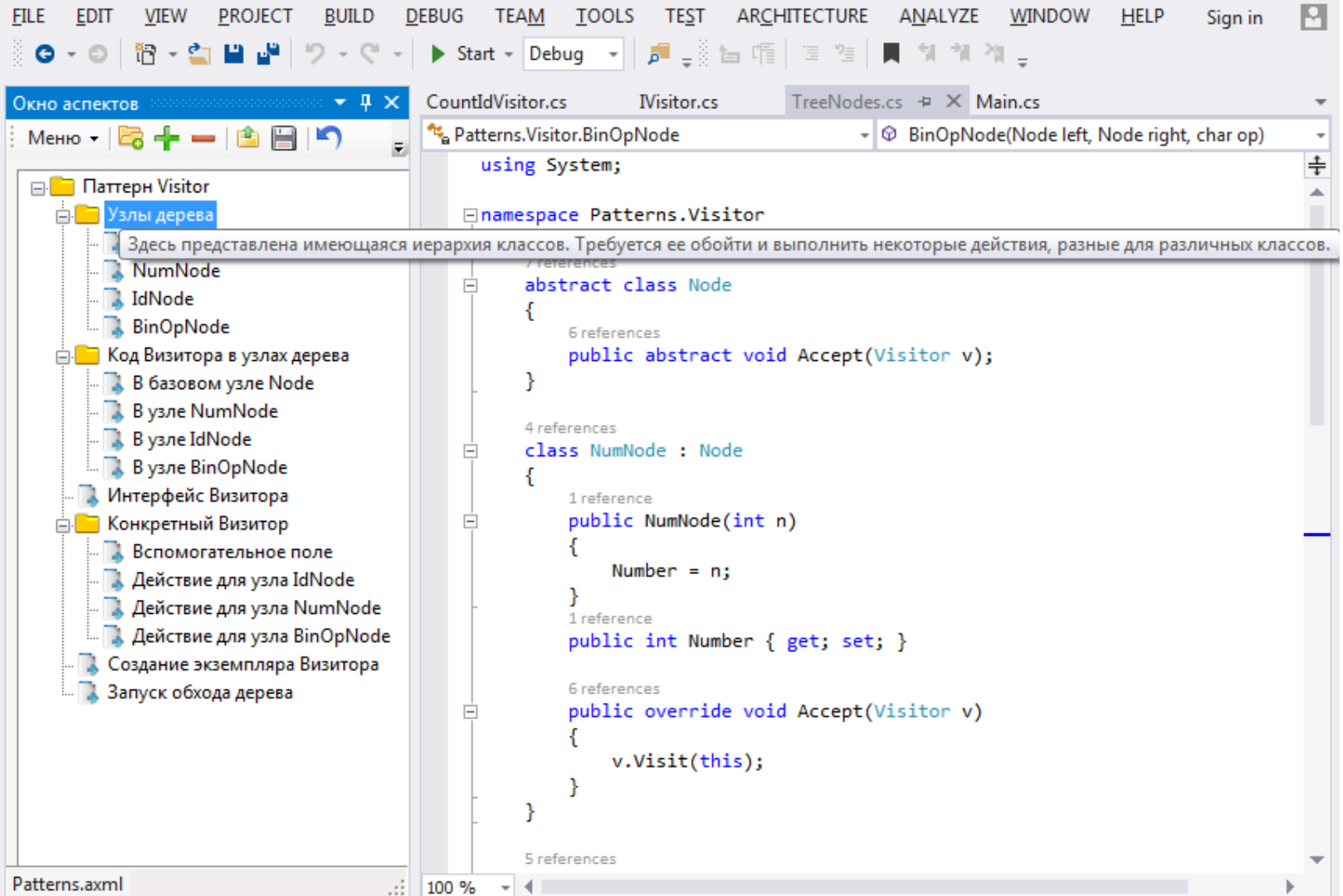
≈

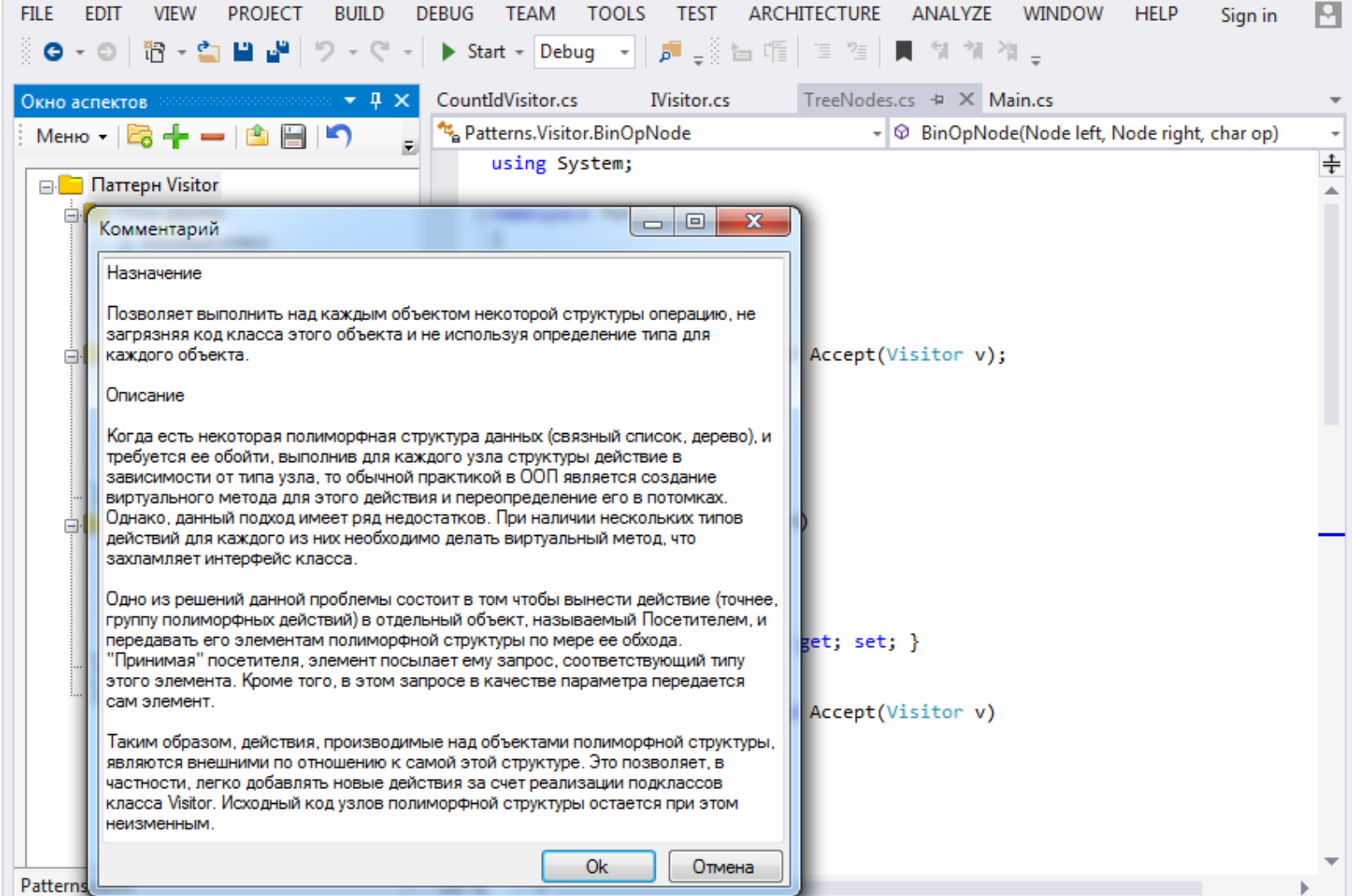
Содержание

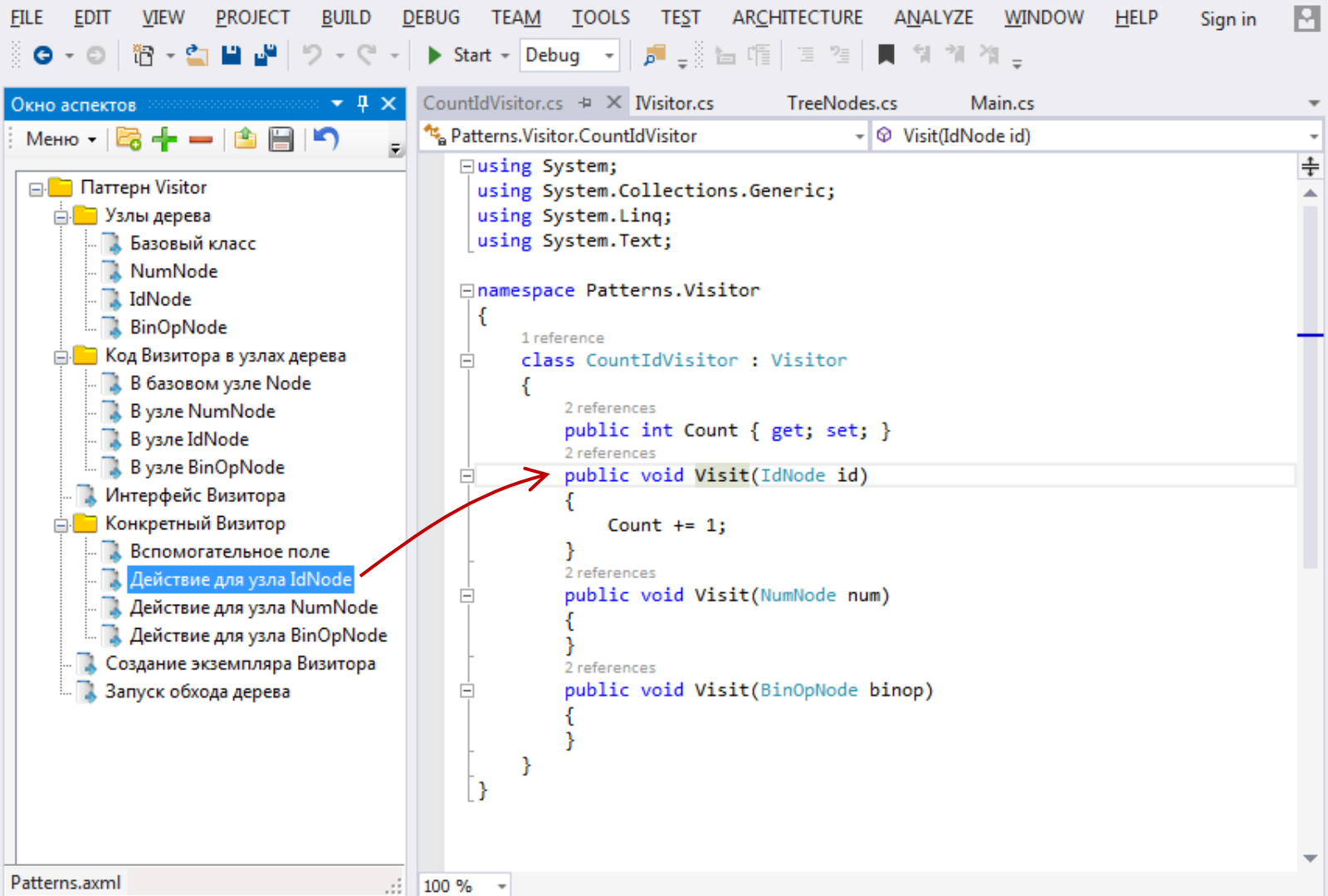
Применение аспектной разметки в обучении

