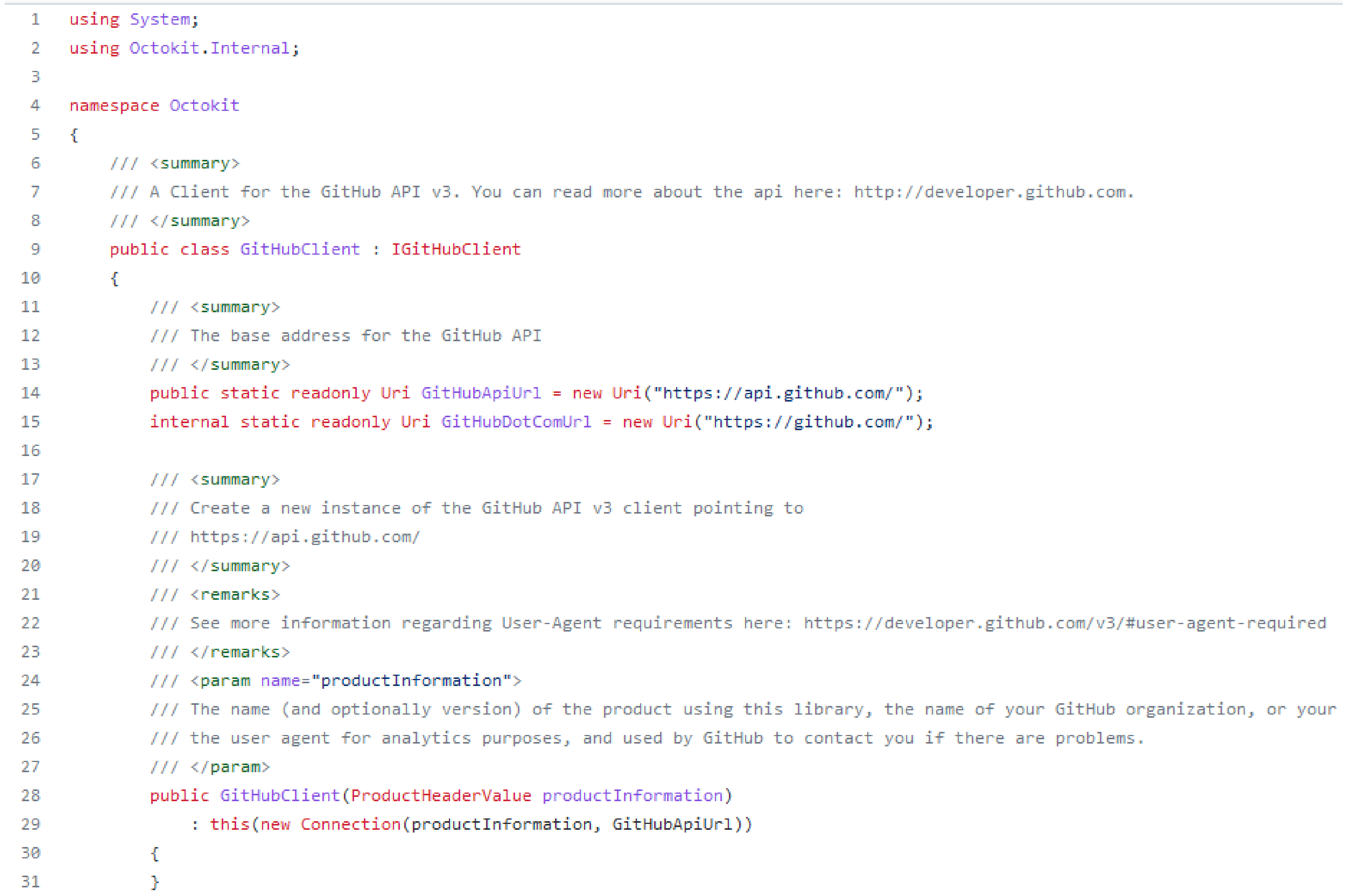
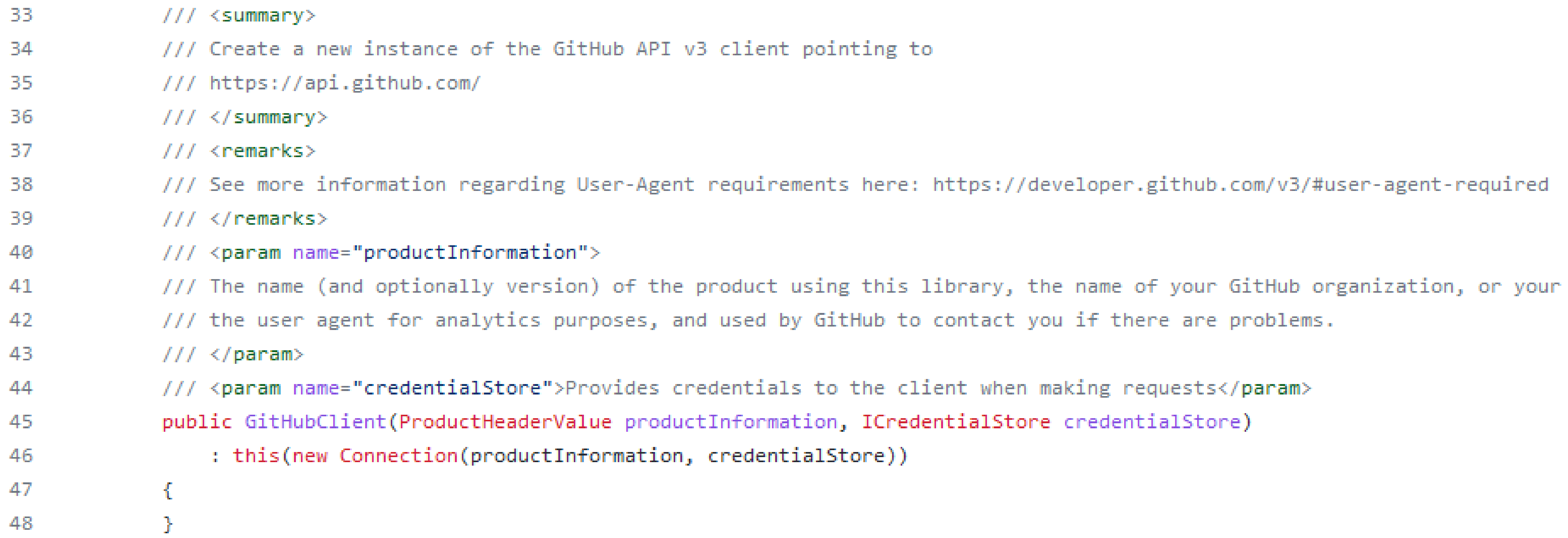
Примеры применения паттернов GoF

Octokit



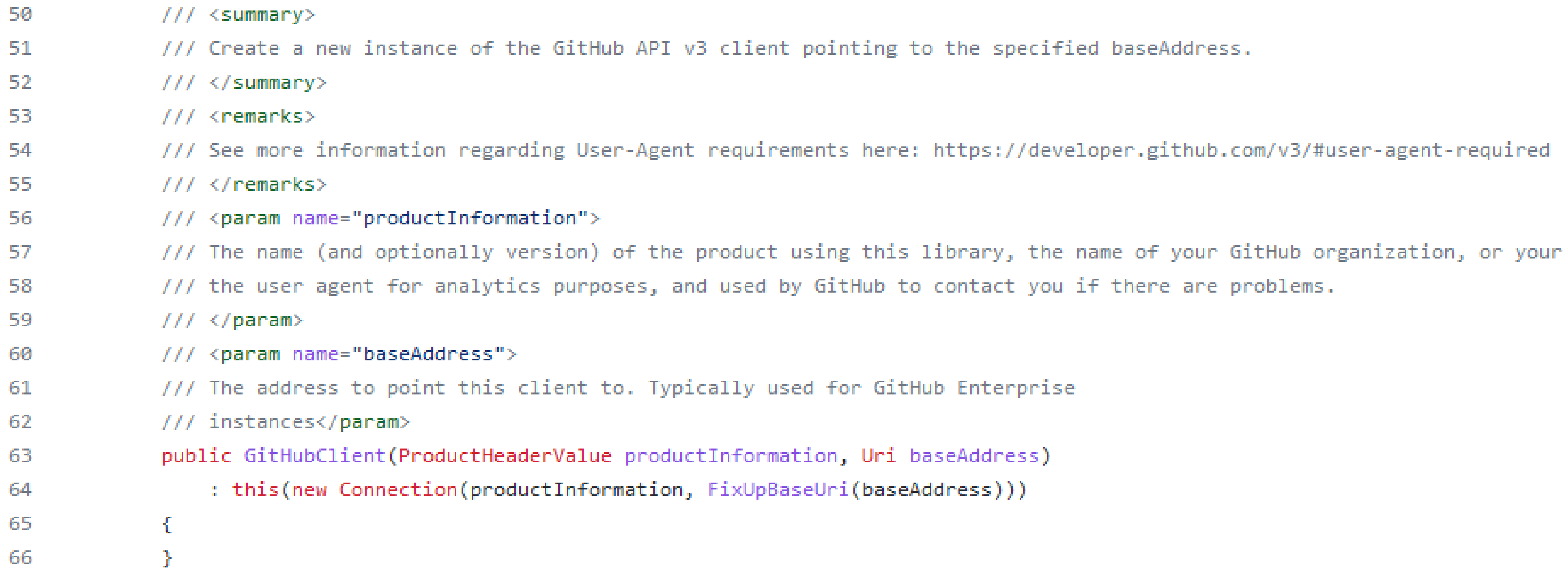
```csharp
using System;
using Octokit.Internal;

namespace Octokit
{
    /// <summary>
    /// A Client for the GitHub API v3. You can read more about the api here: http://developer.github.com.
    /// </summary>
    public class GitHubClient : IGitHubClient
    {
        /// <summary>
        /// The base address for the GitHub API
        /// </summary>
        public static readonly Uri GitHubApiUrl = new Uri("https://api.github.com/");
        internal static readonly Uri GitHubDotComUrl = new Uri("https://github.com/");

        /// <summary>
        /// Create a new instance of the GitHub API v3 client pointing to
        /// https://api.github.com/
        /// </summary>
        /// <remarks>
        /// See more information regarding User-Agent requirements here: https://developer.github.com/v3/#user-agent-required
        /// </remarks>
        /// <param name="productInformation">
        /// The name (and optionally version) of the product using this library, the name of your GitHub organization, or your
        /// the user agent for analytics purposes, and used by GitHub to contact you if there are problems.
        /// </param>
        public GitHubClient(ProductHeaderValue productInformation)
            : this(new Connection(productInformation, GitHubApiUrl))
        {
        }
```

Octokit



```
/// <summary>
/// Create a new instance of the GitHub API v3 client pointing to
/// https://api.github.com/
/// </summary>
/// <remarks>
/// See more information regarding User-Agent requirements here: https://developer.github.com/v3/#user-agent-required
/// </remarks>
/// <param name="productInformation">
/// The name (and optionally version) of the product using this library, the name of your GitHub organization, or your
/// the user agent for analytics purposes, and used by GitHub to contact you if there are problems.
/// </param>
/// <param name="credentialStore">Provides credentials to the client when making requests</param>
public GitHubClient(ProductHeaderValue productInformation, ICredentialStore credentialStore)
    : this(new Connection(productInformation, credentialStore))
{
}
```

Octokit

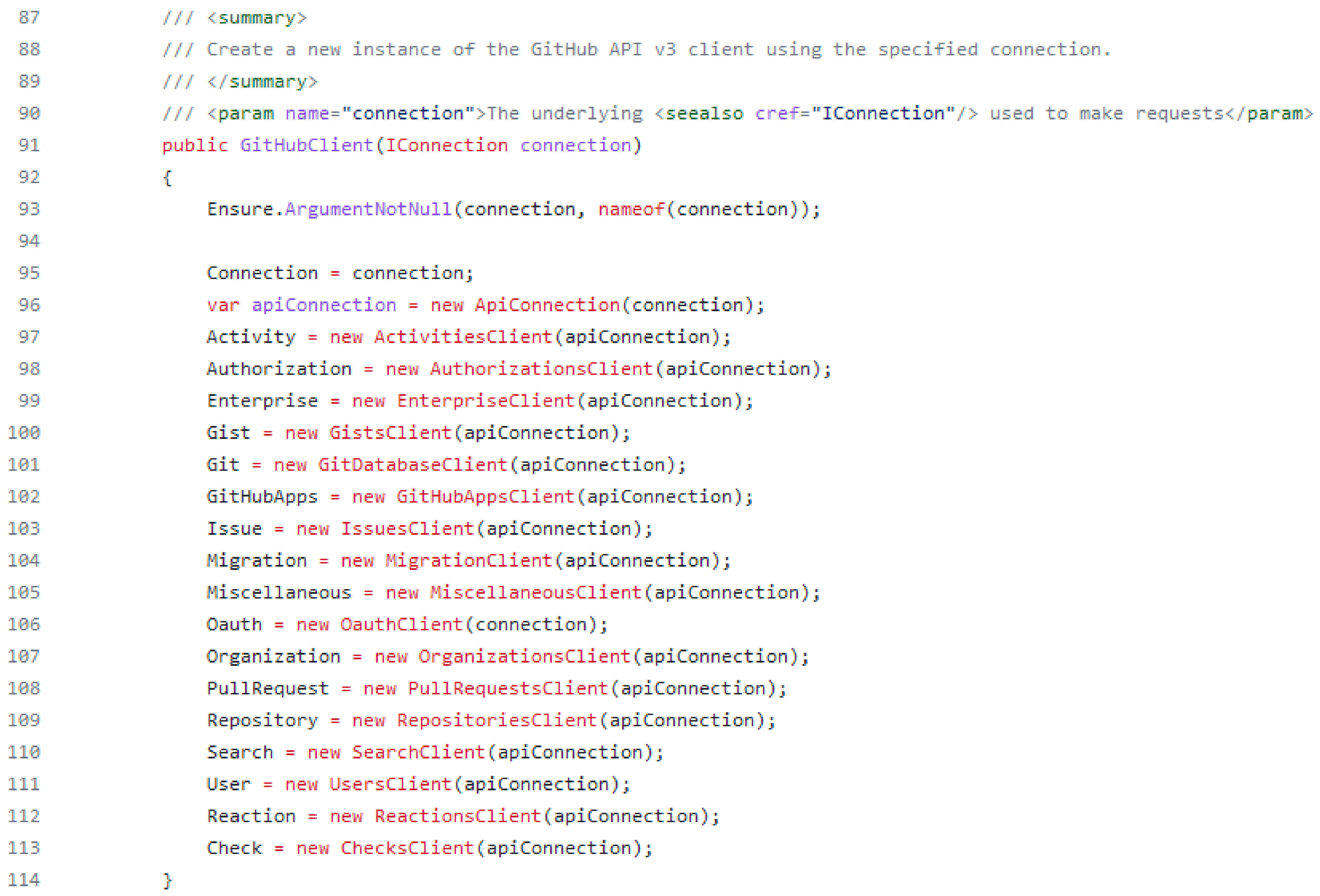


```csharp
        /// <summary>
        /// Create a new instance of the GitHub API v3 client pointing to the specified baseAddress.
        /// </summary>
        /// <remarks>
        /// See more information regarding User-Agent requirements here: https://developer.github.com/v3/#user-agent-required
        /// </remarks>
        /// <param name="productInformation">
        /// The name (and optionally version) of the product using this library, the name of your GitHub organization, or your
        /// the user agent for analytics purposes, and used by GitHub to contact you if there are problems.
        /// </param>
        /// <param name="baseAddress">
        /// The address to point this client to. Typically used for GitHub Enterprise
        /// instances</param>
        public GitHubClient(ProductHeaderValue productInformation, Uri baseAddress)
            : this(new Connection(productInformation, FixUpBaseUri(baseAddress)))
        {
        }
```

Octokit

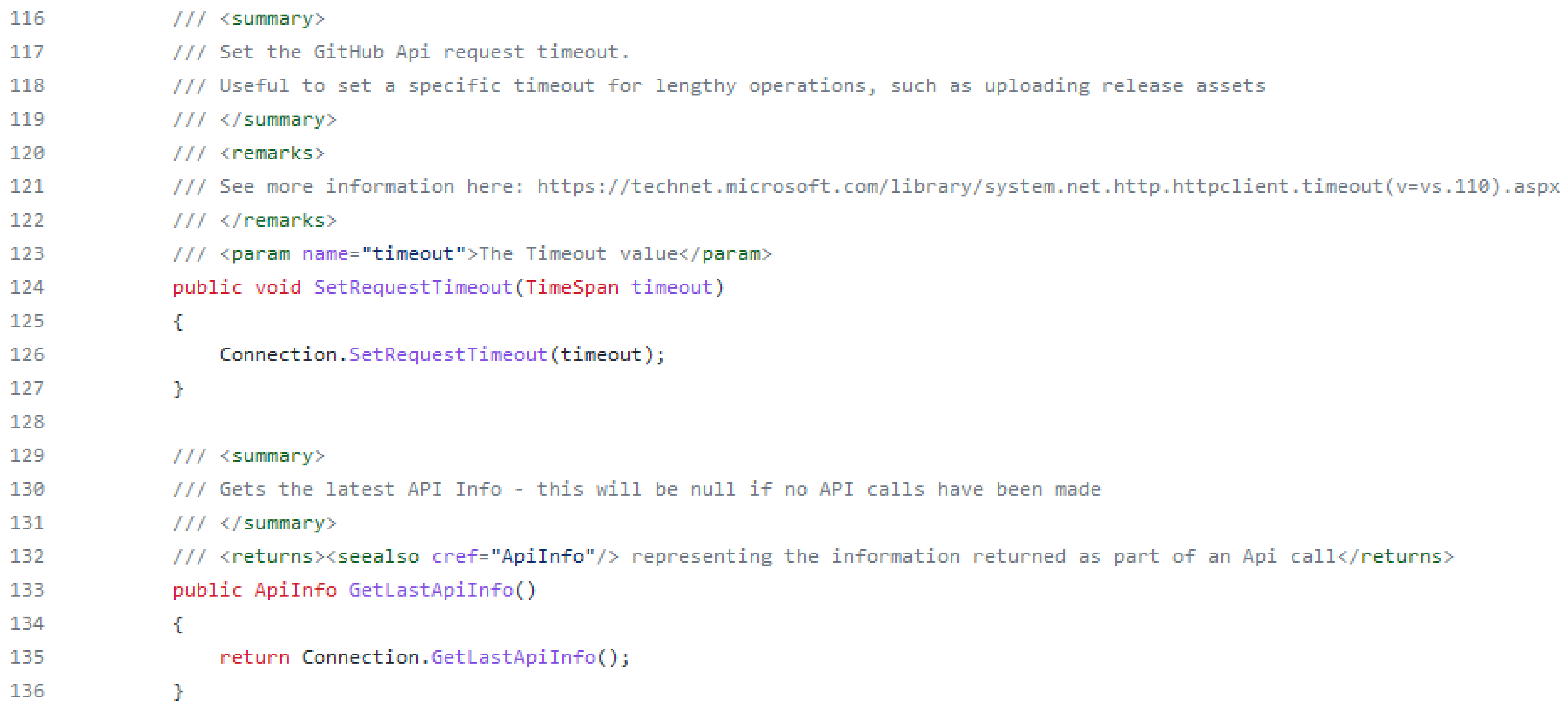
```
        /// <summary>
        /// Create a new instance of the GitHub API v3 client pointing to the specified baseAddress.
        /// </summary>
        /// <remarks>
        /// See more information regarding User-Agent requirements here: https://developer.github.com/v3/#user-agent-required
        /// </remarks>
        /// <param name="productInformation">
        /// The name (and optionally version) of the product using this library, the name of your GitHub organization, or your
        /// the user agent for analytics purposes, and used by GitHub to contact you if there are problems.
        /// </param>
        /// <param name="credentialStore">Provides credentials to the client when making requests</param>
        /// <param name="baseAddress">
        /// The address to point this client to. Typically used for GitHub Enterprise
        /// instances</param>
        public GitHubClient(ProductHeaderValue productInformation, ICredentialStore credentialStore, Uri baseAddress)
            : this(new Connection(productInformation, FixUpBaseUri(baseAddress), credentialStore))
        {
        }
```