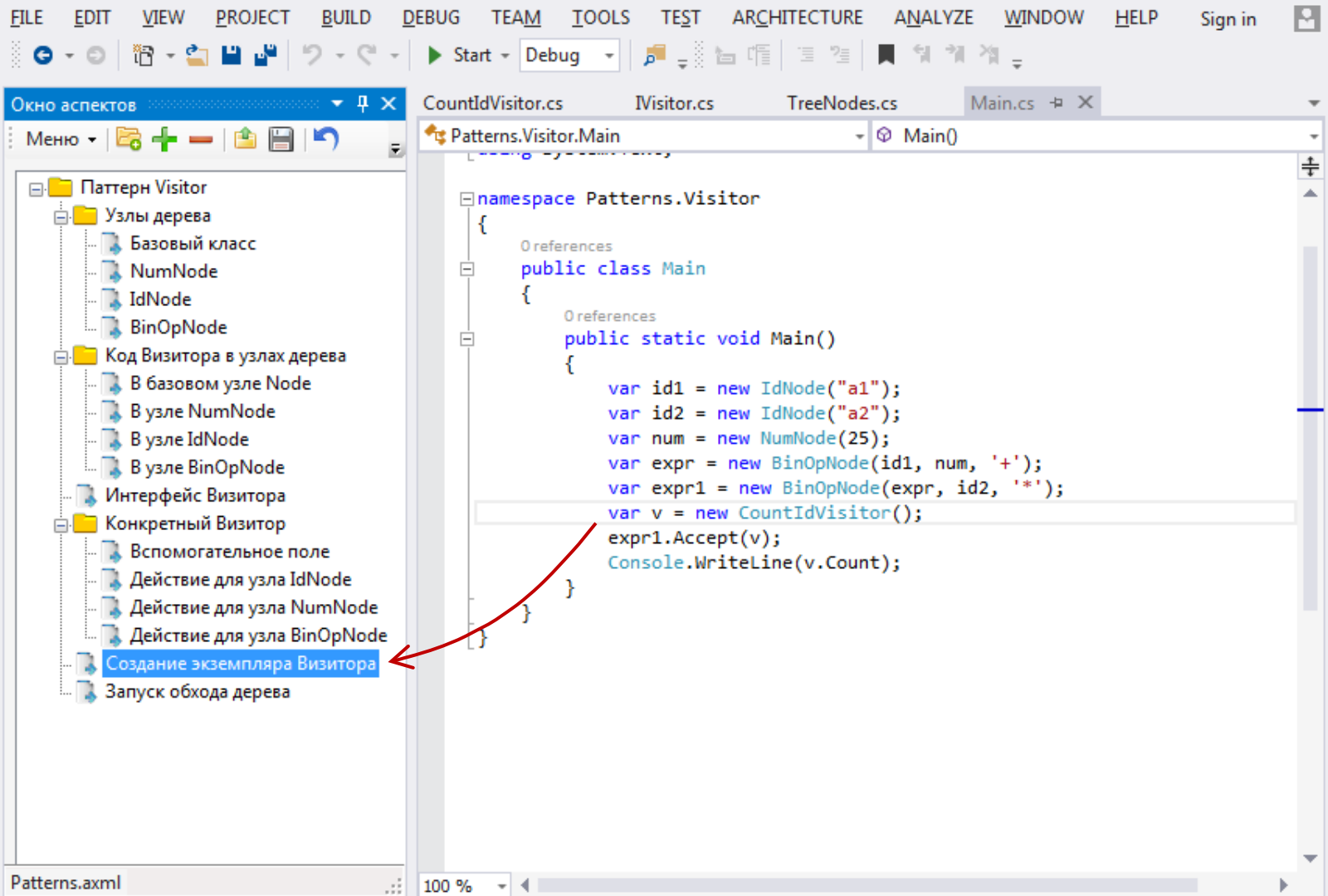
Паттерн проектирования *Visitor*

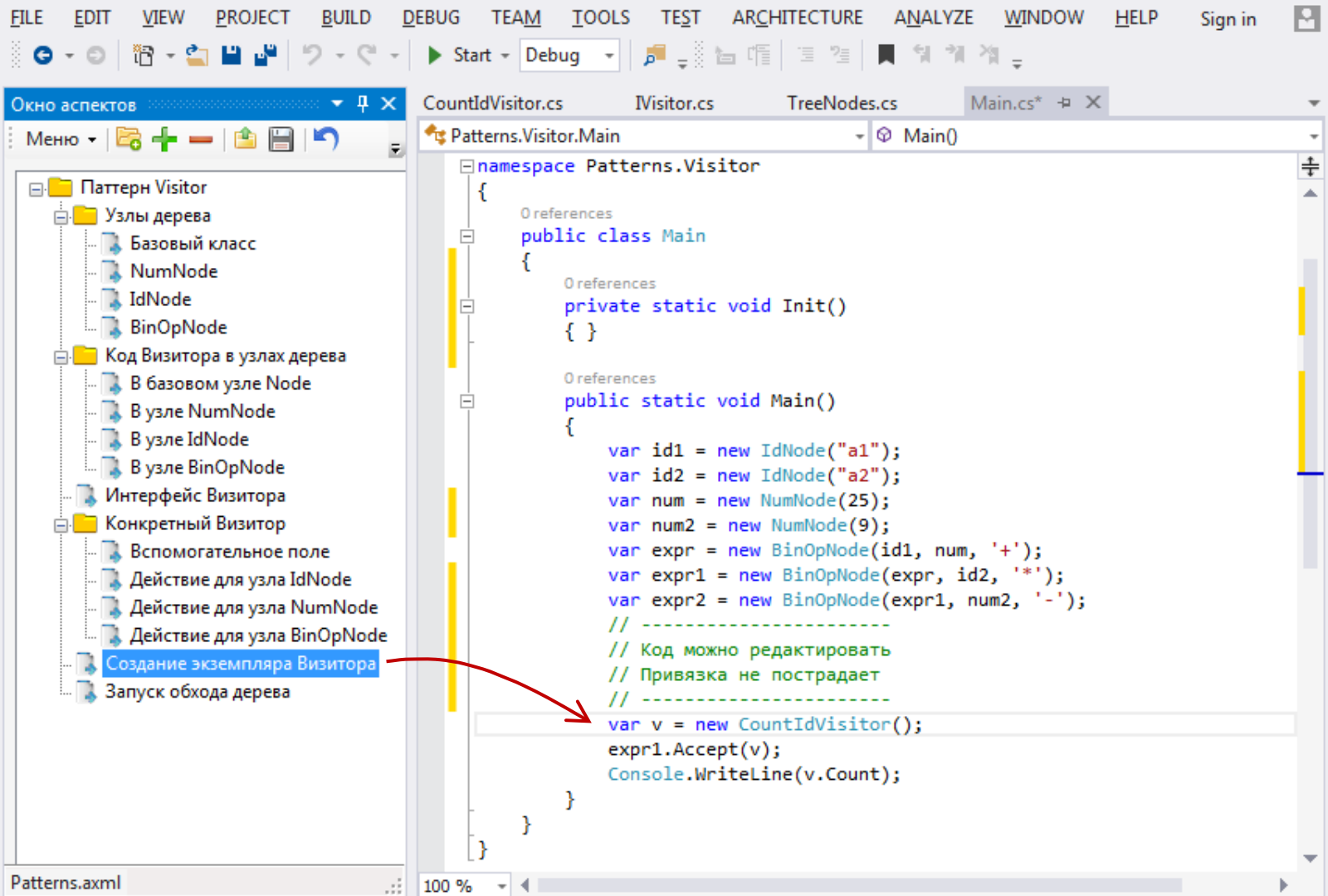

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Sign in
Start Debug
Окно аспектов
Меню
Паттерн Visitor
Узлы дерева
Базовый класс
NumNode
IdNode
BinOpNode
Код Визитора в узлах дерева
В базовом узле Node
В узле NumNode
В узле IdNode
В узле BinOpNode
Интерфейс Визитора
Конкретный Визитор
Вспомогательное поле
Действие для узла IdNode
Действие для узла NumNode
Действие для узла BinOpNode
Создание экземпляра Визитора
Запуск обхода дерева
Patterns.axml
CountIdVisitor.cs
IVisitor.cs
TreeNodes.cs
Main.cs
Patterns.Visitor.BinOpNode
BinOpNode(Node left, Node right, char op)
using System;
namespace Patterns.Visitor
{
7 references
abstract class Node
{
6 references
public abstract void Accept(Visitor v);
}
4 references
class NumNode : Node
{
1 reference
public NumNode(int n)
{
Number = n;
}
1 reference
public int Number { get; set; }
6 references
public override void Accept(Visitor v)
{
v.Visit(this);
}
}
5 references
100 %


FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Sign in
Start Debug
Окно аспектов
Меню
Паттерн Visitor
Узлы дерева
NumNode
IdNode
BinOpNode
Код Визитора в узлах дерева
В базовом узле Node
В узле NumNode
В узле IdNode
В узле BinOpNode
Интерфейс Визитора
Конкретный Визитор
Вспомогательное поле
Действие для узла IdNode
Действие для узла NumNode
Действие для узла BinOpNode
Создание экземпляра Визитора
Запуск обхода дерева
Patterns.axml
CountIdVisitor.cs
IVisitor.cs
TreeNodes.cs
Main.cs
Patterns.Visitor.BinOpNode
BinOpNode(Node left, Node right, char op)
using System;
namespace Patterns.Visitor
Здесь представлена имеющаяся иерархия классов. Требуется ее обойти и выполнить некоторые действия, разные для различных классов.
7 references
abstract class Node
{
6 references
public abstract void Accept(Visitor v);
}
4 references
class NumNode : Node
{
1 reference
public NumNode(int n)
{
Number = n;
}
1 reference
public int Number { get; set; }
6 references
public override void Accept(Visitor v)
{
v.Visit(this);
}
}
5 references
100 %


FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Sign in
Start Debug
Окно аспектов
Меню
Паттерн Visitor
CountIdVisitor.cs IVisitor.cs TreeNodes.cs Main.cs
Patterns.Visitor.BinOpNode
BinOpNode(Node left, Node right, char op)
using System;