Octokit



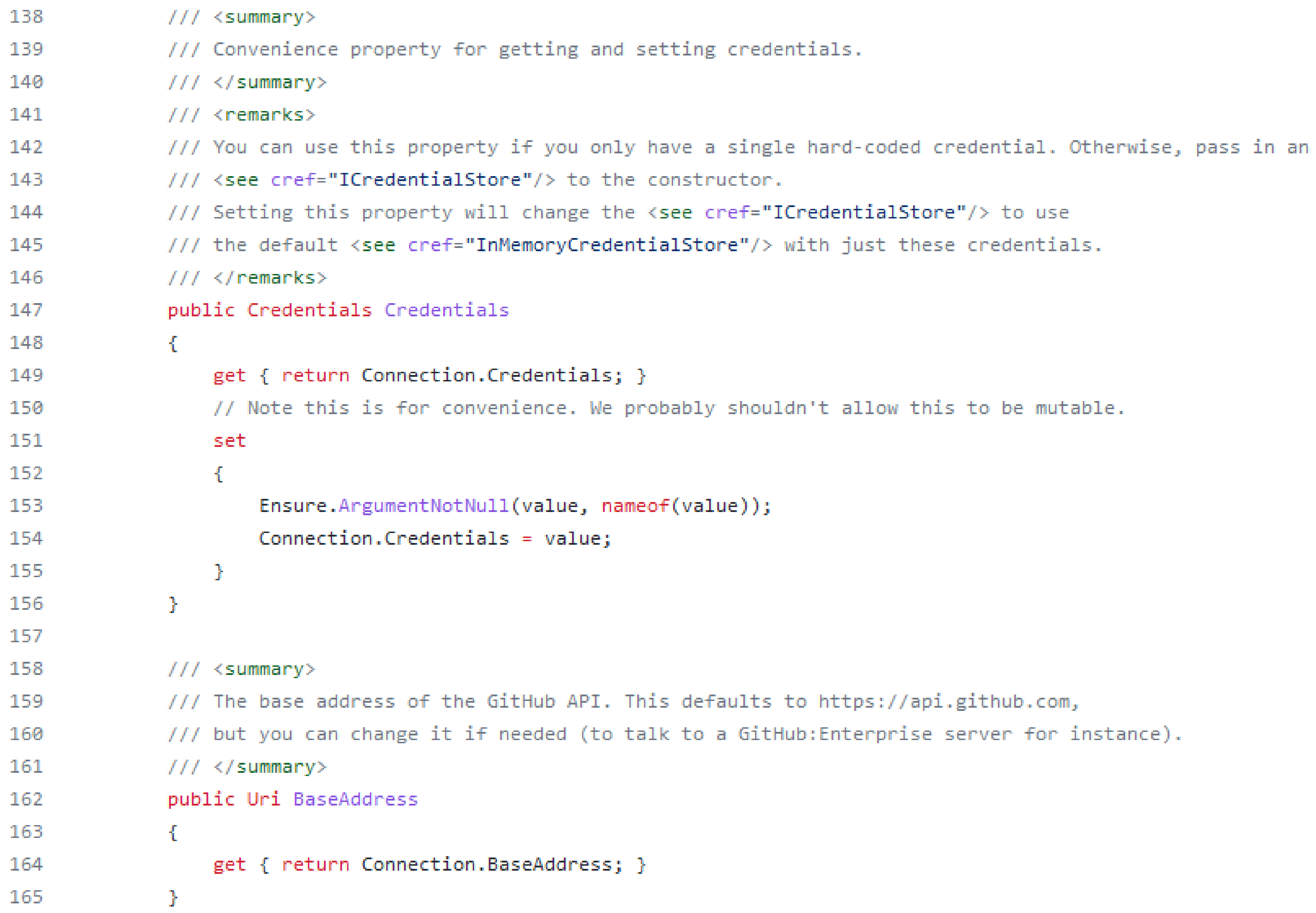
```csharp
/// <summary>
/// Create a new instance of the GitHub API v3 client using the specified connection.
/// </summary>
/// <param name="connection">The underlying <seealso cref="IConnection"/> used to make requests</param>
public GitHubClient(IConnection connection)
{
    Ensure.ArgumentNotNull(connection, nameof(connection));

    Connection = connection;
    var apiConnection = new ApiConnection(connection);
    Activity = new ActivitiesClient(apiConnection);
    Authorization = new AuthorizationsClient(apiConnection);
    Enterprise = new EnterpriseClient(apiConnection);
    Gist = new GistsClient(apiConnection);
    Git = new GitDatabaseClient(apiConnection);
    GitHubApps = new GitHubAppsClient(apiConnection);
    Issue = new IssuesClient(apiConnection);
    Migration = new MigrationClient(apiConnection);
    Miscellaneous = new MiscellaneousClient(apiConnection);
    Oauth = new OauthClient(connection);
    Organization = new OrganizationsClient(apiConnection);
    PullRequest = new PullRequestsClient(apiConnection);
    Repository = new RepositoriesClient(apiConnection);
    Search = new SearchClient(apiConnection);
    User = new UsersClient(apiConnection);
    Reaction = new ReactionsClient(apiConnection);
    Check = new ChecksClient(apiConnection);
}
```

Octokit



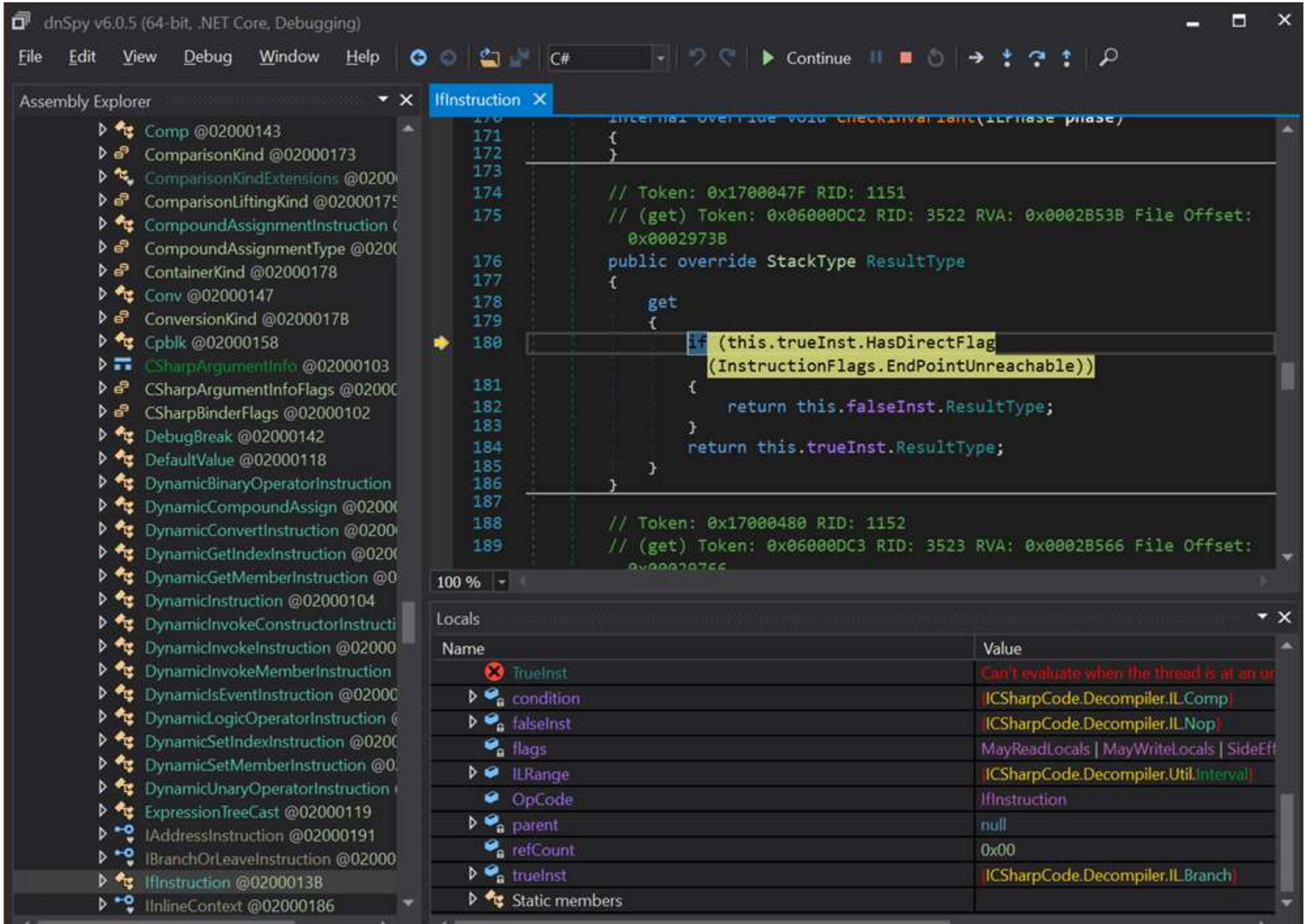
```
        /// <summary>
        /// Set the GitHub Api request timeout.
        /// Useful to set a specific timeout for lengthy operations, such as uploading release assets
        /// </summary>
        /// <remarks>
        /// See more information here: https://technet.microsoft.com/library/system.net.http.httpclient.timeout(v=vs.110).aspx
        /// </remarks>
        /// <param name="timeout">The Timeout value</param>
        public void SetRequestTimeout(TimeSpan timeout)
        {
            Connection.SetRequestTimeout(timeout);
        }

        /// <summary>
        /// Gets the latest API Info - this will be null if no API calls have been made
        /// </summary>
        /// <returns><seealso cref="ApiInfo"/> representing the information returned as part of an Api call</returns>
        public ApiInfo GetLastApiInfo()
        {
            return Connection.GetLastApiInfo();
        }
```

Octokit



```csharp
        /// <summary>
        /// Convenience property for getting and setting credentials.
        /// </summary>
        /// <remarks>
        /// You can use this property if you only have a single hard-coded credential. Otherwise, pass in an
        /// <see cref="ICredentialStore"/> to the constructor.
        /// Setting this property will change the <see cref="ICredentialStore"/> to use
        /// the default <see cref="InMemoryCredentialStore"/> with just these credentials.
        /// </remarks>
        public Credentials Credentials
        {
            get { return Connection.Credentials; }
            // Note this is for convenience. We probably shouldn't allow this to be mutable.
            set
            {
                Ensure.ArgumentNotNull(value, nameof(value));
                Connection.Credentials = value;
            }
        }

        /// <summary>
        /// The base address of the GitHub API. This defaults to https://api.github.com,
        /// but you can change it if needed (to talk to a GitHub:Enterprise server for instance).
        /// </summary>
        public Uri BaseAddress
        {
            get { return Connection.BaseAddress; }
        }
```

ILSpy

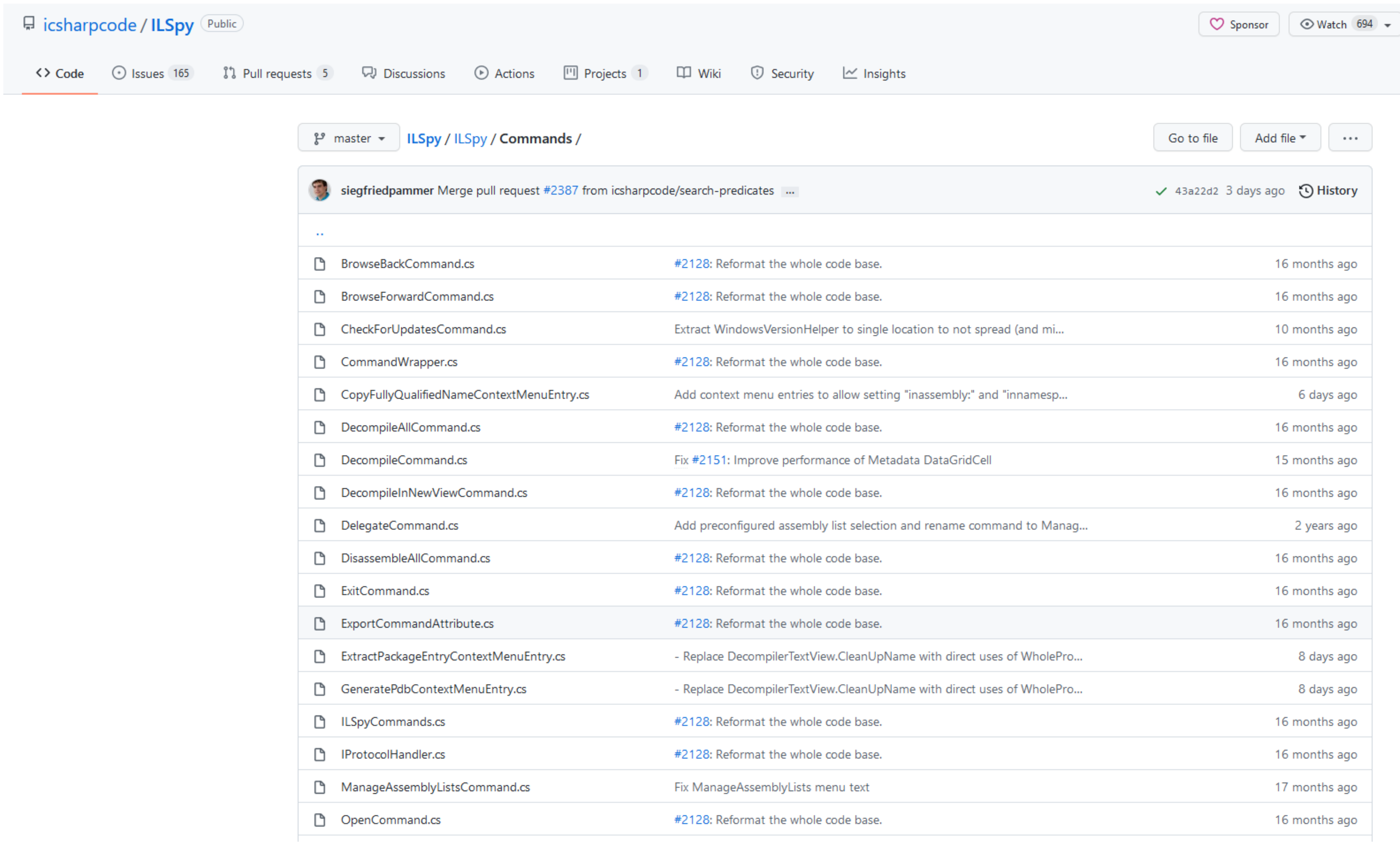

dnSpy v6.0.5 (64-bit, .NET Core, Debugging)
File Edit View Debug Window Help
C#
Continue
Assembly Explorer
IfInstruction
100 %
Locals

```
Comp @02000143
ComparisonKind @02000173
ComparisonKindExtensions @0200
ComparisonLiftingKind @0200017
CompoundAssignmentInstruction
CompoundAssignmentType @020
ContainerKind @02000178
Conv @02000147
ConversionKind @0200017B
Cpblk @02000158
CSharpArgumentInfo @02000103
CSharpArgumentInfoFlags @0200
CSharpBinderFlags @02000102
DebugBreak @02000142
DefaultValue @02000118
DynamicBinaryOperatorInstruction
DynamicCompoundAssign @0200
DynamicConvertInstruction @0200
DynamicGetIndexInstruction @020
DynamicGetMemberInstruction @0
DynamicInstruction @02000104
DynamicInvokeConstructorInstructi
DynamicInvokeInstruction @02000
DynamicInvokeMemberInstruction
DynamicIsEventInstruction @0200
DynamicLogicOperatorInstruction
DynamicSetIndexInstruction @020
DynamicSetMemberInstruction @0
DynamicUnaryOperatorInstruction
ExpressionTreeCast @02000119
IAddressInstruction @02000191
IBranchOrLeaveInstruction @0200
IfInstruction @0200013B
IInlineContext @02000186
```

```
        internal override void CheckInvariant(ILPhase phase)
        {
        }

        // Token: 0x1700047F RID: 1151
        // (get) Token: 0x06000DC2 RID: 3522 RVA: 0x0002B53B File Offset:
          0x0002973B
        public override StackType ResultType
        {
            get
            {
                if (this.trueInst.HasDirectFlag
                (InstructionFlags.EndPointUnreachable))
                {
                    return this.falseInst.ResultType;
                }
                return this.trueInst.ResultType;
            }
        }

        // Token: 0x17000480 RID: 1152
        // (get) Token: 0x06000DC3 RID: 3523 RVA: 0x0002B566 File Offset:
```

Name	Value
TrueInst	Can't evaluate when the thread is at an u
condition	{ICSharpCode.Decompiler.IL.Comp}
falseInst	{ICSharpCode.Decompiler.IL.Nop}
flags	MayReadLocals | MayWriteLocals | SideEff
ILRange	{ICSharpCode.Decompiler.Util.Interval}
OpCode	IfInstruction
parent	null
refCount	0x00
trueInst	{ICSharpCode.Decompiler.IL.Branch}
Static members	

ILSpy



icsharpcode / ILSpy Public

Sponsor · Watch 694

<> Code · Issues 165 · Pull requests 5 · Discussions · Actions · Projects 1 · Wiki · Security · Insights

master · ILSpy / ILSpy / Commands /

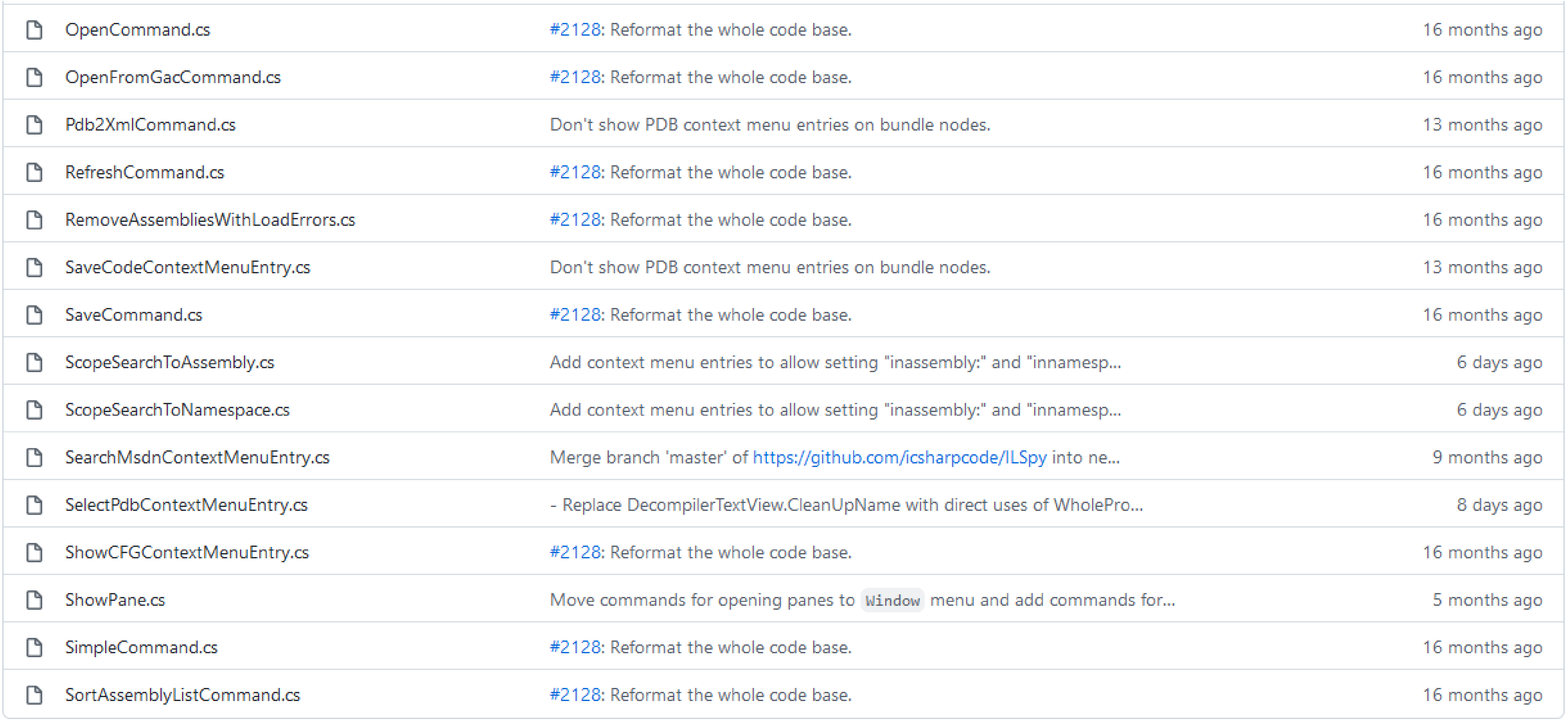
Go to file · Add file · ...

siegfriedpammer Merge pull request #2387 from icsharpcode/search-predicates ... ✓ 43a22d2 3 days ago History

..

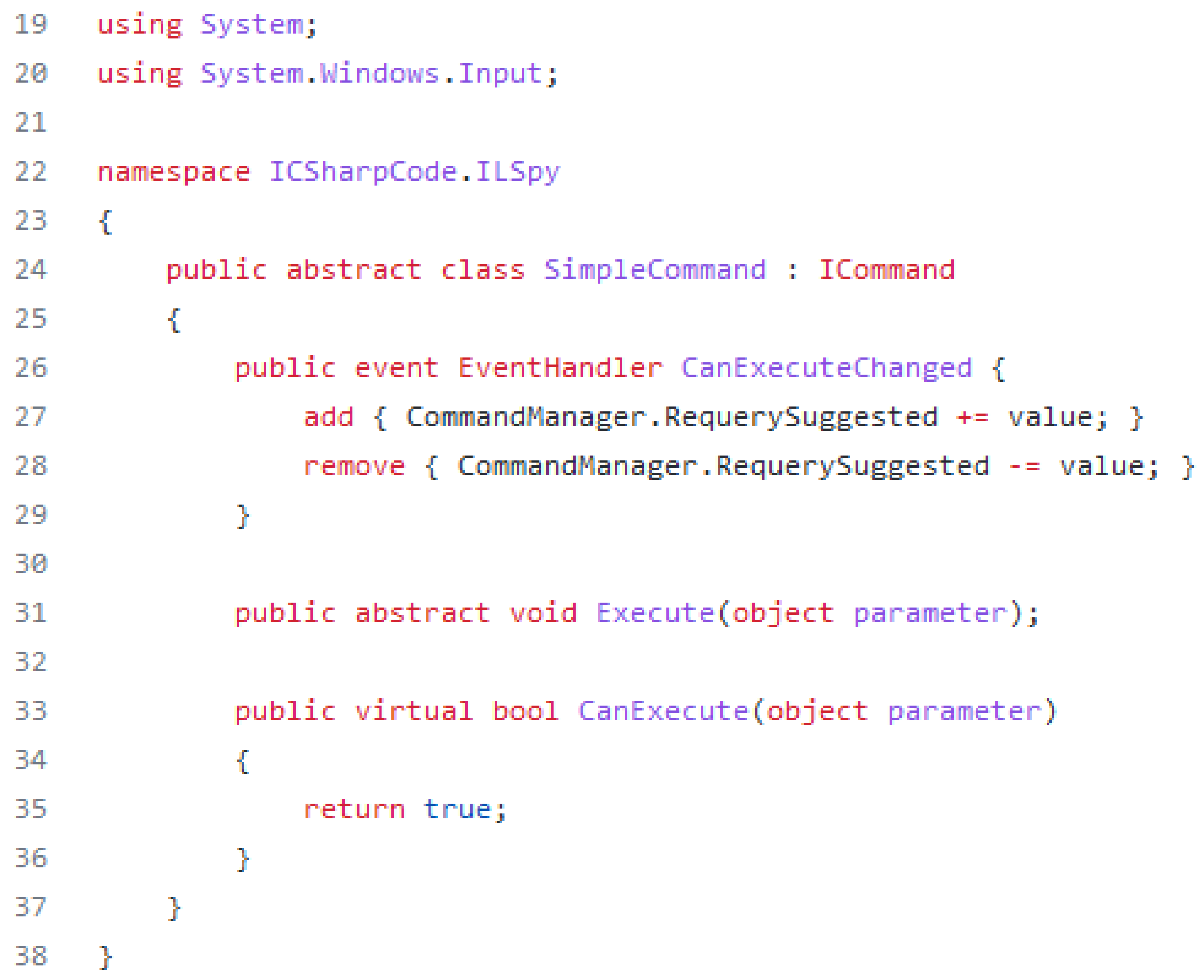
BrowseBackCommand.cs	#2128: Reformat the whole code base.	16 months ago
BrowseForwardCommand.cs	#2128: Reformat the whole code base.	16 months ago
CheckForUpdatesCommand.cs	Extract WindowsVersionHelper to single location to not spread (and mi...	10 months ago
CommandWrapper.cs	#2128: Reformat the whole code base.	16 months ago
CopyFullyQualifiedNameContextMenuEntry.cs	Add context menu entries to allow setting "inassembly:" and "innamesp...	6 days ago
DecompileAllCommand.cs	#2128: Reformat the whole code base.	16 months ago
DecompileCommand.cs	Fix #2151: Improve performance of Metadata DataGridCell	15 months ago
DecompileInNewViewCommand.cs	#2128: Reformat the whole code base.	16 months ago
DelegateCommand.cs	Add preconfigured assembly list selection and rename command to Manag...	2 years ago
DisassembleAllCommand.cs	#2128: Reformat the whole code base.	16 months ago
ExitCommand.cs	#2128: Reformat the whole code base.	16 months ago
ExportCommandAttribute.cs	#2128: Reformat the whole code base.	16 months ago
ExtractPackageEntryContextMenuEntry.cs	- Replace DecompilerTextView.CleanUpName with direct uses of WholePro...	8 days ago
GeneratePdbContextMenuEntry.cs	- Replace DecompilerTextView.CleanUpName with direct uses of WholePro...	8 days ago
ILSpyCommands.cs	#2128: Reformat the whole code base.	16 months ago
IProtocolHandler.cs	#2128: Reformat the whole code base.	16 months ago
ManageAssemblyListsCommand.cs	Fix ManageAssemblyLists menu text	17 months ago
OpenCommand.cs	#2128: Reformat the whole code base.	16 months ago

ILSpy



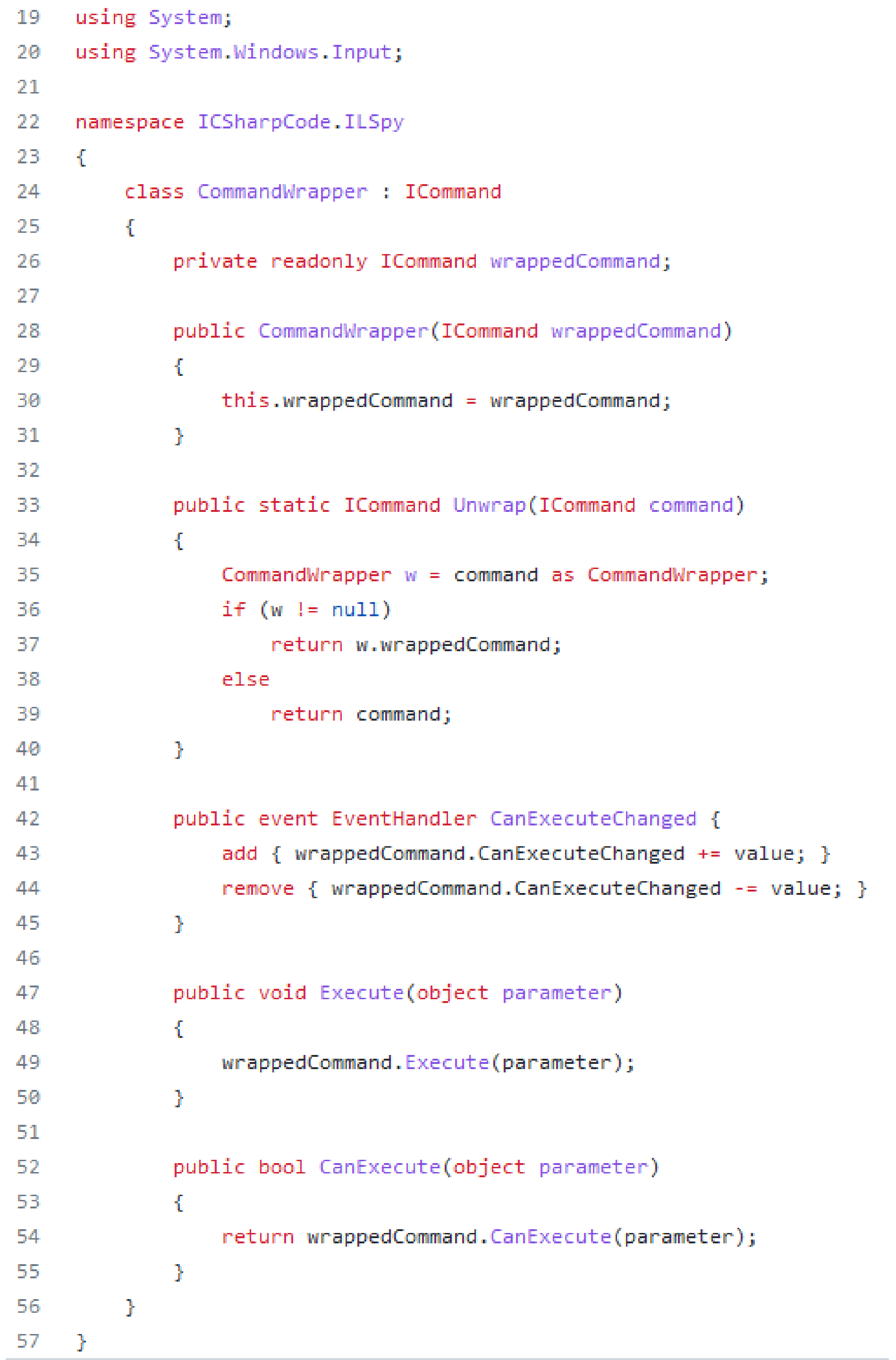
OpenCommand.cs	#2128: Reformat the whole code base.	16 months ago
OpenFromGacCommand.cs	#2128: Reformat the whole code base.	16 months ago
Pdb2XmlCommand.cs	Don't show PDB context menu entries on bundle nodes.	13 months ago
RefreshCommand.cs	#2128: Reformat the whole code base.	16 months ago
RemoveAssembliesWithLoadErrors.cs	#2128: Reformat the whole code base.	16 months ago
SaveCodeContextMenuEntry.cs	Don't show PDB context menu entries on bundle nodes.	13 months ago
SaveCommand.cs	#2128: Reformat the whole code base.	16 months ago
ScopeSearchToAssembly.cs	Add context menu entries to allow setting "inassembly:" and "innamesp…	6 days ago
ScopeSearchToNamespace.cs	Add context menu entries to allow setting "inassembly:" and "innamesp…	6 days ago
SearchMsdnContextMenuEntry.cs	Merge branch 'master' of https://github.com/icsharpcode/ILSpy into ne…	9 months ago
SelectPdbContextMenuEntry.cs	- Replace DecompilerTextView.CleanUpName with direct uses of WholePro…	8 days ago
ShowCFGContextMenuEntry.cs	#2128: Reformat the whole code base.	16 months ago
ShowPane.cs	Move commands for opening panes to `Window` menu and add commands for…	5 months ago
SimpleCommand.cs	#2128: Reformat the whole code base.	16 months ago
SortAssemblyListCommand.cs	#2128: Reformat the whole code base.	16 months ago

ILSpy



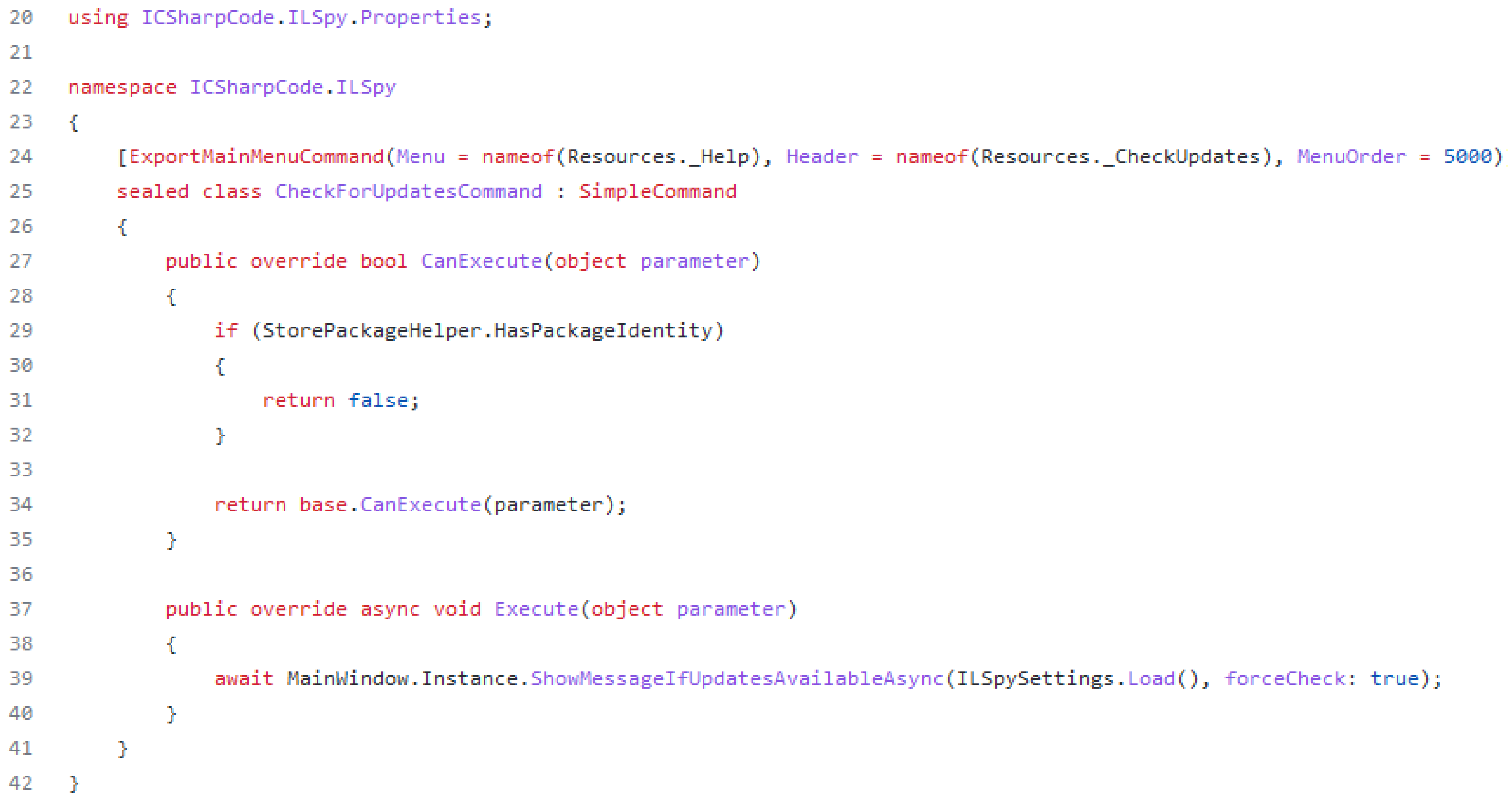
```csharp
using System;
using System.Windows.Input;

namespace ICSharpCode.ILSpy
{
    public abstract class SimpleCommand : ICommand
    {
        public event EventHandler CanExecuteChanged {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }

        public abstract void Execute(object parameter);

        public virtual bool CanExecute(object parameter)
        {
            return true;
        }
    }
}
```