Accept(Visitor v);
get; set; }
Accept(Visitor v)
Комментарий
Назначение
Позволяет выполнить над каждым объектом некоторой структуры операцию, не загрязняя код класса этого объекта и не используя определение типа для каждого объекта.
Описание
Когда есть некоторая полиморфная структура данных (связный список, дерево), и требуется ее обойти, выполнив для каждого узла структуры действие в зависимости от типа узла, то обычной практикой в ООП является создание виртуального метода для этого действия и переопределение его в потомках. Однако, данный подход имеет ряд недостатков. При наличии нескольких типов действий для каждого из них необходимо делать виртуальный метод, что захламляет интерфейс класса.
Одно из решений данной проблемы состоит в том чтобы вынести действие (точнее, группу полиморфных действий) в отдельный объект, называемый Посетителем, и передавать его элементам полиморфной структуры по мере ее обхода. "Принимая" посетителя, элемент посылает ему запрос, соответствующий типу этого элемента. Кроме того, в этом запросе в качестве параметра передается сам элемент.
Таким образом, действия, производимые над объектами полиморфной структуры, являются внешними по отношению к самой этой структуре. Это позволяет, в частности, легко добавлять новые действия за счет реализации подклассов класса Visitor. Исходный код узлов полиморфной структуры остается при этом неизменным.
Ok
Отмена

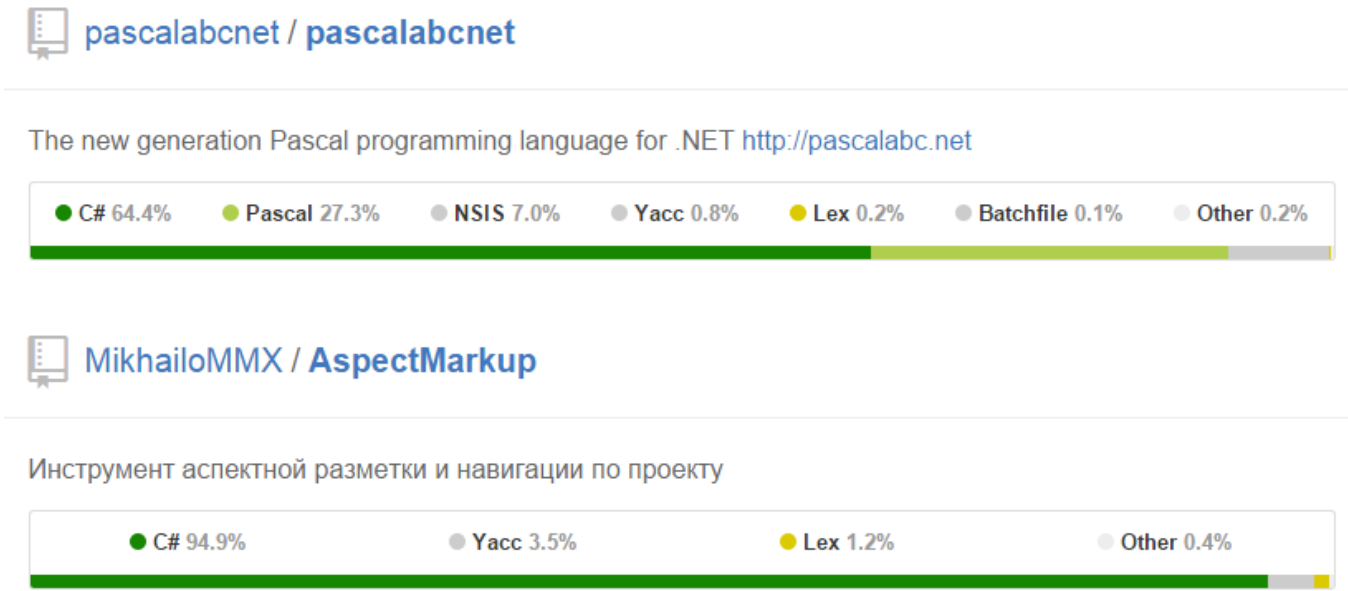

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Sign in
Start Debug
Окно аспектов
Меню
Паттерн Visitor
Узлы дерева
Базовый класс
NumNode
IdNode
BinOpNode
Код Визитора в узлах дерева
В базовом узле Node
В узле NumNode
В узле IdNode
В узле BinOpNode
Интерфейс Визитора
Конкретный Визитор
Вспомогательное поле
Действие для узла IdNode
Действие для узла NumNode
Действие для узла BinOpNode
Создание экземпляра Визитора
Запуск обхода дерева
Patterns.axml
CountIdVisitor.cs
IVisitor.cs
TreeNodes.cs
Main.cs
Patterns.Visitor.CountIdVisitor
Visit(IdNode id)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Patterns.Visitor
{
1 reference
class CountIdVisitor : Visitor
{
2 references
public int Count { get; set; }
2 references
public void Visit(IdNode id)
{
Count += 1;
}
2 references
public void Visit(NumNode num)
{
}
2 references
public void Visit(BinOpNode binop)
{
}
}
}
100 %


FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Sign in
Start Debug
Окно аспектов
Меню
Паттерн Visitor
Узлы дерева
Базовый класс
NumNode
IdNode
BinOpNode
Код Визитора в узлах дерева
В базовом узле Node
В узле NumNode
В узле IdNode
В узле BinOpNode
Интерфейс Визитора
Конкретный Визитор
Вспомогательное поле
Действие для узла IdNode
Действие для узла NumNode
Действие для узла BinOpNode
Создание экземпляра Визитора
Запуск обхода дерева
Patterns.axml
CountIdVisitor.cs
IVisitor.cs
TreeNodes.cs
Main.cs
Patterns.Visitor.Main
Main()
namespace Patterns.Visitor
{
0 references
public class Main
{
0 references
public static void Main()
{
var id1 = new IdNode("a1");
var id2 = new IdNode("a2");
var num = new NumNode(25);
var expr = new BinOpNode(id1, num, '+');
var expr1 = new BinOpNode(expr, id2, '*');
var v = new CountIdVisitor();
expr1.Accept(v);
Console.WriteLine(v.Count);
}
}
}
100 %


FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Sign in
Start Debug
Окно аспектов
Меню
Паттерн Visitor
Узлы дерева
Базовый класс
NumNode
IdNode
BinOpNode
Код Визитора в узлах дерева
В базовом узле Node
В узле NumNode
В узле IdNode
В узле BinOpNode
Интерфейс Визитора
Конкретный Визитор
Вспомогательное поле
Действие для узла IdNode
Действие для узла NumNode
Действие для узла BinOpNode
Создание экземпляра Визитора
Запуск обхода дерева
Patterns.axml
CountIdVisitor.cs
IVisitor.cs
TreeNodes.cs
Main.cs*
Patterns.Visitor.Main
Main()
namespace Patterns.Visitor
{
0 references
public class Main
{
0 references
private static void Init()
{ }
0 references
public static void Main()
{
var id1 = new IdNode("a1");
var id2 = new IdNode("a2");
var num = new NumNode(25);
var num2 = new NumNode(9);
var expr = new BinOpNode(id1, num, '+');
var expr1 = new BinOpNode(expr, id2, '*');
var expr2 = new BinOpNode(expr1, num2, '-');
// ----------------------
// Код можно редактировать
// Привязка не пострадает
// ----------------------
var v = new CountIdVisitor();
expr1.Accept(v);
Console.WriteLine(v.Count);
}
}
}
100 %

Крупные проекты

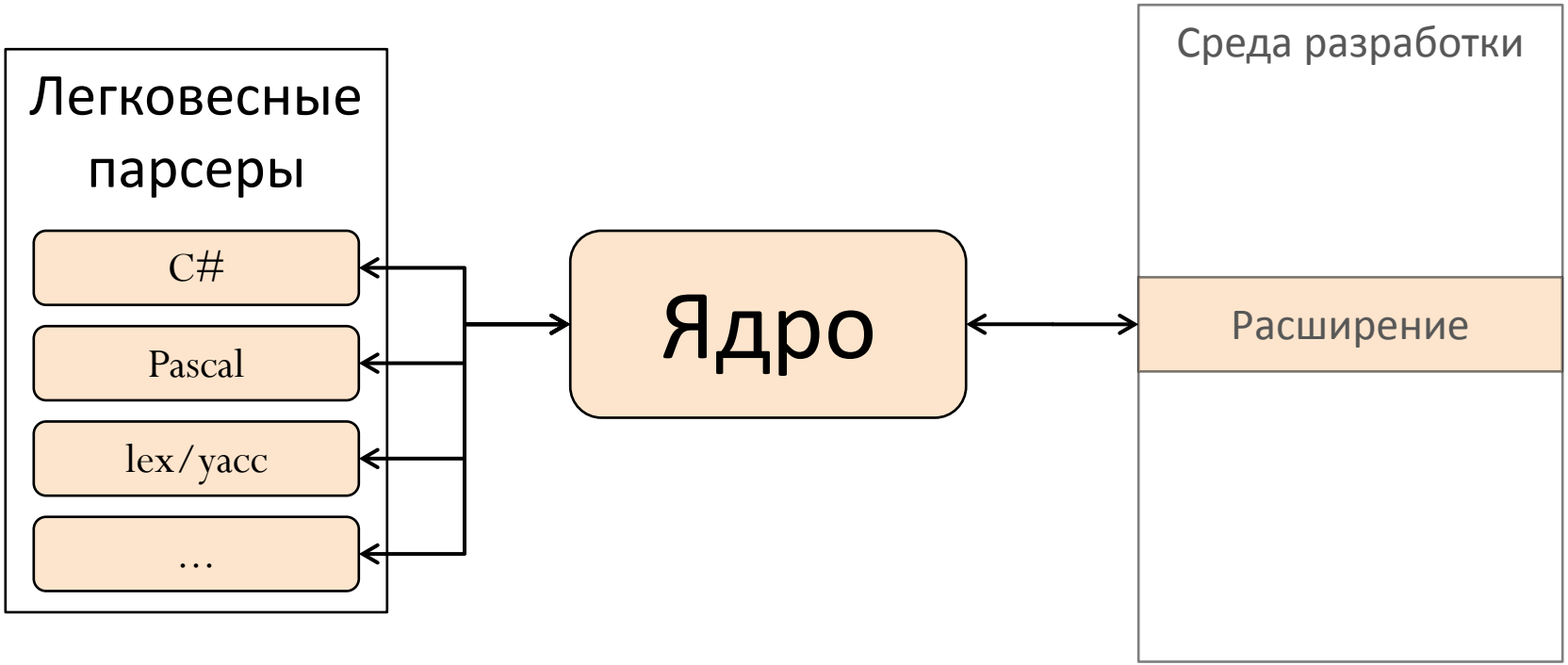
часто пишутся на нескольких языках:



Нельзя полагаться на инфраструктуру среды разработки

- Не поддерживает все языки, участвующие в разработке:
  - *C#*, *lex*, *yacc*, *Pascal*, *DSL-языки*
- Может измениться с выходом новой версии

Структура инструмента



Легковесные парсеры

Главные особенности:

- Максимально упрощенная грамматика
- Устойчивость к ошибкам в исходном коде
- Устойчивость к изменению спецификации языка
- Простота разработки на DSL-языке LightParse