ILSpy



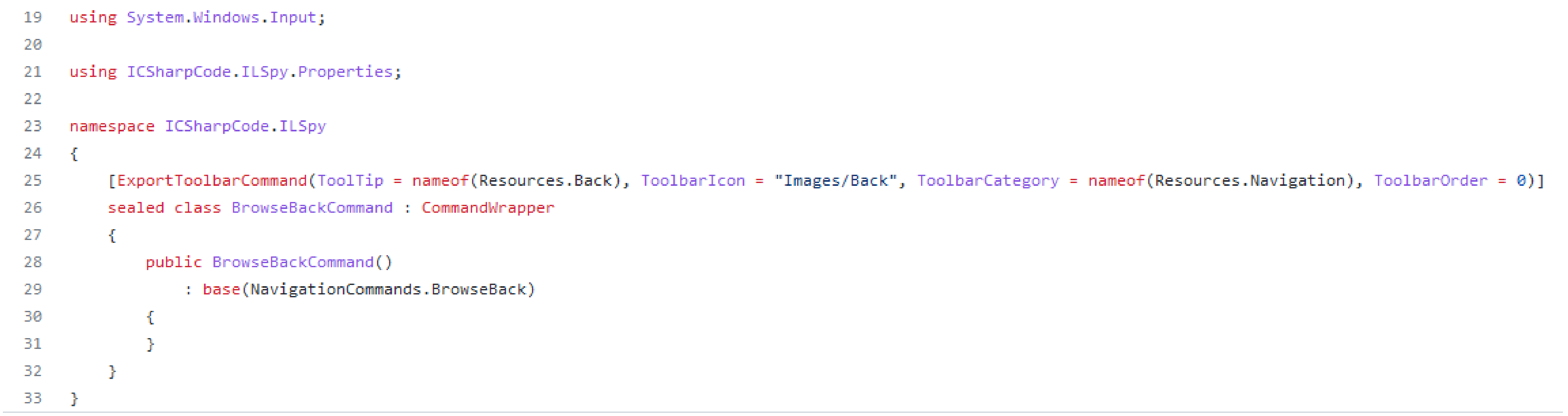
```csharp
using System;
using System.Windows.Input;

namespace ICSharpCode.ILSpy
{
    class CommandWrapper : ICommand
    {
        private readonly ICommand wrappedCommand;

        public CommandWrapper(ICommand wrappedCommand)
        {
            this.wrappedCommand = wrappedCommand;
        }

        public static ICommand Unwrap(ICommand command)
        {
            CommandWrapper w = command as CommandWrapper;
            if (w != null)
                return w.wrappedCommand;
            else
                return command;
        }

        public event EventHandler CanExecuteChanged {
            add { wrappedCommand.CanExecuteChanged += value; }
            remove { wrappedCommand.CanExecuteChanged -= value; }
        }

        public void Execute(object parameter)
        {
            wrappedCommand.Execute(parameter);
        }

        public bool CanExecute(object parameter)
        {
            return wrappedCommand.CanExecute(parameter);
        }
    }
}
```

ILSpy



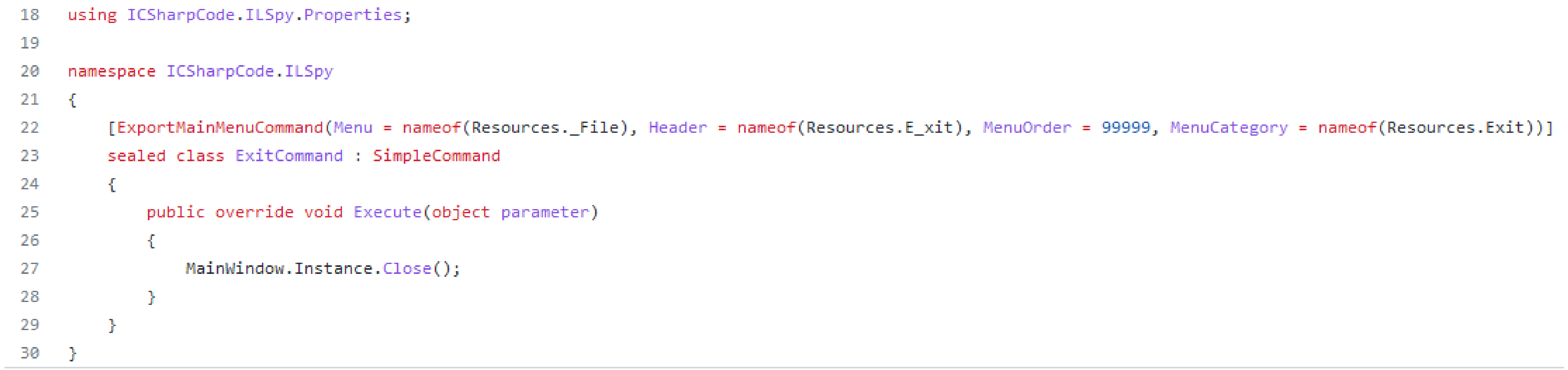
```
using ICSharpCode.ILSpy.Properties;

namespace ICSharpCode.ILSpy
{
    [ExportMainMenuCommand(Menu = nameof(Resources._Help), Header = nameof(Resources._CheckUpdates), MenuOrder = 5000)]
    sealed class CheckForUpdatesCommand : SimpleCommand
    {
        public override bool CanExecute(object parameter)
        {
            if (StorePackageHelper.HasPackageIdentity)
            {
                return false;
            }

            return base.CanExecute(parameter);
        }

        public override async void Execute(object parameter)
        {
            await MainWindow.Instance.ShowMessageIfUpdatesAvailableAsync(ILSpySettings.Load(), forceCheck: true);
        }
    }
}
```

ILSpy



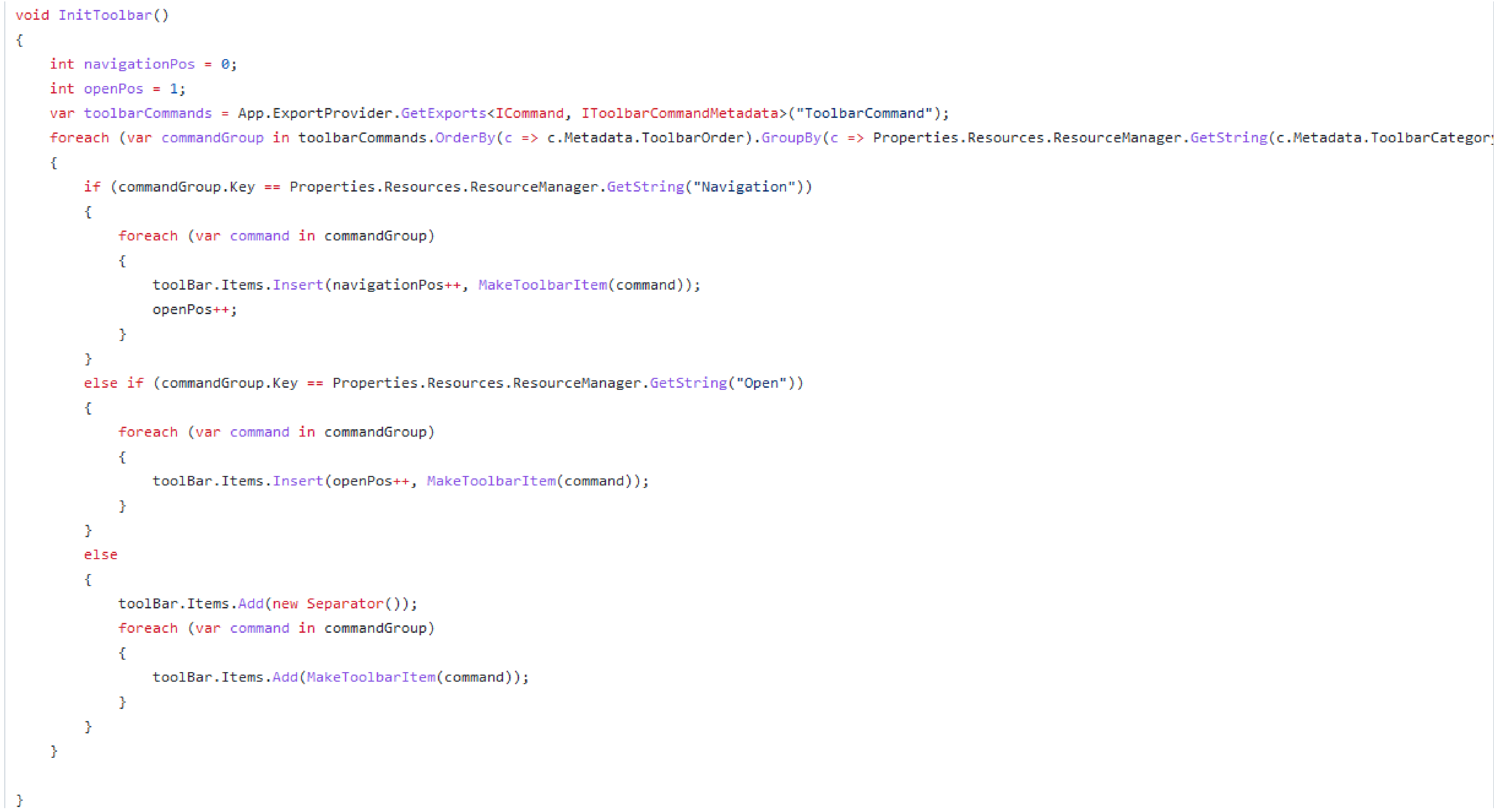
```csharp
using System.Windows.Input;

using ICSharpCode.ILSpy.Properties;

namespace ICSharpCode.ILSpy
{
    [ExportToolbarCommand(ToolTip = nameof(Resources.Back), ToolbarIcon = "Images/Back", ToolbarCategory = nameof(Resources.Navigation), ToolbarOrder = 0)]
    sealed class BrowseBackCommand : CommandWrapper
    {
        public BrowseBackCommand()
            : base(NavigationCommands.BrowseBack)
        {
        }
    }
}
```

ILSpy



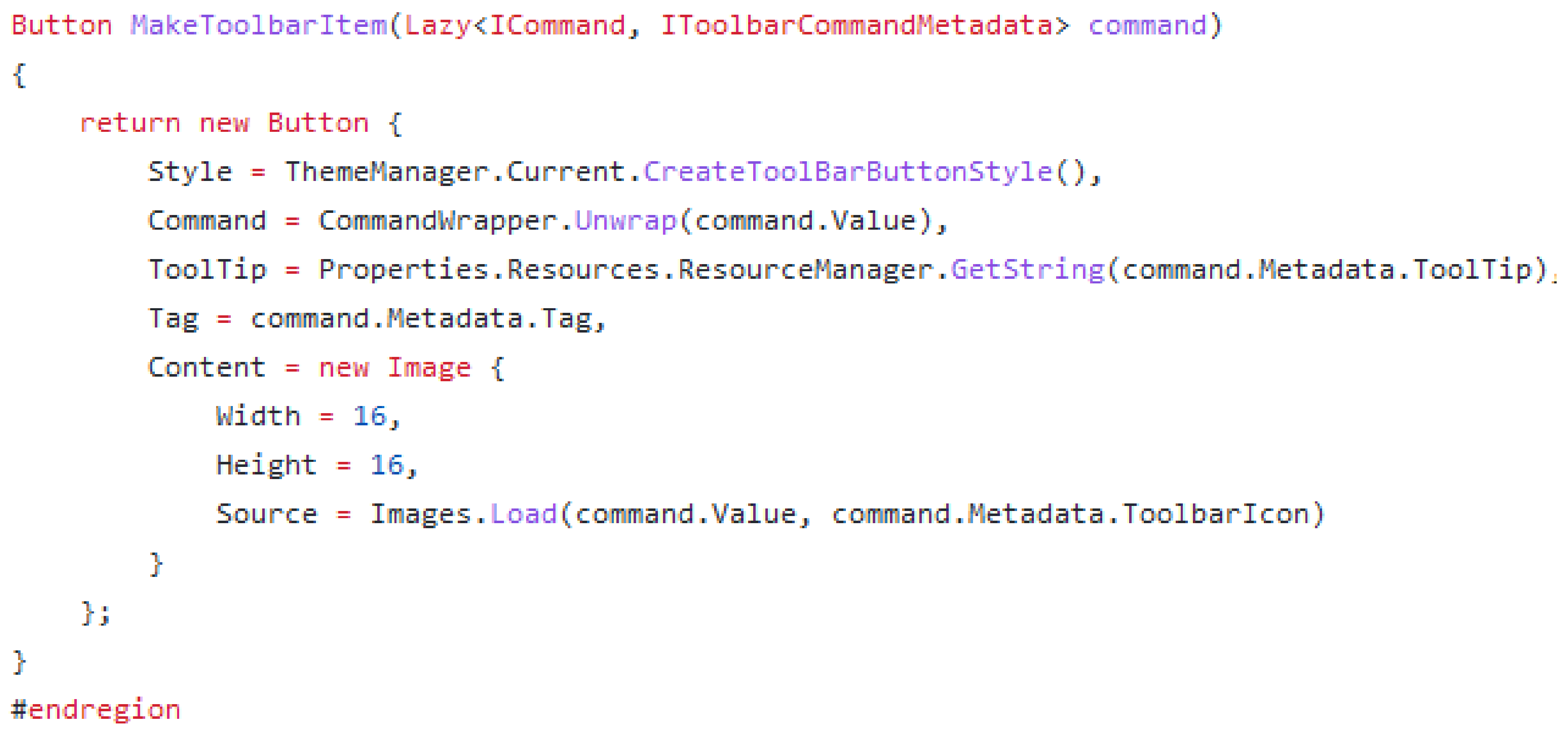
```
using ICSharpCode.ILSpy.Properties;

namespace ICSharpCode.ILSpy
{
    [ExportMainMenuCommand(Menu = nameof(Resources._File), Header = nameof(Resources.E_xit), MenuOrder = 99999, MenuCategory = nameof(Resources.Exit))]
    sealed class ExitCommand : SimpleCommand
    {
        public override void Execute(object parameter)
        {
            MainWindow.Instance.Close();
        }
    }
}
```

ILSpy



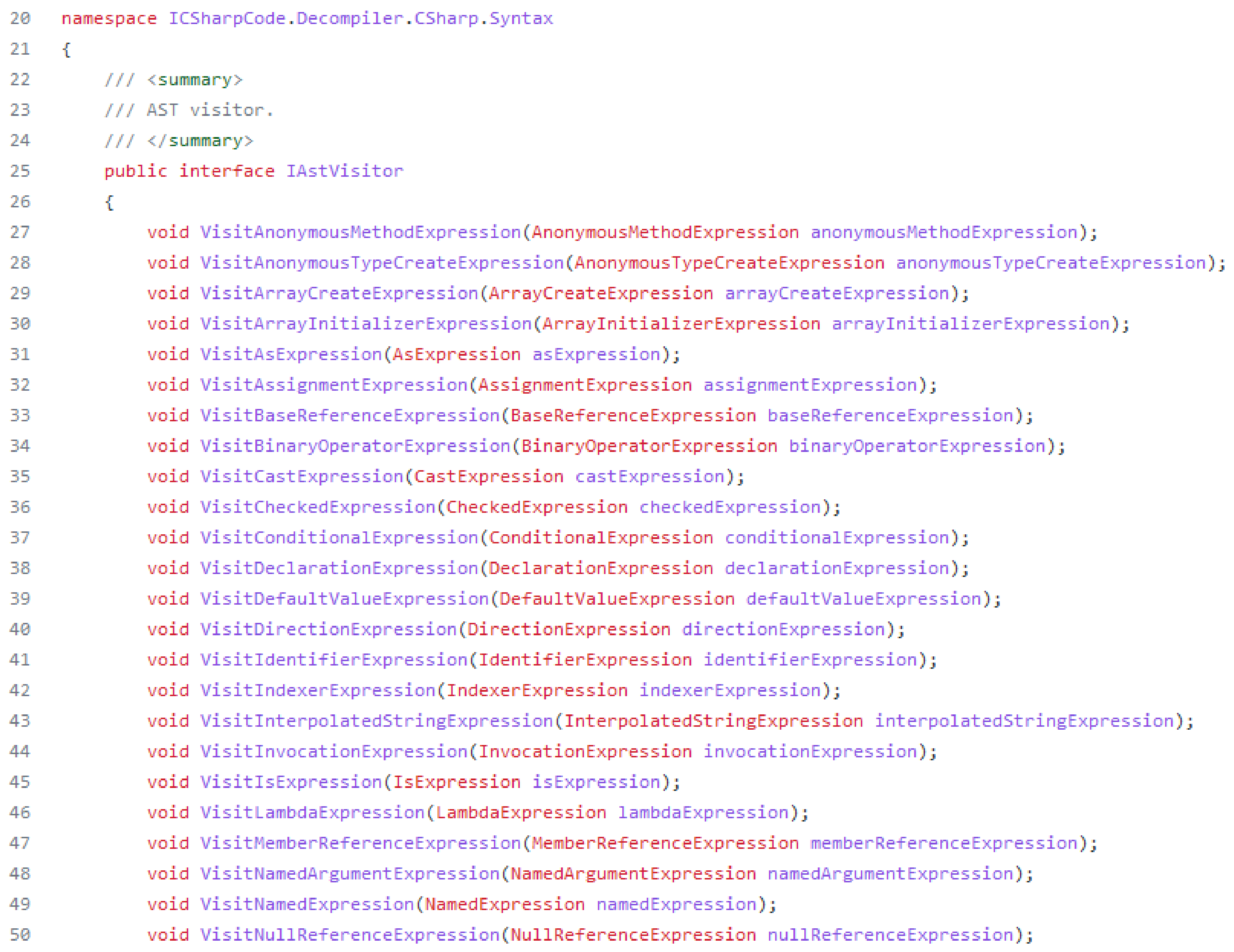
```
void InitToolbar()
{
    int navigationPos = 0;
    int openPos = 1;
    var toolbarCommands = App.ExportProvider.GetExports<ICommand, IToolbarCommandMetadata>("ToolbarCommand");
    foreach (var commandGroup in toolbarCommands.OrderBy(c => c.Metadata.ToolbarOrder).GroupBy(c => Properties.Resources.ResourceManager.GetString(c.Metadata.ToolbarCategor
    {
        if (commandGroup.Key == Properties.Resources.ResourceManager.GetString("Navigation"))
        {
            foreach (var command in commandGroup)
            {
                toolBar.Items.Insert(navigationPos++, MakeToolbarItem(command));
                openPos++;
            }
        }
        else if (commandGroup.Key == Properties.Resources.ResourceManager.GetString("Open"))
        {
            foreach (var command in commandGroup)
            {
                toolBar.Items.Insert(openPos++, MakeToolbarItem(command));
            }
        }
        else
        {
            toolBar.Items.Add(new Separator());
            foreach (var command in commandGroup)
            {
                toolBar.Items.Add(MakeToolbarItem(command));
            }
        }
    }

}
```

ILSpy



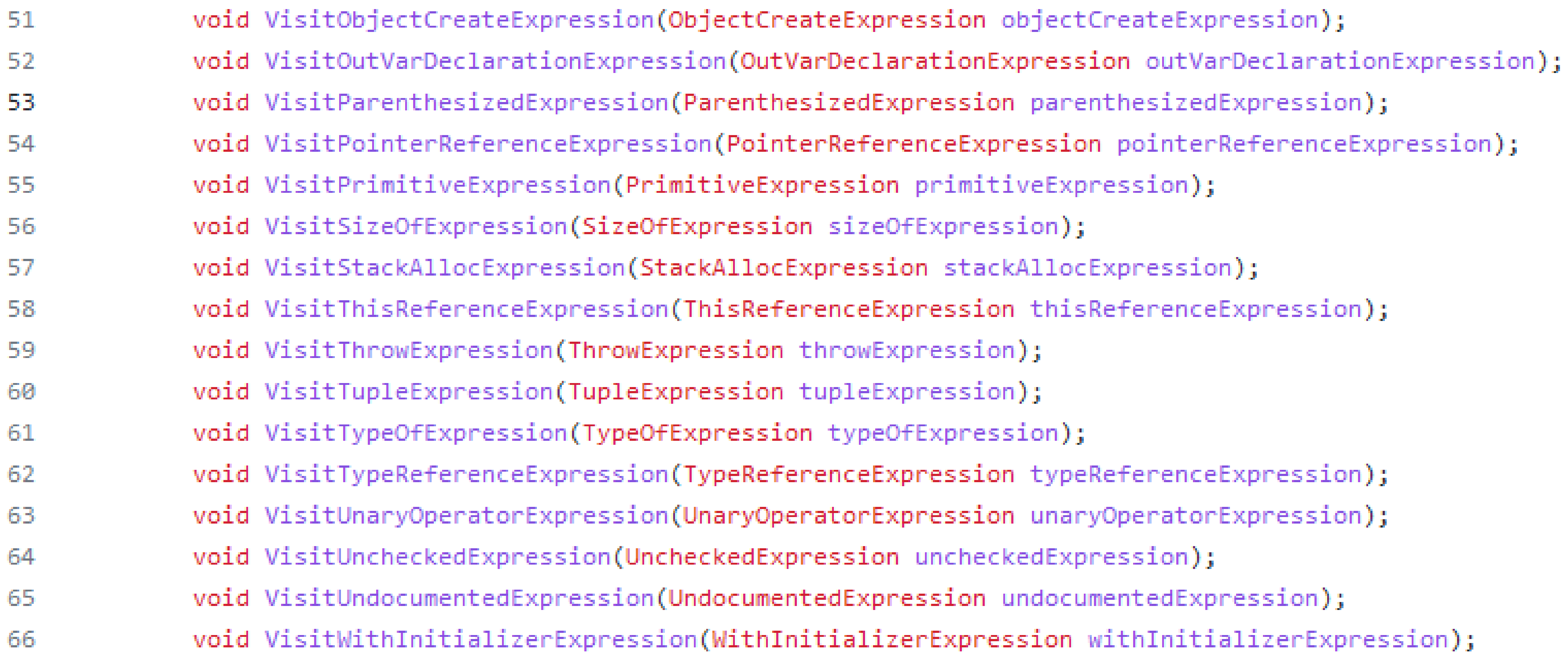
```
Button MakeToolbarItem(Lazy<ICommand, IToolbarCommandMetadata> command)
{
    return new Button {
        Style = ThemeManager.Current.CreateToolBarButtonStyle(),
        Command = CommandWrapper.Unwrap(command.Value),
        ToolTip = Properties.Resources.ResourceManager.GetString(command.Metadata.ToolTip),
        Tag = command.Metadata.Tag,
        Content = new Image {
            Width = 16,
            Height = 16,
            Source = Images.Load(command.Value, command.Metadata.ToolbarIcon)
        }
    };
}
#endregion
```

ILSpy



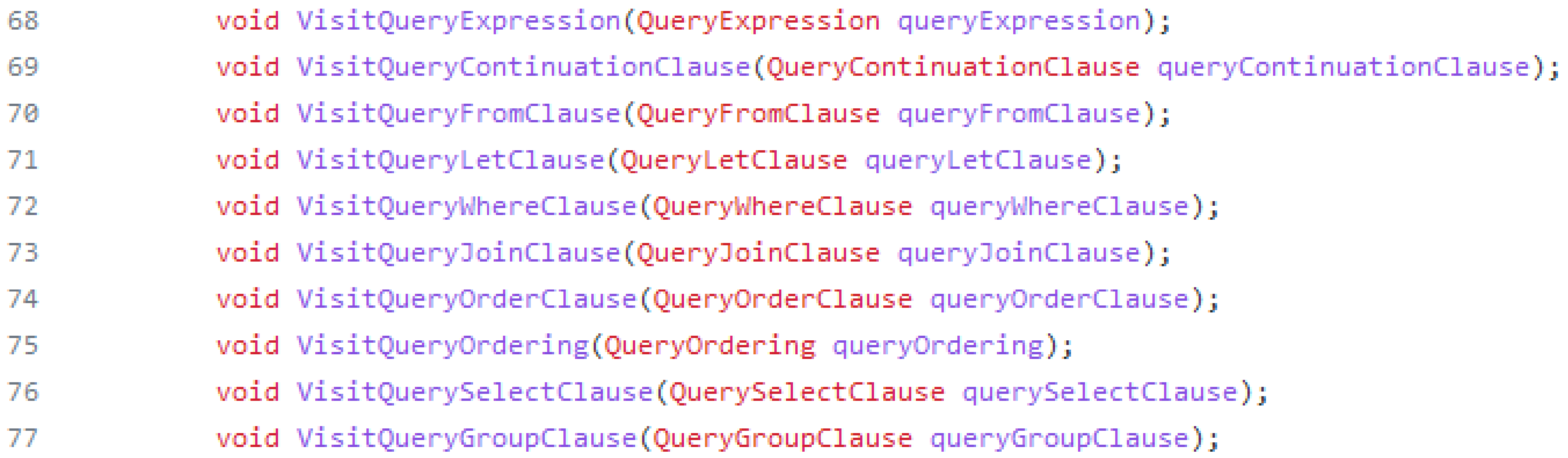
```
namespace ICSharpCode.Decompiler.CSharp.Syntax
{
    /// <summary>
    /// AST visitor.
    /// </summary>
    public interface IAstVisitor
    {
        void VisitAnonymousMethodExpression(AnonymousMethodExpression anonymousMethodExpression);
        void VisitAnonymousTypeCreateExpression(AnonymousTypeCreateExpression anonymousTypeCreateExpression);
        void VisitArrayCreateExpression(ArrayCreateExpression arrayCreateExpression);
        void VisitArrayInitializerExpression(ArrayInitializerExpression arrayInitializerExpression);
        void VisitAsExpression(AsExpression asExpression);
        void VisitAssignmentExpression(AssignmentExpression assignmentExpression);
        void VisitBaseReferenceExpression(BaseReferenceExpression baseReferenceExpression);
        void VisitBinaryOperatorExpression(BinaryOperatorExpression binaryOperatorExpression);
        void VisitCastExpression(CastExpression castExpression);
        void VisitCheckedExpression(CheckedExpression checkedExpression);
        void VisitConditionalExpression(ConditionalExpression conditionalExpression);
        void VisitDeclarationExpression(DeclarationExpression declarationExpression);
        void VisitDefaultValueExpression(DefaultValueExpression defaultValueExpression);
        void VisitDirectionExpression(DirectionExpression directionExpression);
        void VisitIdentifierExpression(IdentifierExpression identifierExpression);
        void VisitIndexerExpression(IndexerExpression indexerExpression);
        void VisitInterpolatedStringExpression(InterpolatedStringExpression interpolatedStringExpression);
        void VisitInvocationExpression(InvocationExpression invocationExpression);
        void VisitIsExpression(IsExpression isExpression);
        void VisitLambdaExpression(LambdaExpression lambdaExpression);
        void VisitMemberReferenceExpression(MemberReferenceExpression memberReferenceExpression);
        void VisitNamedArgumentExpression(NamedArgumentExpression namedArgumentExpression);
        void VisitNamedExpression(NamedExpression namedExpression);
        void VisitNullReferenceExpression(NullReferenceExpression nullReferenceExpression);
```

ILSpy



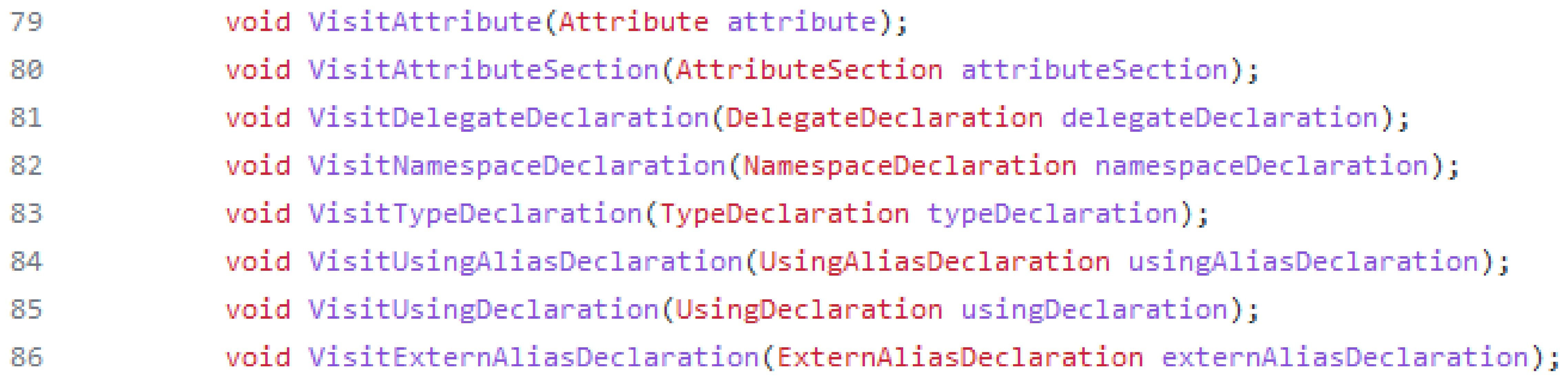
```
        void VisitObjectCreateExpression(ObjectCreateExpression objectCreateExpression);
        void VisitOutVarDeclarationExpression(OutVarDeclarationExpression outVarDeclarationExpression);
        void VisitParenthesizedExpression(ParenthesizedExpression parenthesizedExpression);
        void VisitPointerReferenceExpression(PointerReferenceExpression pointerReferenceExpression);
        void VisitPrimitiveExpression(PrimitiveExpression primitiveExpression);
        void VisitSizeOfExpression(SizeOfExpression sizeOfExpression);
        void VisitStackAllocExpression(StackAllocExpression stackAllocExpression);
        void VisitThisReferenceExpression(ThisReferenceExpression thisReferenceExpression);
        void VisitThrowExpression(ThrowExpression throwExpression);
        void VisitTupleExpression(TupleExpression tupleExpression);
        void VisitTypeOfExpression(TypeOfExpression typeOfExpression);
        void VisitTypeReferenceExpression(TypeReferenceExpression typeReferenceExpression);
        void VisitUnaryOperatorExpression(UnaryOperatorExpression unaryOperatorExpression);
        void VisitUncheckedExpression(UncheckedExpression uncheckedExpression);
        void VisitUndocumentedExpression(UndocumentedExpression undocumentedExpression);
        void VisitWithInitializerExpression(WithInitializerExpression withInitializerExpression);
```

ILSpy



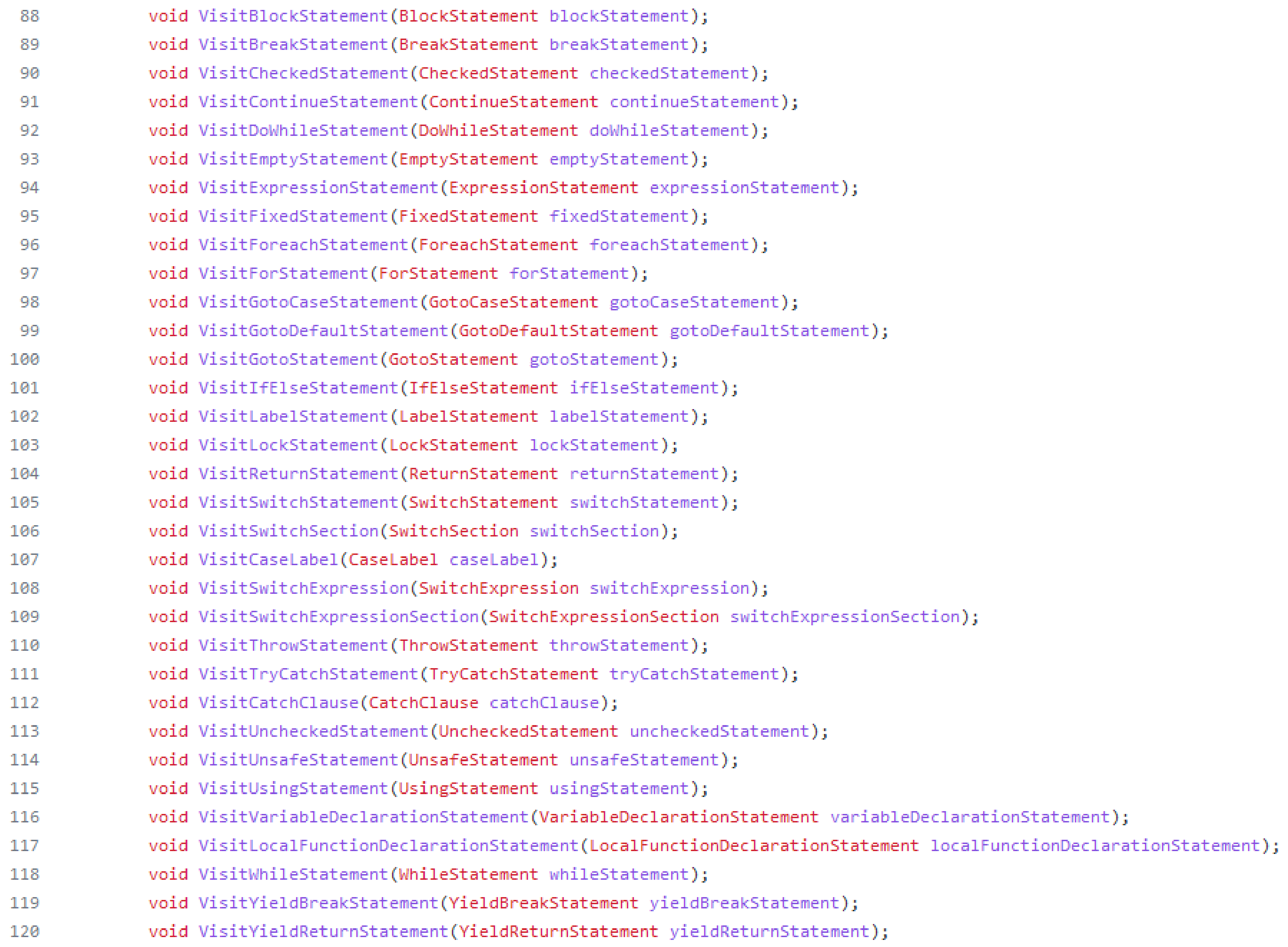
```
void VisitQueryExpression(QueryExpression queryExpression);
void VisitQueryContinuationClause(QueryContinuationClause queryContinuationClause);
void VisitQueryFromClause(QueryFromClause queryFromClause);
void VisitQueryLetClause(QueryLetClause queryLetClause);
void VisitQueryWhereClause(QueryWhereClause queryWhereClause);
void VisitQueryJoinClause(QueryJoinClause queryJoinClause);
void VisitQueryOrderClause(QueryOrderClause queryOrderClause);
void VisitQueryOrdering(QueryOrdering queryOrdering);
void VisitQuerySelectClause(QuerySelectClause querySelectClause);
void VisitQueryGroupClause(QueryGroupClause queryGroupClause);
```

ILSpy



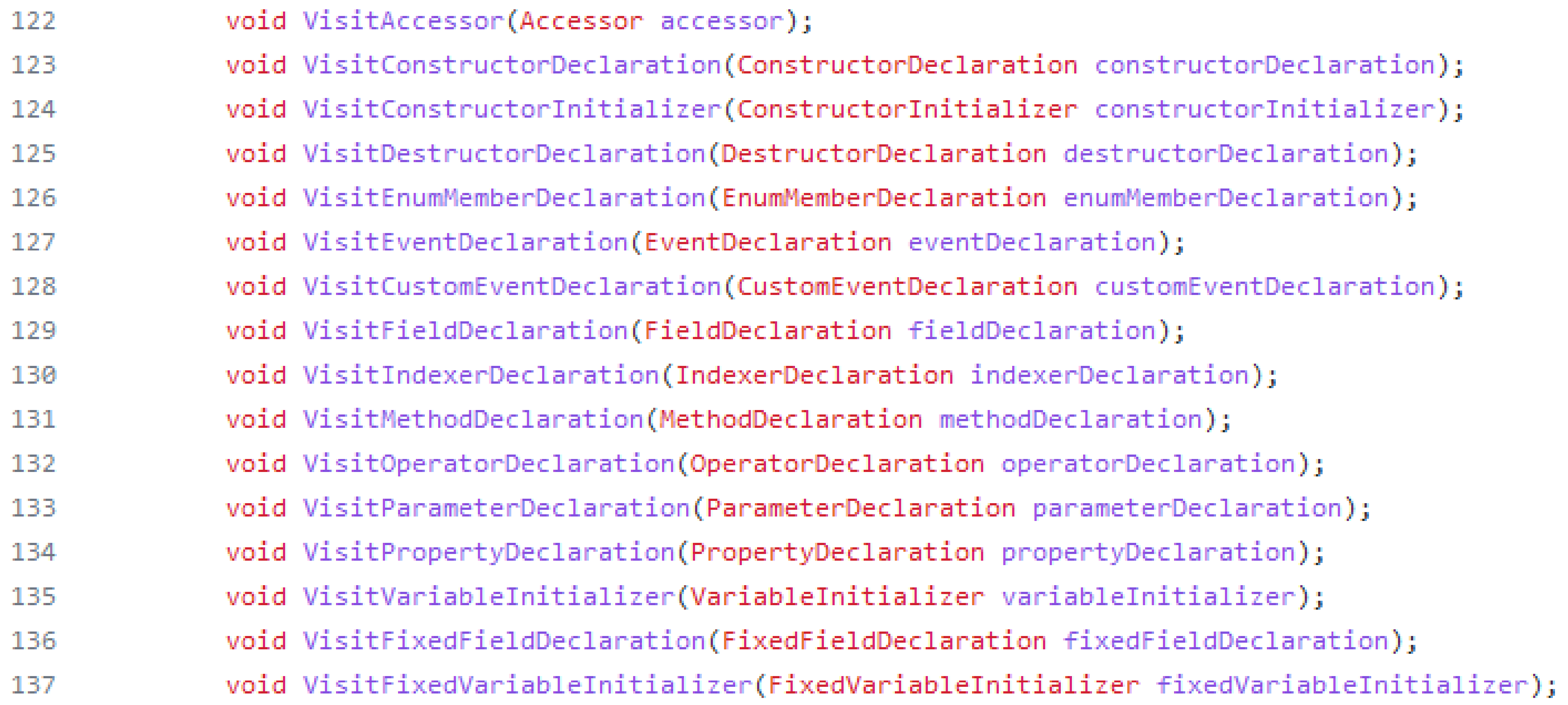
```
        void VisitAttribute(Attribute attribute);
        void VisitAttributeSection(AttributeSection attributeSection);
        void VisitDelegateDeclaration(DelegateDeclaration delegateDeclaration);
        void VisitNamespaceDeclaration(NamespaceDeclaration namespaceDeclaration);
        void VisitTypeDeclaration(TypeDeclaration typeDeclaration);
        void VisitUsingAliasDeclaration(UsingAliasDeclaration usingAliasDeclaration);
        void VisitUsingDeclaration(UsingDeclaration usingDeclaration);
        void VisitExternAliasDeclaration(ExternAliasDeclaration externAliasDeclaration);
```