Легковесный парсер C# на языке LightParse

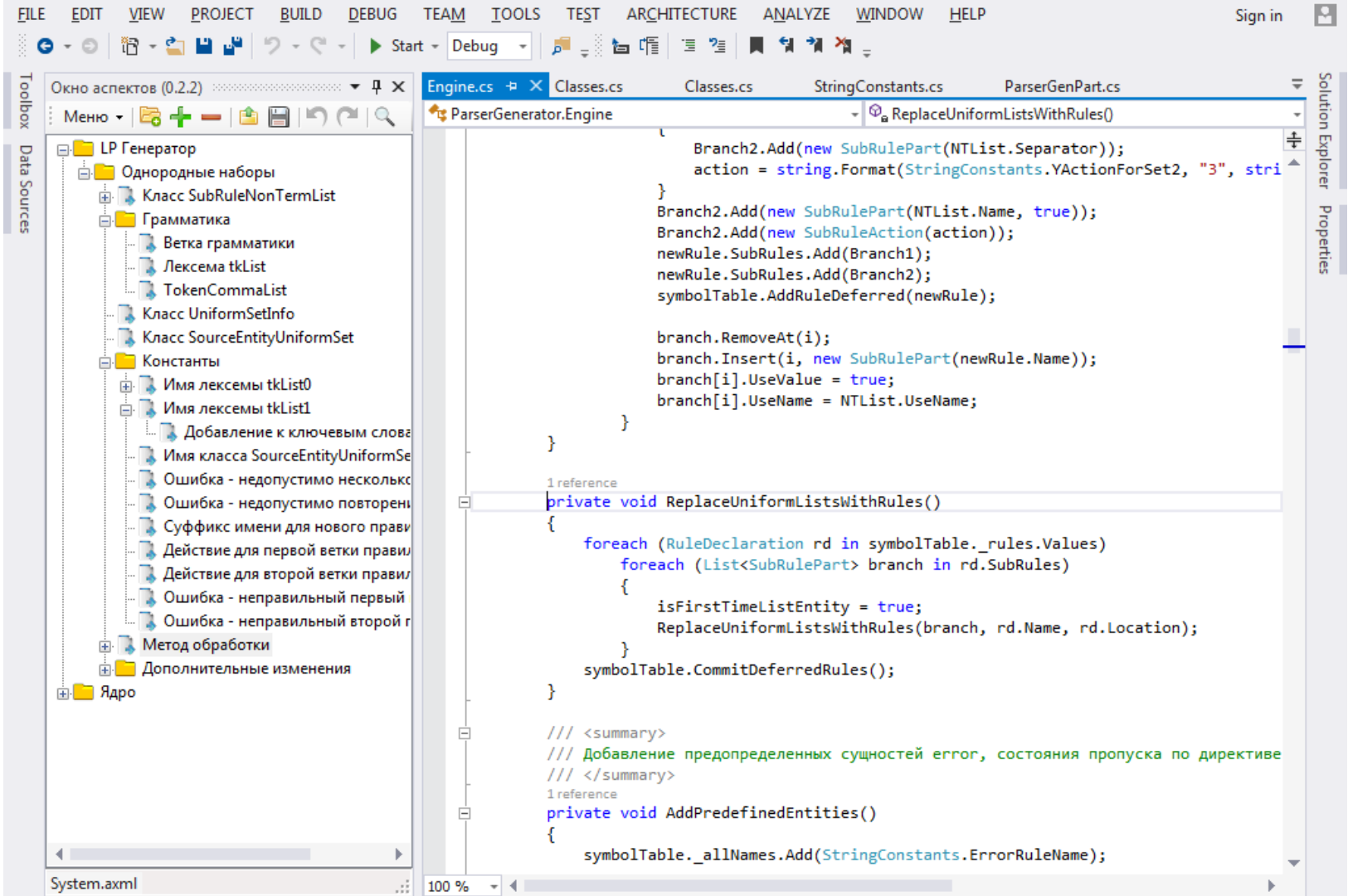
```
%Extension "cs"
%CaseSensitive
%Preprocessor define="define" undef="undef" ifdef="if" elif="elif" else="else" end="endif"

Skip Begin "//"
Skip Begin "#" Preprocessor
Skip Begin "/*" End "*/"
Skip BeginEnd "'" EscapeSymbol "'"
Skip BeginEnd '\"' EscapeSymbol '\\'
Skip Begin '@\"' End '\"'

Token LetterDigits [[:IsLetterOrDigit:]_&]*
Token Sign [[:IsPunctuation:][:IsSymbol:]]

Rule tkClassNamespace : @"class" | @"namespace"
Rule @Block : "{" [Any | Block]* "}"
Rule Tk : @AnyExcept(tkClassNamespace, "{", "}", ";")

Rule Program :          #ProgramNode*
Rule ProgramNode:       #ClassOrNamespace
                        | #Field
                        | #Method
                        | #error

Rule ClassOrNamespace : @Tk* @tkClassNamespace @Tk* "{" #ProgramNode* "}"
Rule Field :            @Tk* ";"
Rule Method :           @Tk* Block
```

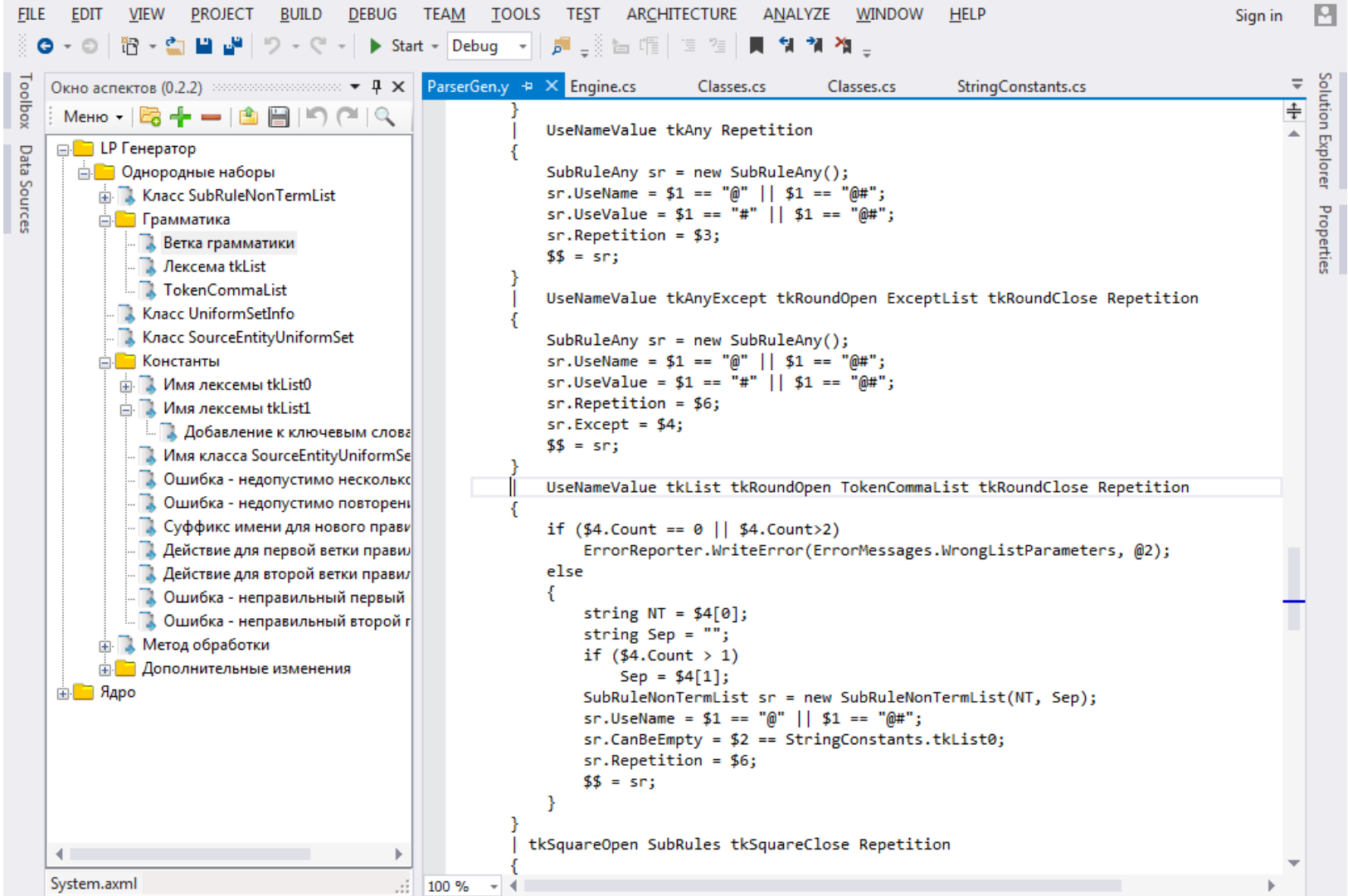
Легковесный парсер LightParse на языке LightParse

```
1   %Extension "lp"
2   %CaseSensitive
3
4   Skip Begin "//"
5
6   Token LetterDigit [[:IsLetterOrDigit:]_]*
7   Token String \"([^\"\\]|\\.)*\"|\'([^\'\\]|\\.)*\'
8   Token Sign [[:IsPunctuation:][:IsSymbol:]]|\\\"|\\\'
9
10  Rule Program: #ProgramNode*
11  Rule ProgramNode:   #Directive
12                      | #dSkip
13                      | #dToken
14                      | #dRule
15                      | error
16
17  Rule Tk: @AnyExcept("Skip", "Rule", "Token", "%")
18
19  Rule @Directive : "%" @Tk*
20  Rule dSkip: @"Skip" @Tk*
21  Rule dToken: "Token" @LetterDigit Tk*
22  Rule dRule: "Rule" Sign? @LetterDigit ":" Tk*
```