ILSpy



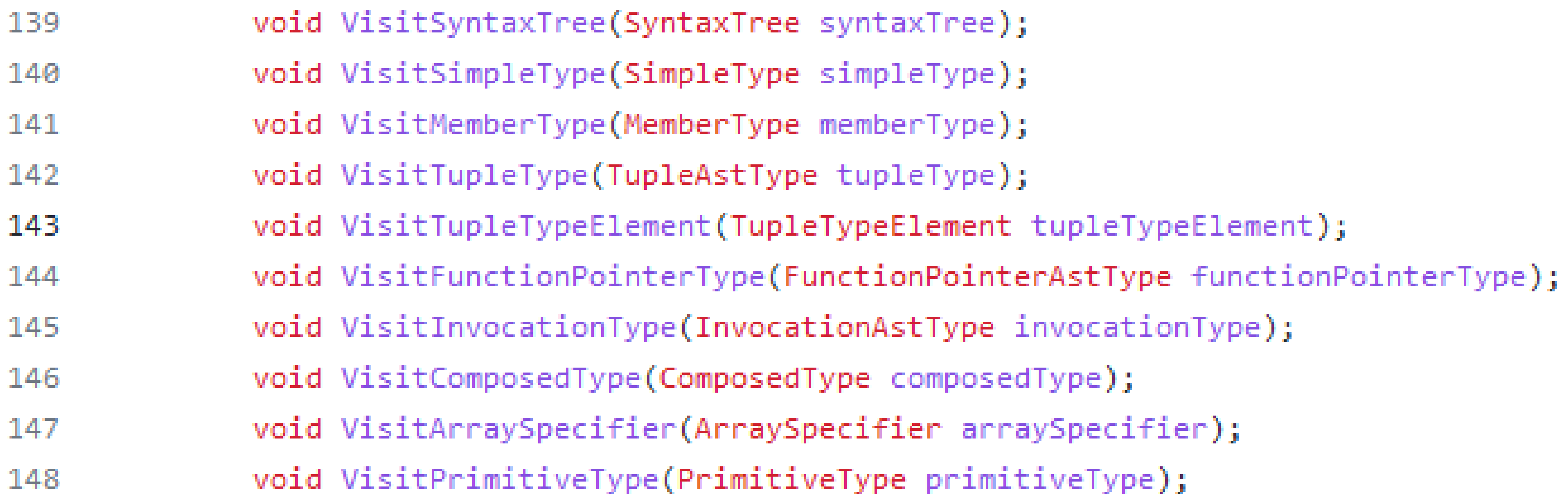
```
        void VisitBlockStatement(BlockStatement blockStatement);
        void VisitBreakStatement(BreakStatement breakStatement);
        void VisitCheckedStatement(CheckedStatement checkedStatement);
        void VisitContinueStatement(ContinueStatement continueStatement);
        void VisitDoWhileStatement(DoWhileStatement doWhileStatement);
        void VisitEmptyStatement(EmptyStatement emptyStatement);
        void VisitExpressionStatement(ExpressionStatement expressionStatement);
        void VisitFixedStatement(FixedStatement fixedStatement);
        void VisitForeachStatement(ForeachStatement foreachStatement);
        void VisitForStatement(ForStatement forStatement);
        void VisitGotoCaseStatement(GotoCaseStatement gotoCaseStatement);
        void VisitGotoDefaultStatement(GotoDefaultStatement gotoDefaultStatement);
        void VisitGotoStatement(GotoStatement gotoStatement);
        void VisitIfElseStatement(IfElseStatement ifElseStatement);
        void VisitLabelStatement(LabelStatement labelStatement);
        void VisitLockStatement(LockStatement lockStatement);
        void VisitReturnStatement(ReturnStatement returnStatement);
        void VisitSwitchStatement(SwitchStatement switchStatement);
        void VisitSwitchSection(SwitchSection switchSection);
        void VisitCaseLabel(CaseLabel caseLabel);
        void VisitSwitchExpression(SwitchExpression switchExpression);
        void VisitSwitchExpressionSection(SwitchExpressionSection switchExpressionSection);
        void VisitThrowStatement(ThrowStatement throwStatement);
        void VisitTryCatchStatement(TryCatchStatement tryCatchStatement);
        void VisitCatchClause(CatchClause catchClause);
        void VisitUncheckedStatement(UncheckedStatement uncheckedStatement);
        void VisitUnsafeStatement(UnsafeStatement unsafeStatement);
        void VisitUsingStatement(UsingStatement usingStatement);
        void VisitVariableDeclarationStatement(VariableDeclarationStatement variableDeclarationStatement);
        void VisitLocalFunctionDeclarationStatement(LocalFunctionDeclarationStatement localFunctionDeclarationStatement);
        void VisitWhileStatement(WhileStatement whileStatement);
        void VisitYieldBreakStatement(YieldBreakStatement yieldBreakStatement);
        void VisitYieldReturnStatement(YieldReturnStatement yieldReturnStatement);
```

ILSpy



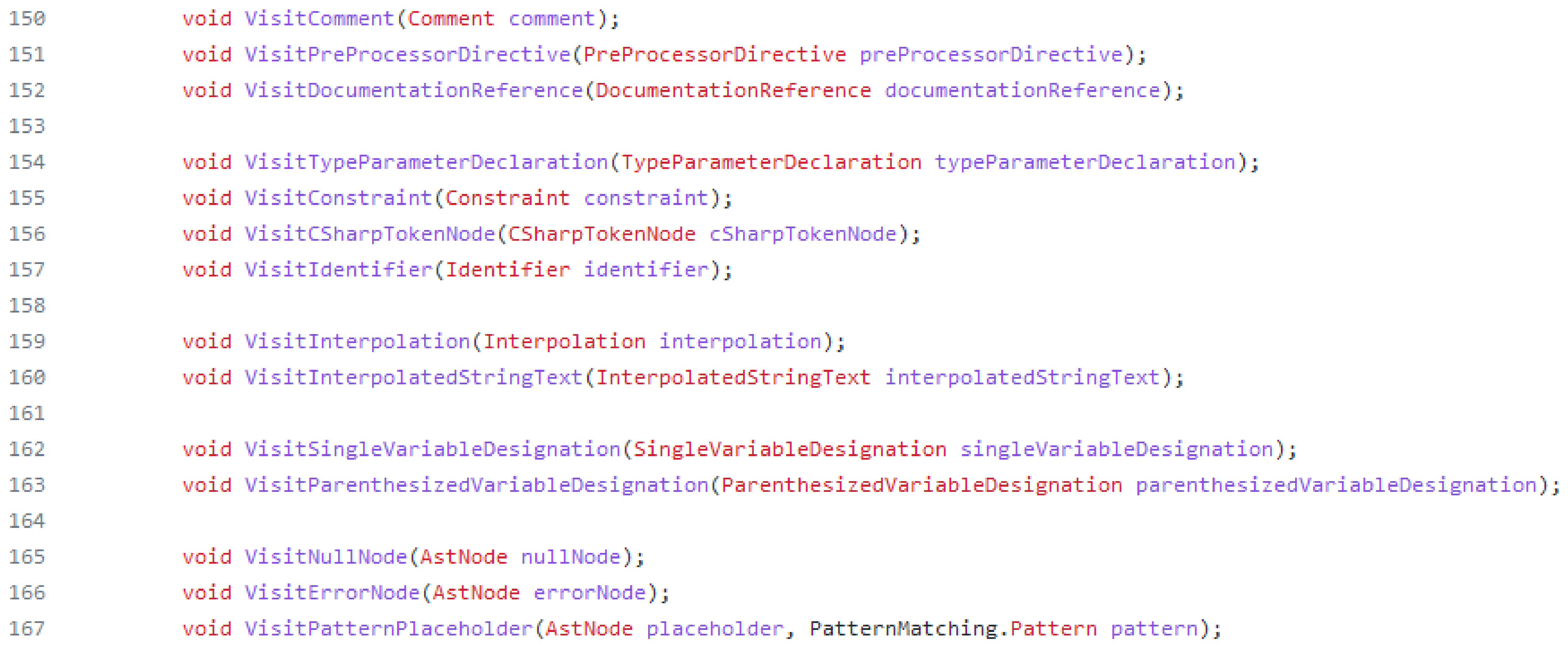
```
        void VisitAccessor(Accessor accessor);
        void VisitConstructorDeclaration(ConstructorDeclaration constructorDeclaration);
        void VisitConstructorInitializer(ConstructorInitializer constructorInitializer);
        void VisitDestructorDeclaration(DestructorDeclaration destructorDeclaration);
        void VisitEnumMemberDeclaration(EnumMemberDeclaration enumMemberDeclaration);
        void VisitEventDeclaration(EventDeclaration eventDeclaration);
        void VisitCustomEventDeclaration(CustomEventDeclaration customEventDeclaration);
        void VisitFieldDeclaration(FieldDeclaration fieldDeclaration);
        void VisitIndexerDeclaration(IndexerDeclaration indexerDeclaration);
        void VisitMethodDeclaration(MethodDeclaration methodDeclaration);
        void VisitOperatorDeclaration(OperatorDeclaration operatorDeclaration);
        void VisitParameterDeclaration(ParameterDeclaration parameterDeclaration);
        void VisitPropertyDeclaration(PropertyDeclaration propertyDeclaration);
        void VisitVariableInitializer(VariableInitializer variableInitializer);
        void VisitFixedFieldDeclaration(FixedFieldDeclaration fixedFieldDeclaration);
        void VisitFixedVariableInitializer(FixedVariableInitializer fixedVariableInitializer);
```

ILSpy



```
        void VisitSyntaxTree(SyntaxTree syntaxTree);
        void VisitSimpleType(SimpleType simpleType);
        void VisitMemberType(MemberType memberType);
        void VisitTupleType(TupleAstType tupleType);
        void VisitTupleTypeElement(TupleTypeElement tupleTypeElement);
        void VisitFunctionPointerType(FunctionPointerAstType functionPointerType);
        void VisitInvocationType(InvocationAstType invocationType);
        void VisitComposedType(ComposedType composedType);
        void VisitArraySpecifier(ArraySpecifier arraySpecifier);
        void VisitPrimitiveType(PrimitiveType primitiveType);
```

ILSpy



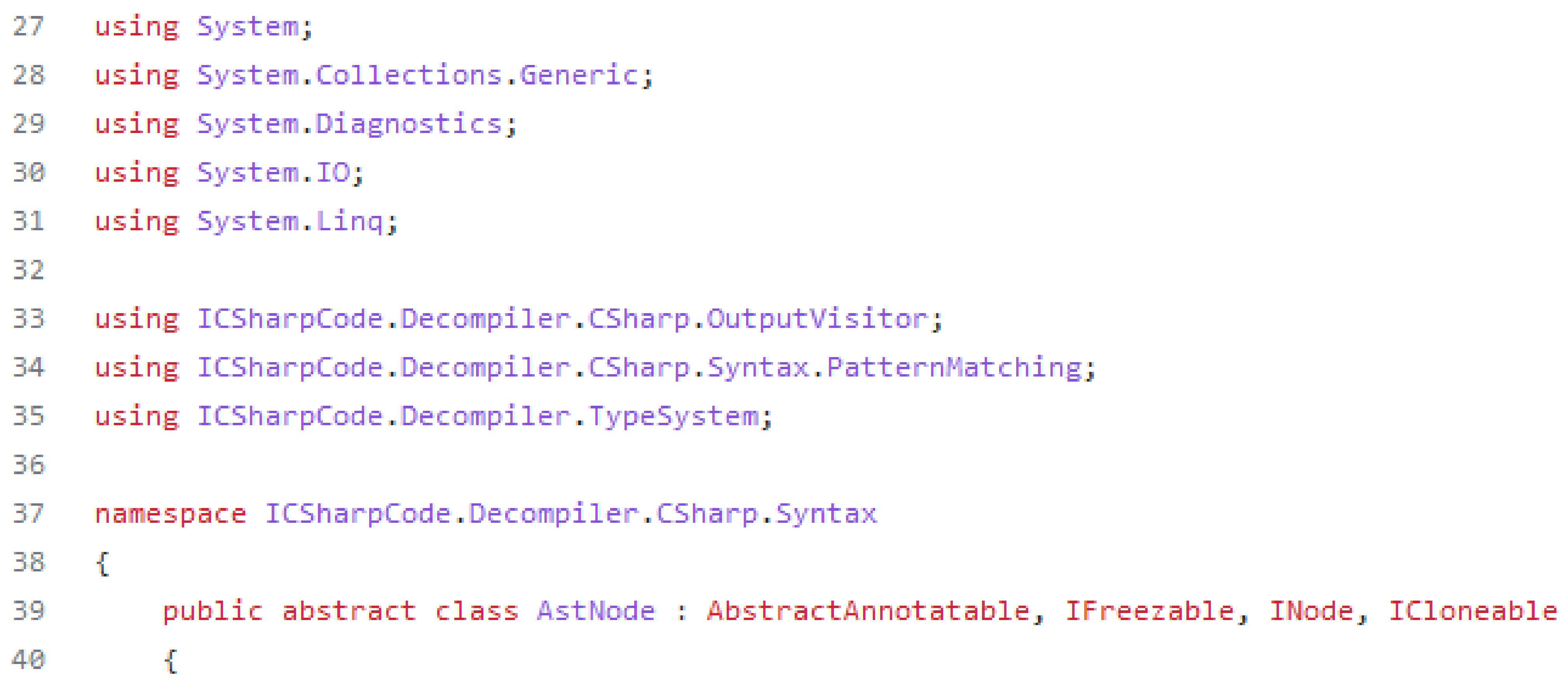
```
        void VisitComment(Comment comment);
        void VisitPreProcessorDirective(PreProcessorDirective preProcessorDirective);
        void VisitDocumentationReference(DocumentationReference documentationReference);

        void VisitTypeParameterDeclaration(TypeParameterDeclaration typeParameterDeclaration);
        void VisitConstraint(Constraint constraint);
        void VisitCSharpTokenNode(CSharpTokenNode cSharpTokenNode);
        void VisitIdentifier(Identifier identifier);

        void VisitInterpolation(Interpolation interpolation);
        void VisitInterpolatedStringText(InterpolatedStringText interpolatedStringText);

        void VisitSingleVariableDesignation(SingleVariableDesignation singleVariableDesignation);
        void VisitParenthesizedVariableDesignation(ParenthesizedVariableDesignation parenthesizedVariableDesignation);

        void VisitNullNode(AstNode nullNode);
        void VisitErrorNode(AstNode errorNode);
        void VisitPatternPlaceholder(AstNode placeholder, PatternMatching.Pattern pattern);
```

ILSpy

```
/// <summary>
/// AST visitor.
/// </summary>
public interface IAstVisitor<out S>
{
    S VisitAnonymousMethodExpression(AnonymousMethodExpression anonymousMethodExpression);
    S VisitAnonymousTypeCreateExpression(AnonymousTypeCreateExpression anonymousTypeCreateExpression);
    S VisitArrayCreateExpression(ArrayCreateExpression arrayCreateExpression);
    S VisitArrayInitializerExpression(ArrayInitializerExpression arrayInitializerExpression);
    S VisitAsExpression(AsExpression asExpression);
    S VisitAssignmentExpression(AssignmentExpression assignmentExpression);
    S VisitBaseReferenceExpression(BaseReferenceExpression baseReferenceExpression);
    S VisitBinaryOperatorExpression(BinaryOperatorExpression binaryOperatorExpression);
```