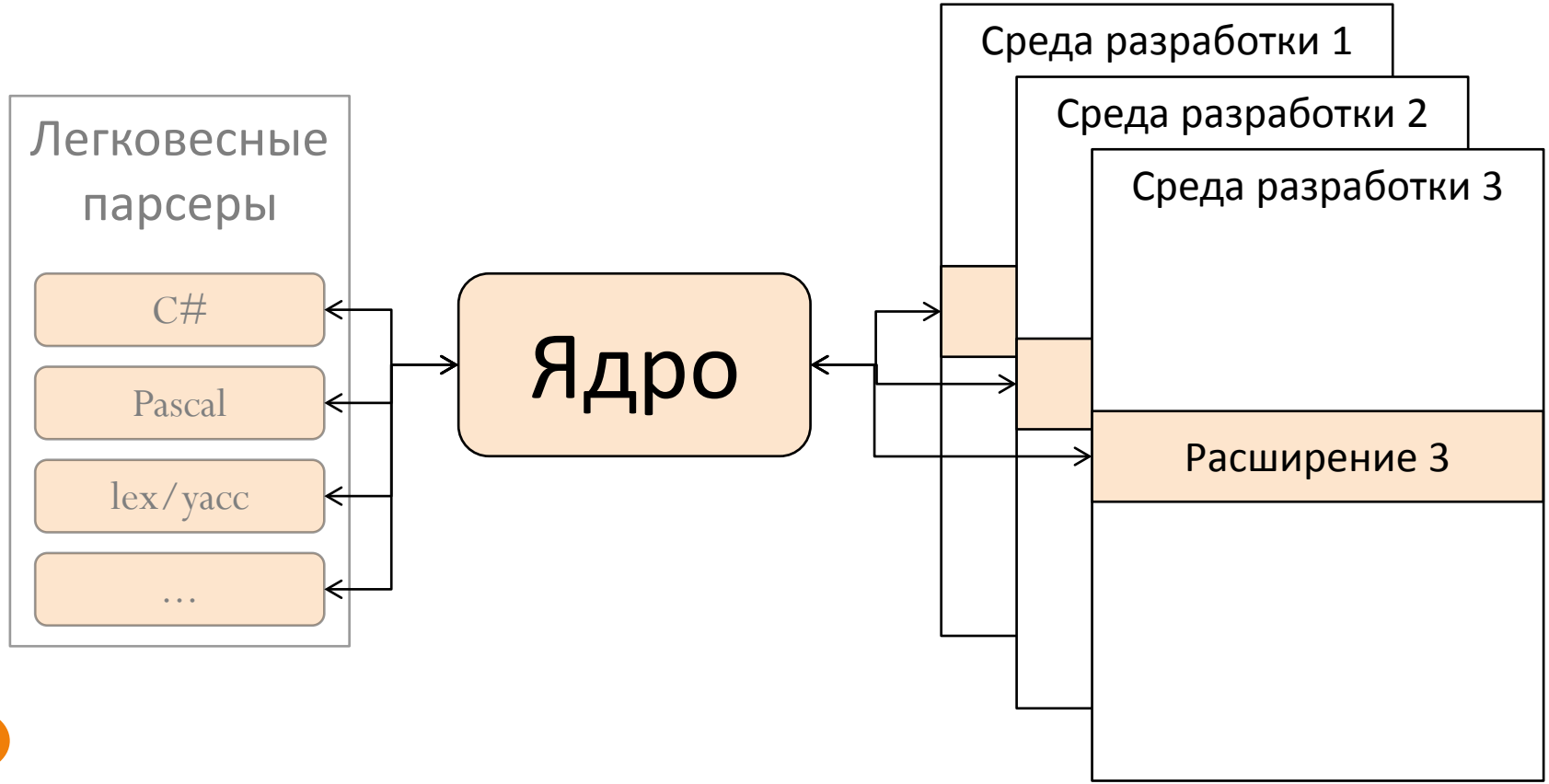
Применение аспектной разметки

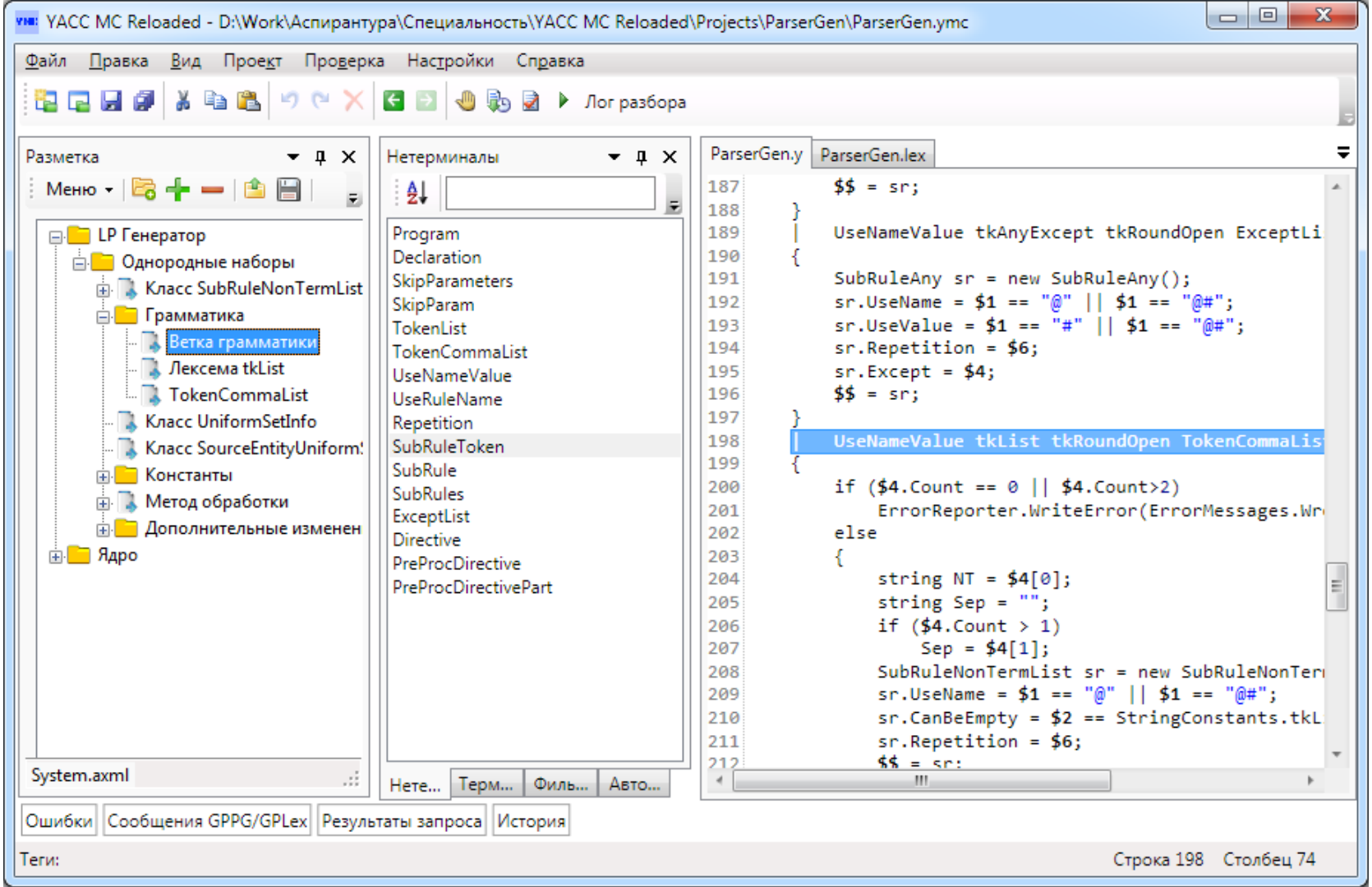
при разработке инструмента аспектной разметки