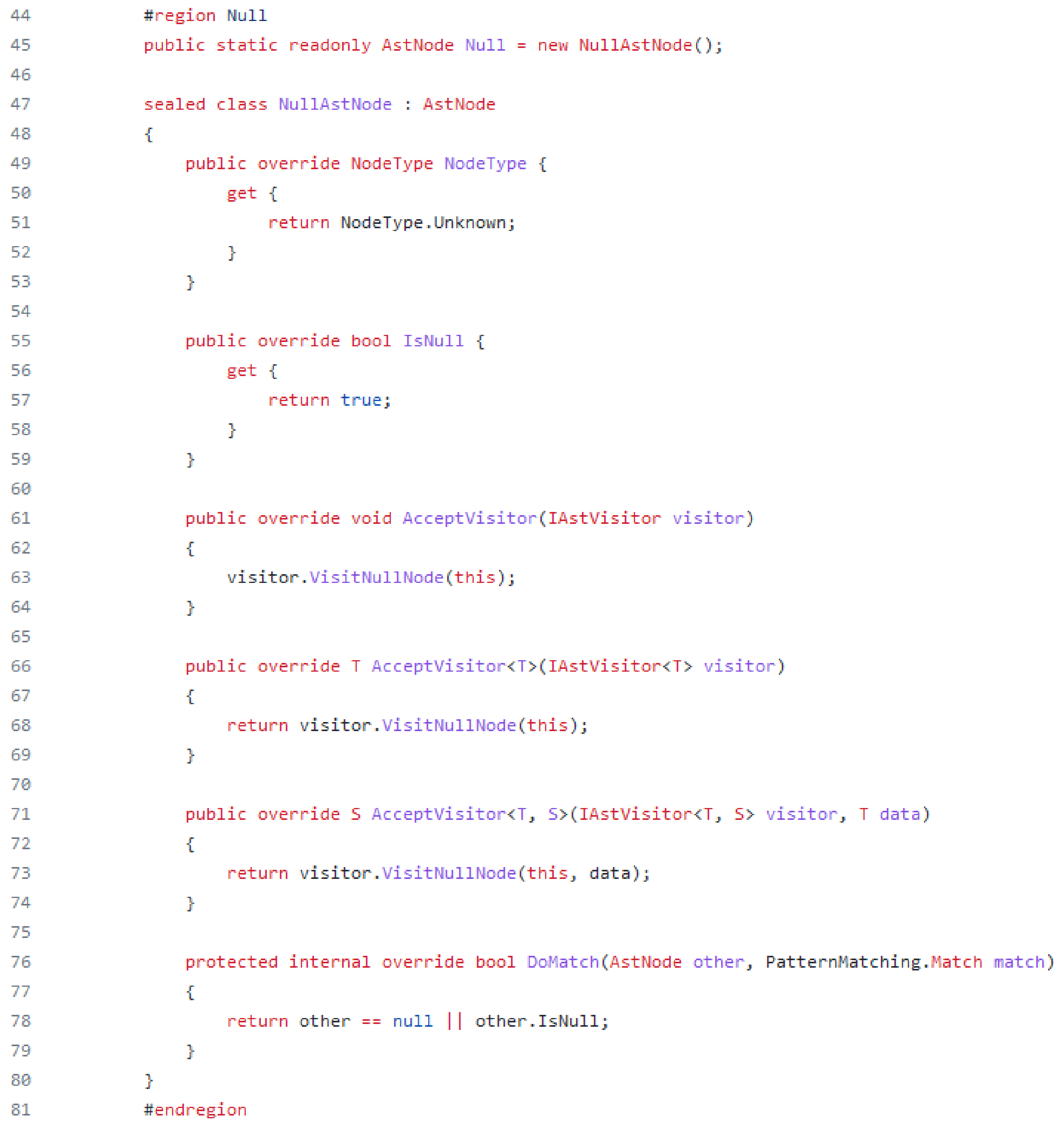
```
/// <summary>
/// AST visitor.
/// </summary>
public interface IAstVisitor<in T, out S>
{
    S VisitAnonymousMethodExpression(AnonymousMethodExpression anonymousMethodExpression, T data);
    S VisitAnonymousTypeCreateExpression(AnonymousTypeCreateExpression anonymousTypeCreateExpression, T data);
    S VisitArrayCreateExpression(ArrayCreateExpression arrayCreateExpression, T data);
    S VisitArrayInitializerExpression(ArrayInitializerExpression arrayInitializerExpression, T data);
    S VisitAsExpression(AsExpression asExpression, T data);
    S VisitAssignmentExpression(AssignmentExpression assignmentExpression, T data);
    S VisitBaseReferenceExpression(BaseReferenceExpression baseReferenceExpression, T data);
    S VisitBinaryOperatorExpression(BinaryOperatorExpression binaryOperatorExpression, T data);
```

ILSpy



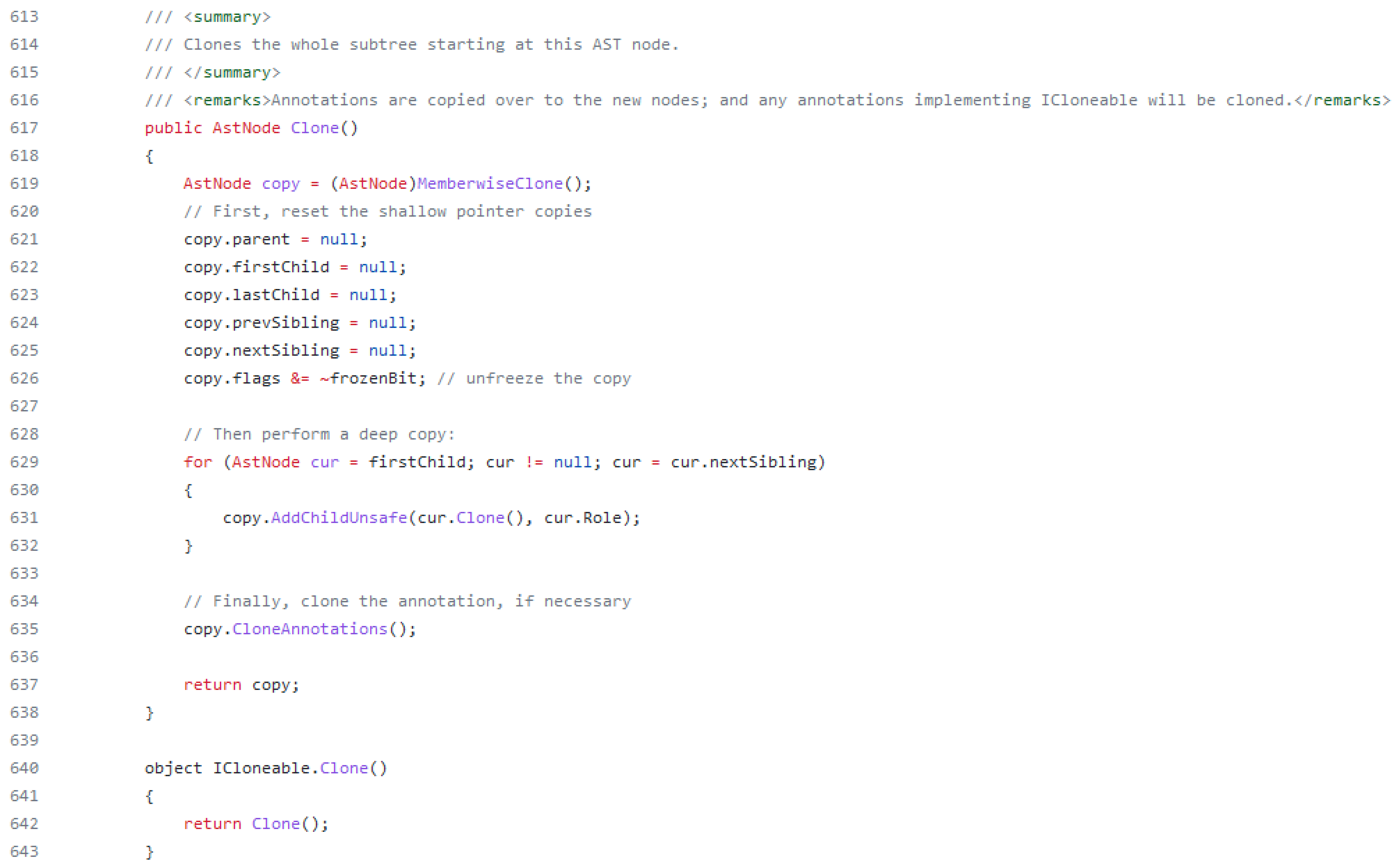
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;

using ICSharpCode.Decompiler.CSharp.OutputVisitor;
using ICSharpCode.Decompiler.CSharp.Syntax.PatternMatching;
using ICSharpCode.Decompiler.TypeSystem;

namespace ICSharpCode.Decompiler.CSharp.Syntax
{
    public abstract class AstNode : AbstractAnnotatable, IFreezable, INode, ICloneable
    {
```

ILSpy



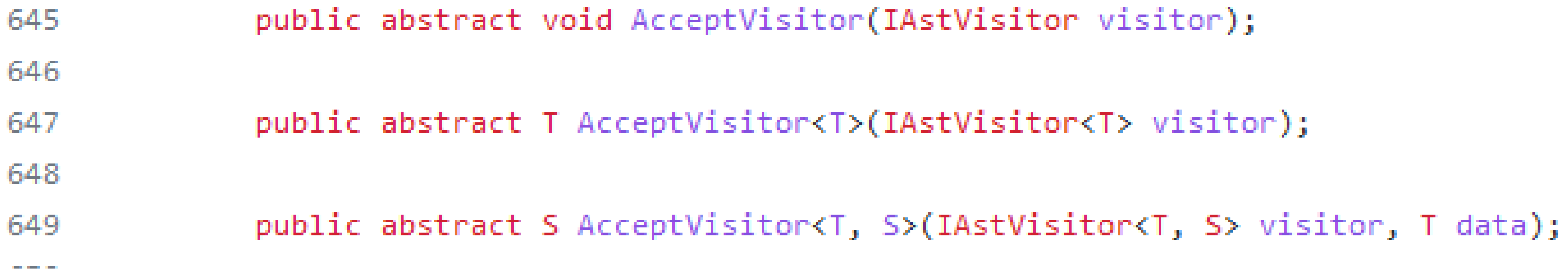
```
        #region Null
        public static readonly AstNode Null = new NullAstNode();

        sealed class NullAstNode : AstNode
        {
            public override NodeType NodeType {
                get {
                    return NodeType.Unknown;
                }
            }

            public override bool IsNull {
                get {
                    return true;
                }
            }

            public override void AcceptVisitor(IAstVisitor visitor)
            {
                visitor.VisitNullNode(this);
            }

            public override T AcceptVisitor<T>(IAstVisitor<T> visitor)
            {
                return visitor.VisitNullNode(this);
            }

            public override S AcceptVisitor<T, S>(IAstVisitor<T, S> visitor, T data)
            {
                return visitor.VisitNullNode(this, data);
            }

            protected internal override bool DoMatch(AstNode other, PatternMatching.Match match)
            {
                return other == null || other.IsNull;
            }
        }
        #endregion
```

ILSpy



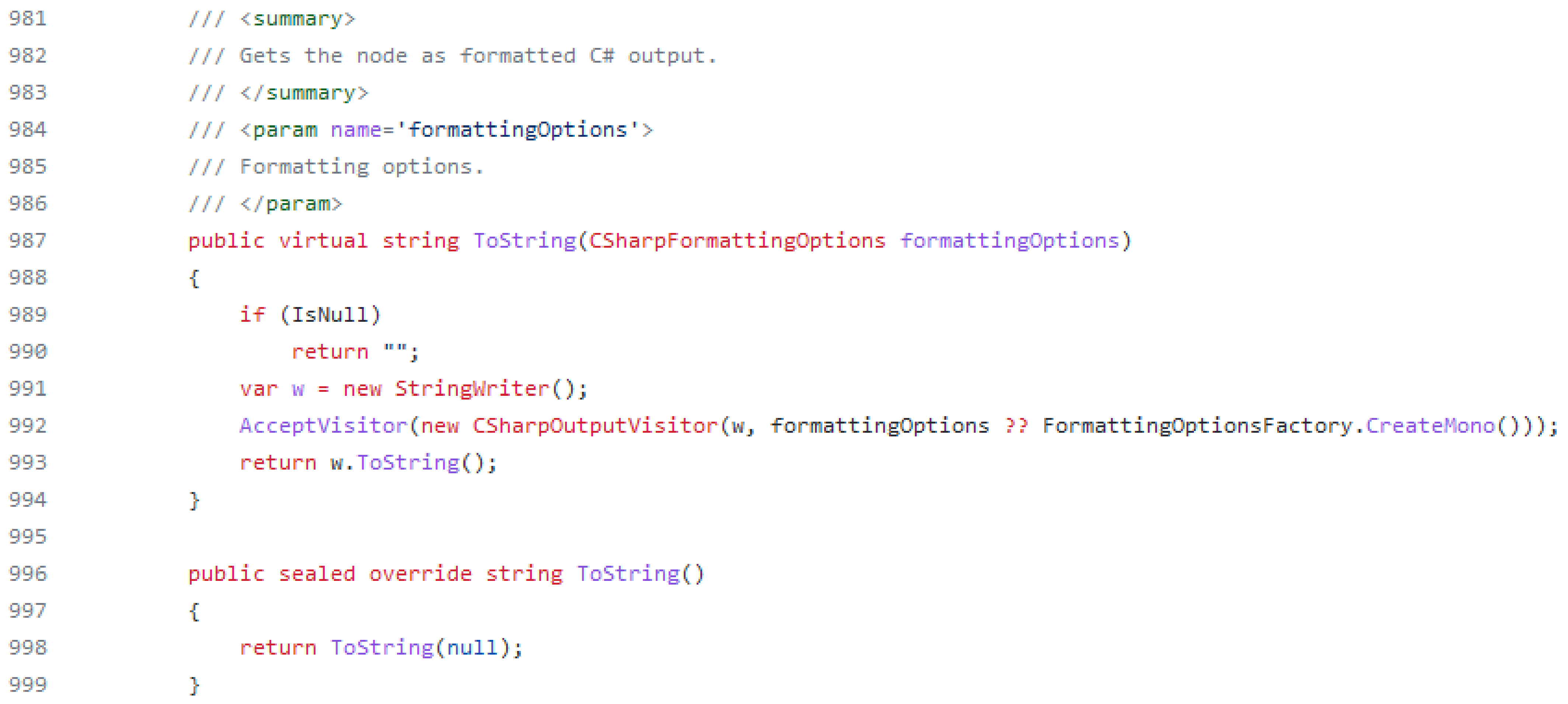
```csharp
/// <summary>
/// Clones the whole subtree starting at this AST node.
/// </summary>
/// <remarks>Annotations are copied over to the new nodes; and any annotations implementing ICloneable will be cloned.</remarks>
public AstNode Clone()
{
    AstNode copy = (AstNode)MemberwiseClone();
    // First, reset the shallow pointer copies
    copy.parent = null;
    copy.firstChild = null;
    copy.lastChild = null;
    copy.prevSibling = null;
    copy.nextSibling = null;
    copy.flags &= ~frozenBit; // unfreeze the copy

    // Then perform a deep copy:
    for (AstNode cur = firstChild; cur != null; cur = cur.nextSibling)
    {
        copy.AddChildUnsafe(cur.Clone(), cur.Role);
    }

    // Finally, clone the annotation, if necessary
    copy.CloneAnnotations();

    return copy;
}

object ICloneable.Clone()
{
    return Clone();
}
```

ILSpy



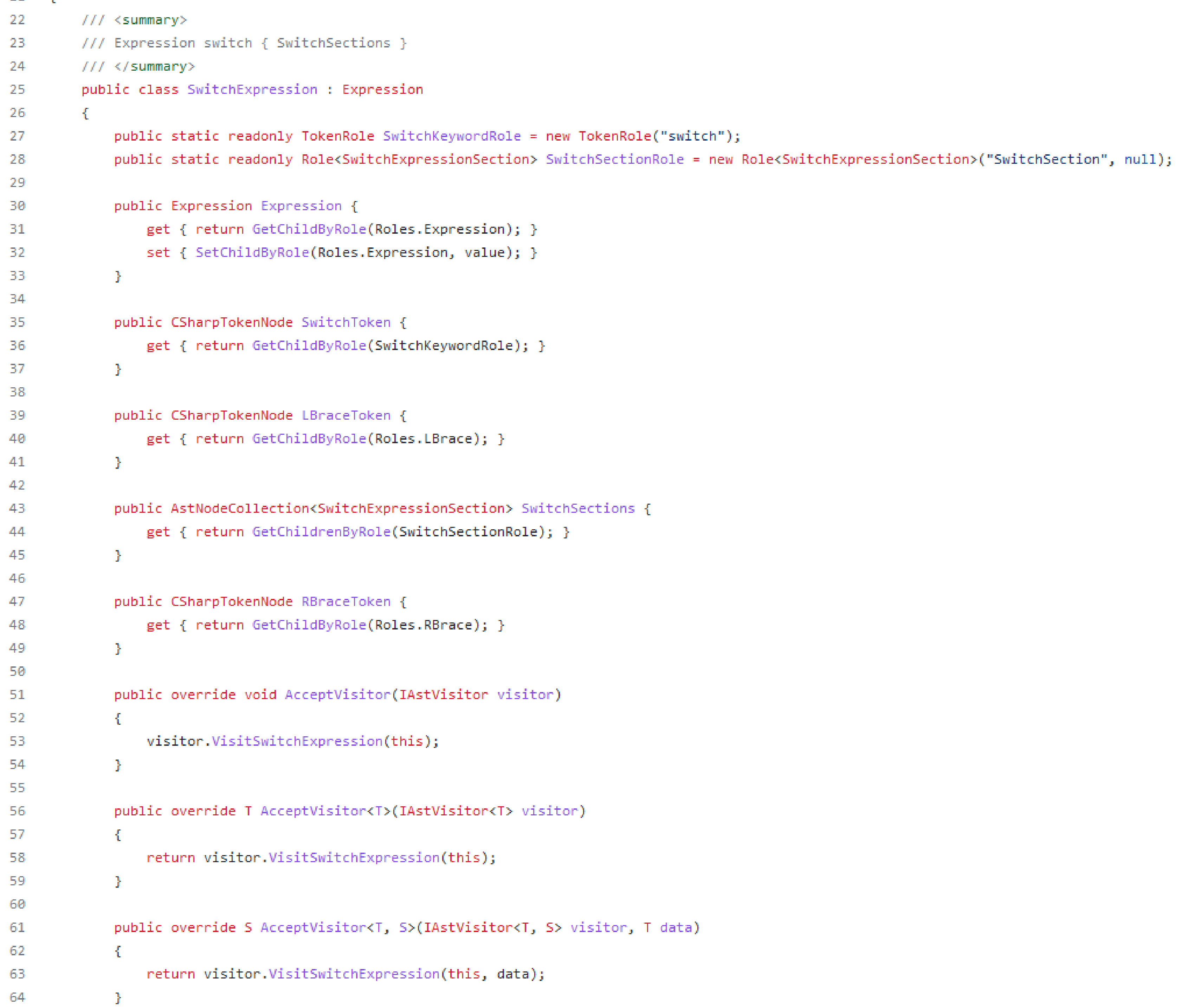
```
        public abstract void AcceptVisitor(IAstVisitor visitor);

        public abstract T AcceptVisitor<T>(IAstVisitor<T> visitor);