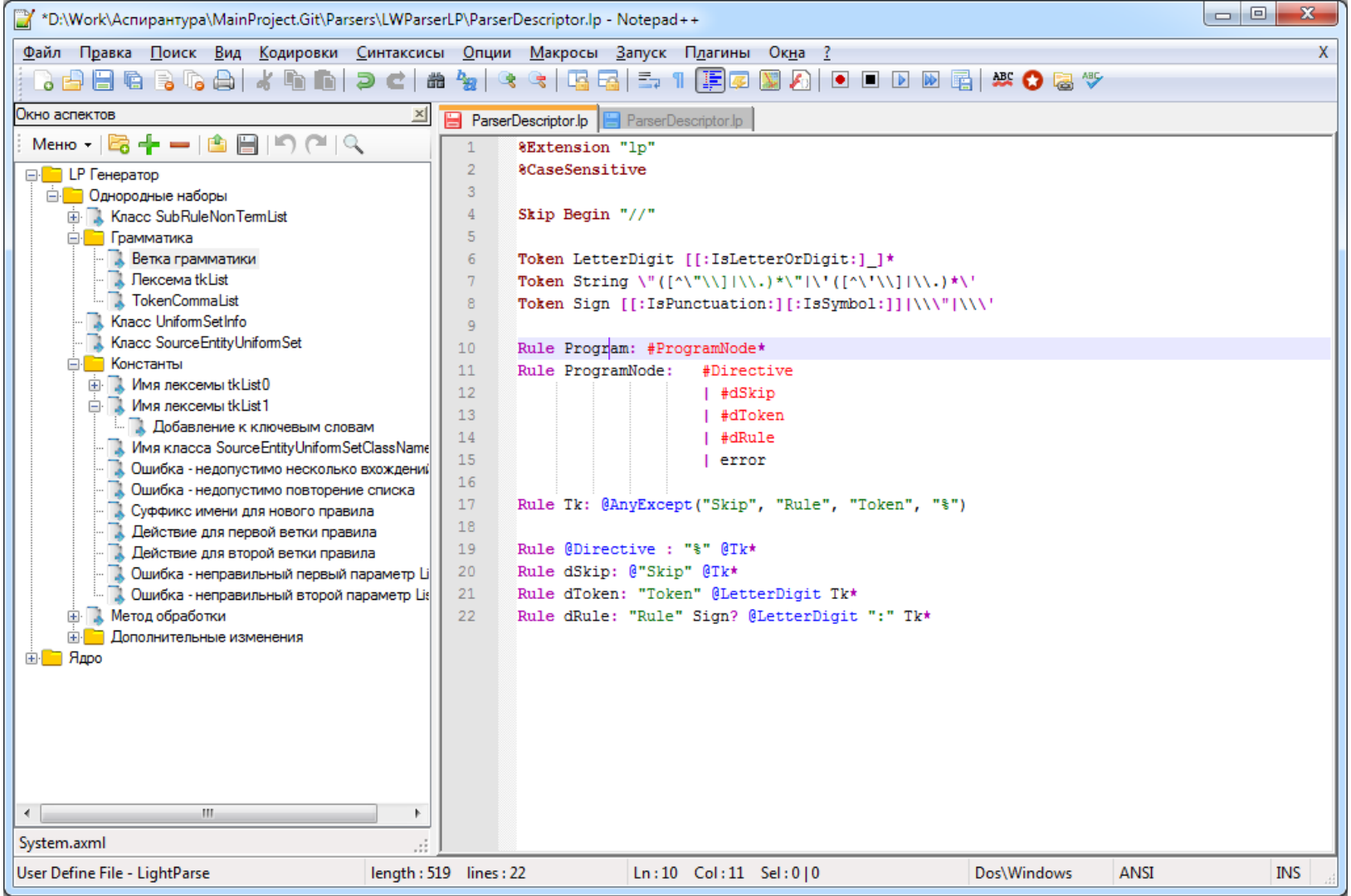

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Sign in
Start
Debug
Toolbox
Data Sources
Окно аспектов (0.2.2)
Меню
LP Генератор
Однородные наборы
Класс SubRuleNonTermList
Грамматика
Ветка грамматики
Лексема tkList
TokenCommaList
Класс UniformSetInfo
Класс SourceEntityUniformSet
Константы
Имя лексемы tkList0
Имя лексемы tkList1
Добавление к ключевым слова
Имя класса SourceEntityUniformSe
Ошибка - недопустимо нескольк
Ошибка - недопустимо повторени
Суффикс имени для нового прави
Действие для первой ветки правил
Действие для второй ветки правил
Ошибка - неправильный первый
Ошибка - неправильный второй г
Метод обработки
Дополнительные изменения
Ядро
System.axml
Engine.cs
Classes.cs
Classes.cs
StringConstants.cs
ParserGenPart.cs
ParserGenerator.Engine
ReplaceUniformListsWithRules()
Branch2.Add(new SubRulePart(NTList.Separator));
action = string.Format(StringConstants.YActionForSet2, "3", stri
}
Branch2.Add(new SubRulePart(NTList.Name, true));
Branch2.Add(new SubRuleAction(action));
newRule.SubRules.Add(Branch1);
newRule.SubRules.Add(Branch2);
symbolTable.AddRuleDeferred(newRule);
branch.RemoveAt(i);
branch.Insert(i, new SubRulePart(newRule.Name));
branch[i].UseValue = true;
branch[i].UseName = NTList.UseName;
}
}
1 reference
private void ReplaceUniformListsWithRules()
{
foreach (RuleDeclaration rd in symbolTable._rules.Values)
foreach (List<SubRulePart> branch in rd.SubRules)
{
isFirstTimeListEntity = true;
ReplaceUniformListsWithRules(branch, rd.Name, rd.Location);
}
symbolTable.CommitDeferredRules();
}
/// <summary>
/// Добавление предопределенных сущностей error, состояния пропуска по директиве
/// </summary>
1 reference
private void AddPredefinedEntities()
{
symbolTable._allNames.Add(StringConstants.ErrorRuleName);
100 %
Solution Explorer
Properties

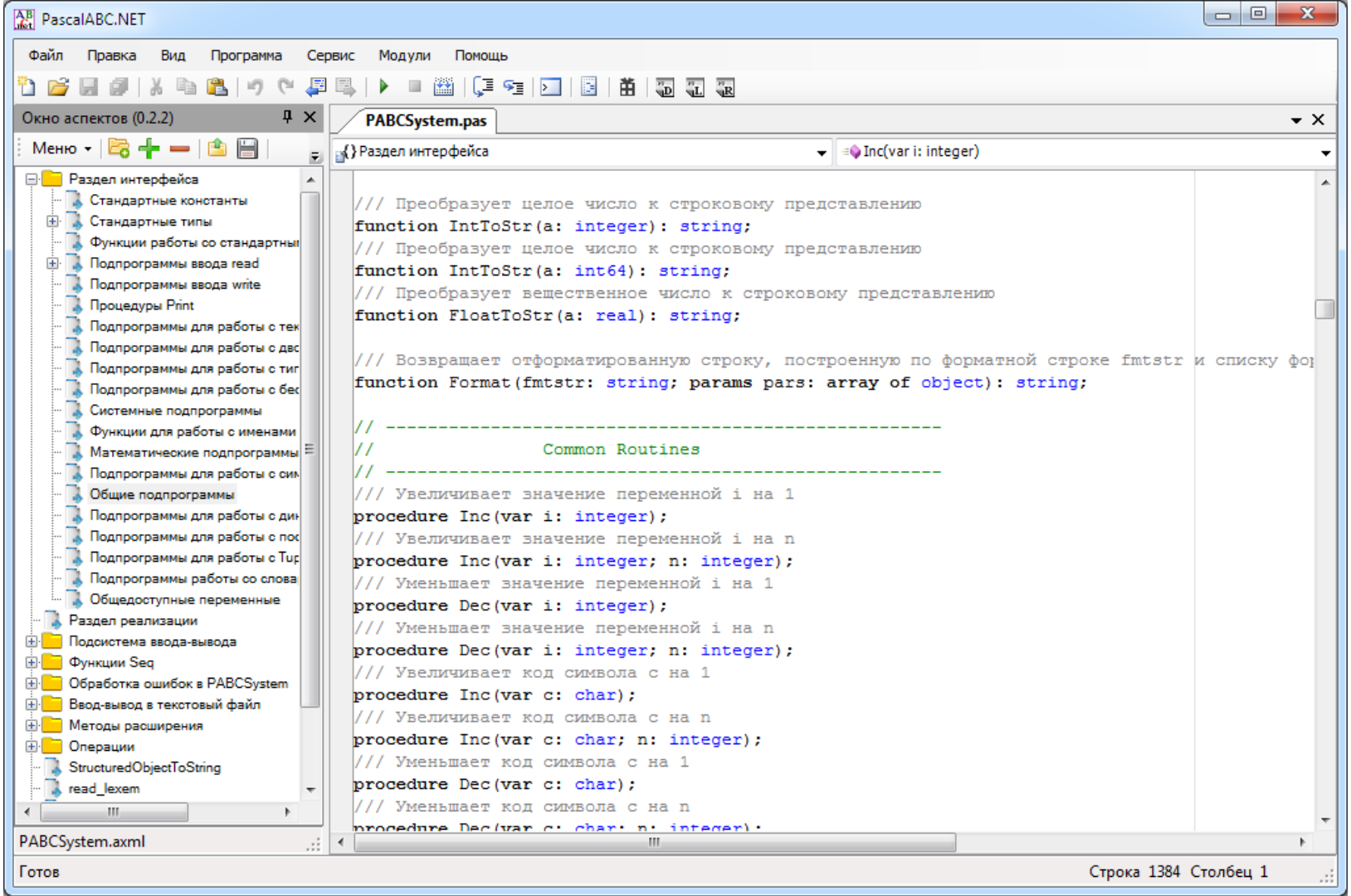

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Sign in
Start
Debug
Toolbox
Data Sources
Окно аспектов (0.2.2)
Меню
LP Генератор
Однородные наборы
Класс SubRuleNonTermList
Грамматика
Ветка грамматики
Лексема tkList
TokenCommaList
Класс UniformSetInfo
Класс SourceEntityUniformSet
Константы
Имя лексемы tkList0
Имя лексемы tkList1
Добавление к ключевым слова
Имя класса SourceEntityUniformSe
Ошибка - недопустимо нескольк
Ошибка - недопустимо повторени
Суффикс имени для нового прави
Действие для первой ветки правил
Действие для второй ветки правил
Ошибка - неправильный первый
Ошибка - неправильный второй г
Метод обработки
Дополнительные изменения
Ядро
System.axml
ParserGen.y
Engine.cs
Classes.cs
Classes.cs
StringConstants.cs
}
| UseNameValue tkAny Repetition
{
SubRuleAny sr = new SubRuleAny();
sr.UseName = $1 == "@" || $1 == "@#";
sr.UseValue = $1 == "#" || $1 == "@#";
sr.Repetition = $3;
$$ = sr;
}
| UseNameValue tkAnyExcept tkRoundOpen ExceptList tkRoundClose Repetition
{
SubRuleAny sr = new SubRuleAny();
sr.UseName = $1 == "@" || $1 == "@#";
sr.UseValue = $1 == "#" || $1 == "@#";
sr.Repetition = $6;
sr.Except = $4;
$$ = sr;
}
| UseNameValue tkList tkRoundOpen TokenCommaList tkRoundClose Repetition
{
if ($4.Count == 0 || $4.Count>2)
ErrorReporter.WriteError(ErrorMessages.WrongListParameters, @2);
else
{
string NT = $4[0];
string Sep = "";
if ($4.Count > 1)
Sep = $4[1];
SubRuleNonTermList sr = new SubRuleNonTermList(NT, Sep);
sr.UseName = $1 == "@" || $1 == "@#";
sr.CanBeEmpty = $2 == StringConstants.tkList0;
sr.Repetition = $6;
$$ = sr;
}
}
| tkSquareOpen SubRules tkSquareClose Repetition
{
100 %
Solution Explorer
Properties

Структура инструмента




YACC MC Reloaded - D:\Work\Аспирантура\Специальность\YACC MC Reloaded\Projects\ParserGen\ParserGen.ymc
Файл Правка Вид Проект Проверка Настройки Справка
Лог разбора
Разметка
Меню
LP Генератор
Однородные наборы
Класс SubRuleNonTermList
Грамматика
Ветка грамматики
Лексема tkList
TokenCommaList
Класс UniformSetInfo
Класс SourceEntityUniform
Константы
Метод обработки
Дополнительные изменен
Ядро
System.axml
Нетерминалы
Program
Declaration
SkipParameters
SkipParam
TokenList
TokenCommaList
UseNameValue
UseRuleName
Repetition
SubRuleToken
SubRule
SubRules
ExceptList
Directive
PreProcDirective
PreProcDirectivePart
Нете...
Терм...
Филь...
Авто...
ParserGen.y
ParserGen.lex
$$ = sr;
}
| UseNameValue tkAnyExcept tkRoundOpen ExceptLi
{
SubRuleAny sr = new SubRuleAny();
sr.UseName = $1 == "@" || $1 == "@#";
sr.UseValue = $1 == "#" || $1 == "@#";
sr.Repetition = $6;
sr.Except = $4;
$$ = sr;
}
| UseNameValue tkList tkRoundOpen TokenCommaLis
{
if ($4.Count == 0 || $4.Count>2)
ErrorReporter.WriteError(ErrorMessages.Wr
else
{
string NT = $4[0];
string Sep = "";
if ($4.Count > 1)
Sep = $4[1];
SubRuleNonTermList sr = new SubRuleNonTer
sr.UseName = $1 == "@" || $1 == "@#";
sr.CanBeEmpty = $2 == StringConstants.tkL
sr.Repetition = $6;
$$ = sr;
Ошибки
Сообщения GPPG/GPLex
Результаты запроса
История
Теги:
Строка 198 Столбец 74


*D:\Work\Аспирантура\MainProject.Git\Parsers\LWParserLP\ParserDescriptor.lp - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксисы Опции Макросы Запуск Плагины Окна ?
Окно аспектов
Меню
LP Генератор
Однородные наборы
Класс SubRuleNonTermList
Грамматика
Ветка грамматики
Лексема tkList
TokenCommaList
Класс UniformSetInfo
Класс SourceEntityUniformSet
Константы
Имя лексемы tkList0
Имя лексемы tkList1
Добавление к ключевым словам
Имя класса SourceEntityUniformSetClassName
Ошибка - недопустимо несколько вхождений
Ошибка - недопустимо повторение списка
Суффикс имени для нового правила
Действие для первой ветки правила
Действие для второй ветки правила
Ошибка - неправильный первый параметр Li
Ошибка - неправильный второй параметр Lis
Метод обработки
Дополнительные изменения
Ядро
System.axml
ParserDescriptor.lp
ParserDescriptor.lp
1 %Extension "lp"
2 %CaseSensitive
3
4 Skip Begin "//"
5
6 Token LetterDigit [[:IsLetterOrDigit:]_]*
7 Token String \"([^\"\\]|\\.)*\"|\'([^\'\\]|\\.)*\'
8 Token Sign [[:IsPunctuation:][:IsSymbol:]]|\\\"|\\\'
9
10 Rule Program: #ProgramNode*
11 Rule ProgramNode: #Directive
12 | #dSkip
13 | #dToken
14 | #dRule
15 | error
16
17 Rule Tk: @AnyExcept("Skip", "Rule", "Token", "%")
18
19 Rule @Directive : "%" @Tk*
20 Rule dSkip: @"Skip" @Tk*
21 Rule dToken: "Token" @LetterDigit Tk*
22 Rule dRule: "Rule" Sign? @LetterDigit ":" Tk*
User Define File - LightParse
length : 519 lines : 22
Ln : 10 Col : 11 Sel : 0 | 0
Dos\Windows
ANSI
INS


PascalABC.NET
Файл Правка Вид Программа Сервис Модули Помощь
Окно аспектов (0.2.2)
Меню
Раздел интерфейса
Стандартные константы
Стандартные типы
Функции работы со стандартны
Подпрограммы ввода read
Подпрограммы ввода write
Процедуры Print
Подпрограммы для работы с тек
Подпрограммы для работы с дво
Подпрограммы для работы с тиг
Подпрограммы для работы с бес
Системные подпрограммы
Функции для работы с именами
Математические подпрограммы
Подпрограммы для работы с сим
Общие подпрограммы
Подпрограммы для работы с дин
Подпрограммы для работы с пос
Подпрограммы для работы с Tup
Подпрограммы работы со слова
Общедоступные переменные
Раздел реализации
Подсистема ввода-вывода
Функции Seq
Обработка ошибок в PABCSystem
Ввод-вывод в текстовый файл
Методы расширения
Операции
StructuredObjectToString
read_lexem
PABCSystem.axml
PABCSystem.pas
Раздел интерфейса
Inc(var i: integer)
/// Преобразует целое число к строковому представлению
function IntToStr(a: integer): string;
/// Преобразует целое число к строковому представлению
function IntToStr(a: int64): string;
/// Преобразует вещественное число к строковому представлению
function FloatToStr(a: real): string;
/// Возвращает отформатированную строку, построенную по форматной строке fmtstr и списку фо
function Format(fmtstr: string; params pars: array of object): string;
// ---
// Common Routines
// ---
/// Увеличивает значение переменной i на 1
procedure Inc(var i: integer);
/// Увеличивает значение переменной i на n
procedure Inc(var i: integer; n: integer);
/// Уменьшает значение переменной i на 1
procedure Dec(var i: integer);
/// Уменьшает значение переменной i на n
procedure Dec(var i: integer; n: integer);
/// Увеличивает код символа c на 1
procedure Inc(var c: char);
/// Увеличивает код символа c на n
procedure Inc(var c: char; n: integer);
/// Уменьшает код символа c на 1
procedure Dec(var c: char);
/// Уменьшает код символа c на n
Готов
Строка 1384 Столбец 1

Расширяемость

Добавление поддержки:

- Нового языка: **20-40** строк кода
- Новой среды разработки: **100-200** строк кода

Результат

Инструмент *аспектной разметки*, поддерживающий

- Языки
  - C#, lex, yacc, Java, Pacsal, XML, LightParse
- Среды разработки
  - Visual Studio, Notepad++, PascalABC.NET, YaccMC

Продолжение следует…