        public abstract S AcceptVisitor<T, S>(IAstVisitor<T, S> visitor, T data);
```

ILSpy



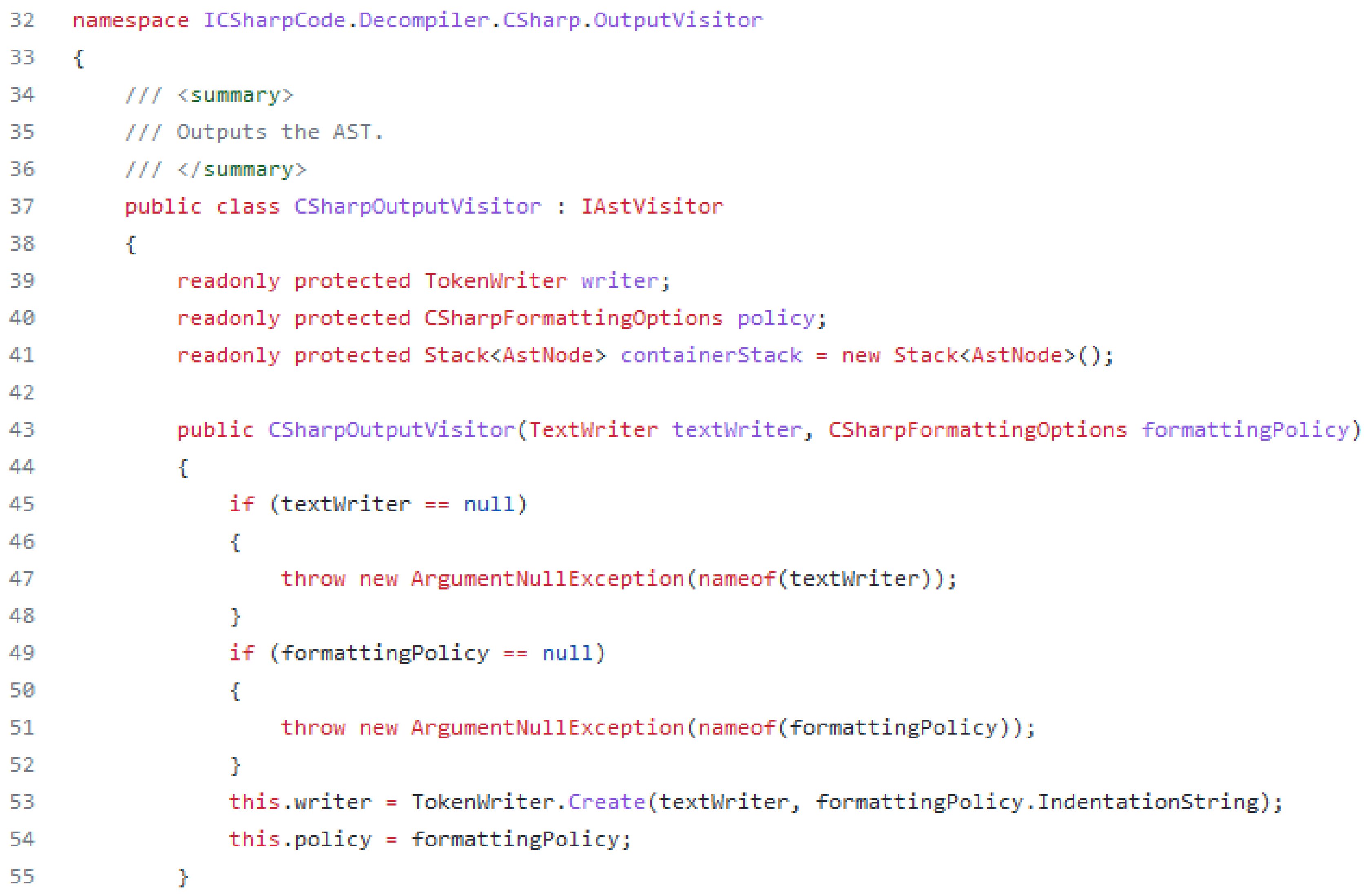
```
/// <summary>
/// Gets the node as formatted C# output.
/// </summary>
/// <param name='formattingOptions'>
/// Formatting options.
/// </param>
public virtual string ToString(CSharpFormattingOptions formattingOptions)
{
    if (IsNull)
        return "";
    var w = new StringWriter();
    AcceptVisitor(new CSharpOutputVisitor(w, formattingOptions ?? FormattingOptionsFactory.CreateMono()));
    return w.ToString();
}

public sealed override string ToString()
{
    return ToString(null);
}
```

ILSpy



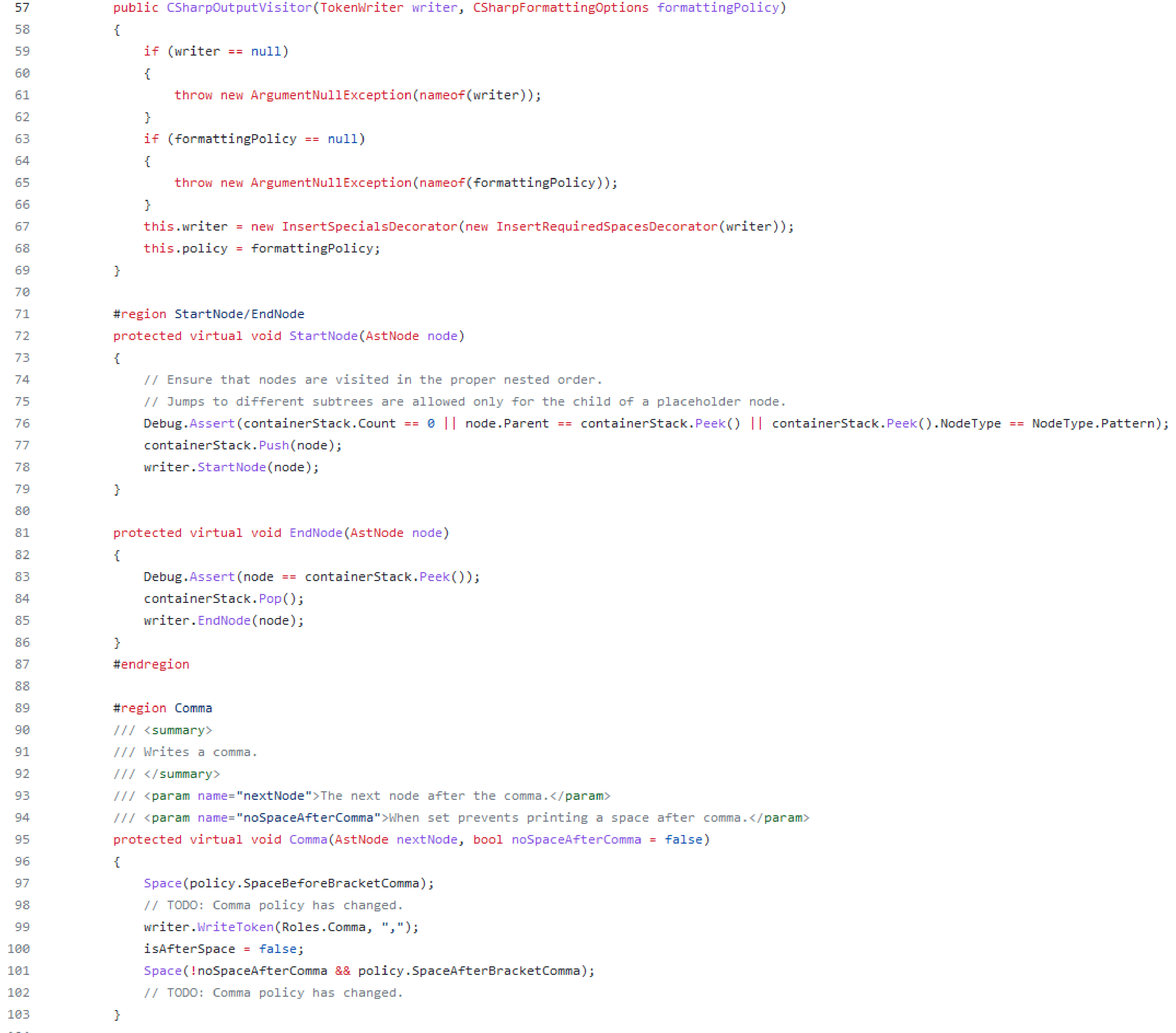
```
{
    /// <summary>
    /// Expression switch { SwitchSections }
    /// </summary>
    public class SwitchExpression : Expression
    {
        public static readonly TokenRole SwitchKeywordRole = new TokenRole("switch");
        public static readonly Role<SwitchExpressionSection> SwitchSectionRole = new Role<SwitchExpressionSection>("SwitchSection", null);

        public Expression Expression {
            get { return GetChildByRole(Roles.Expression); }
            set { SetChildByRole(Roles.Expression, value); }
        }

        public CSharpTokenNode SwitchToken {
            get { return GetChildByRole(SwitchKeywordRole); }
        }

        public CSharpTokenNode LBraceToken {
            get { return GetChildByRole(Roles.LBrace); }
        }

        public AstNodeCollection<SwitchExpressionSection> SwitchSections {
            get { return GetChildrenByRole(SwitchSectionRole); }
        }

        public CSharpTokenNode RBraceToken {
            get { return GetChildByRole(Roles.RBrace); }
        }

        public override void AcceptVisitor(IAstVisitor visitor)
        {
            visitor.VisitSwitchExpression(this);
        }

        public override T AcceptVisitor<T>(IAstVisitor<T> visitor)
        {
            return visitor.VisitSwitchExpression(this);
        }

        public override S AcceptVisitor<T, S>(IAstVisitor<T, S> visitor, T data)
        {
            return visitor.VisitSwitchExpression(this, data);
        }
```

ILSpy



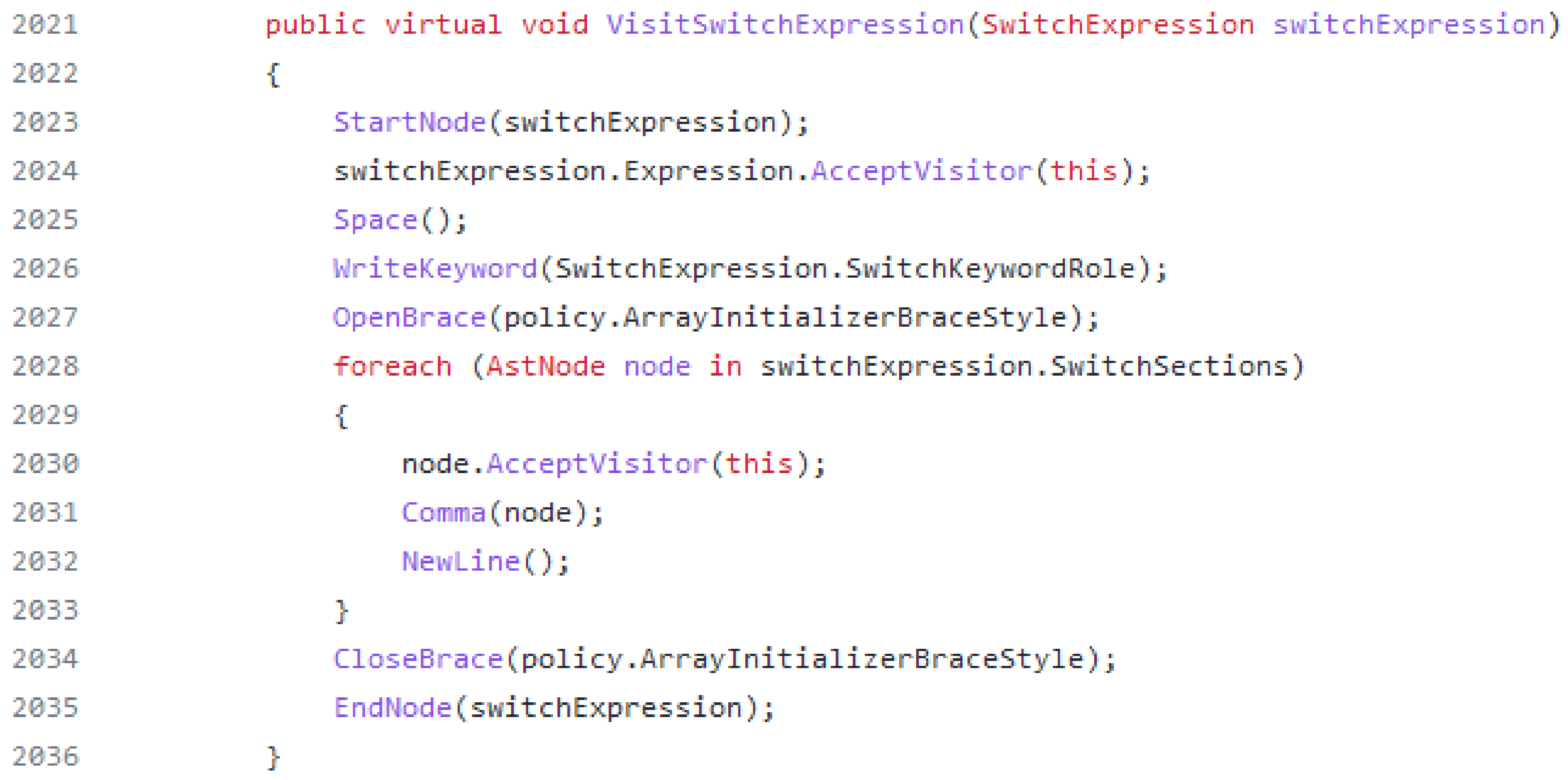
```
namespace ICSharpCode.Decompiler.CSharp.OutputVisitor
{
    /// <summary>
    /// Outputs the AST.
    /// </summary>
    public class CSharpOutputVisitor : IAstVisitor
    {
        readonly protected TokenWriter writer;
        readonly protected CSharpFormattingOptions policy;
        readonly protected Stack<AstNode> containerStack = new Stack<AstNode>();

        public CSharpOutputVisitor(TextWriter textWriter, CSharpFormattingOptions formattingPolicy)
        {
            if (textWriter == null)
            {
                throw new ArgumentNullException(nameof(textWriter));
            }
            if (formattingPolicy == null)
            {
                throw new ArgumentNullException(nameof(formattingPolicy));
            }
            this.writer = TokenWriter.Create(textWriter, formattingPolicy.IndentationString);
            this.policy = formattingPolicy;
        }
```

ILSpy



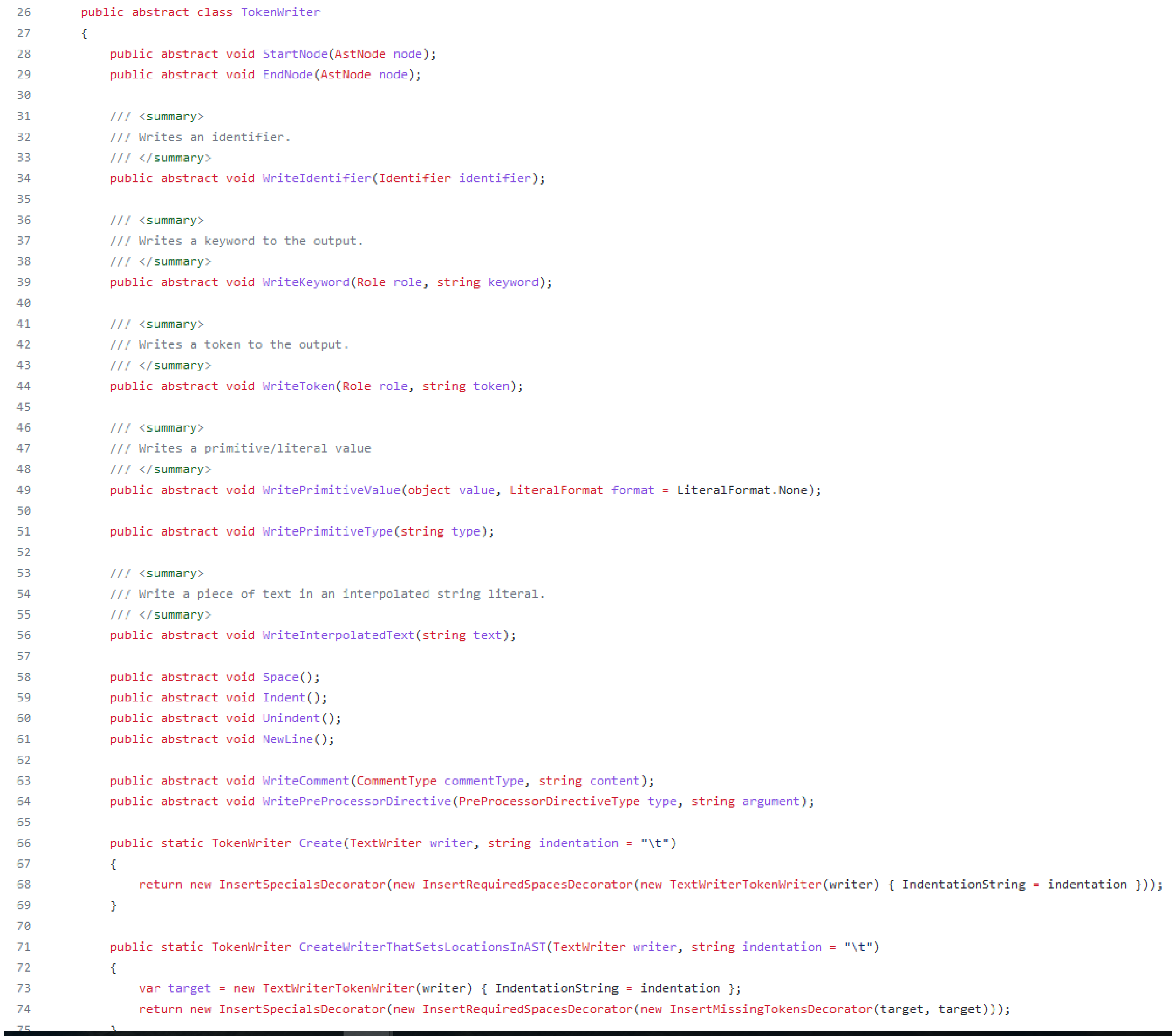
```csharp
public CSharpOutputVisitor(TokenWriter writer, CSharpFormattingOptions formattingPolicy)
{
    if (writer == null)
    {
        throw new ArgumentNullException(nameof(writer));
    }
    if (formattingPolicy == null)
    {
        throw new ArgumentNullException(nameof(formattingPolicy));
    }
    this.writer = new InsertSpecialsDecorator(new InsertRequiredSpacesDecorator(writer));
    this.policy = formattingPolicy;
}

#region StartNode/EndNode
protected virtual void StartNode(AstNode node)
{
    // Ensure that nodes are visited in the proper nested order.
    // Jumps to different subtrees are allowed only for the child of a placeholder node.
    Debug.Assert(containerStack.Count == 0 || node.Parent == containerStack.Peek() || containerStack.Peek().NodeType == NodeType.Pattern);
    containerStack.Push(node);
    writer.StartNode(node);
}

protected virtual void EndNode(AstNode node)
{
    Debug.Assert(node == containerStack.Peek());
    containerStack.Pop();
    writer.EndNode(node);
}
#endregion

#region Comma
/// <summary>
/// Writes a comma.
/// </summary>
/// <param name="nextNode">The next node after the comma.</param>
/// <param name="noSpaceAfterComma">When set prevents printing a space after comma.</param>
protected virtual void Comma(AstNode nextNode, bool noSpaceAfterComma = false)
{
    Space(policy.SpaceBeforeBracketComma);
    // TODO: Comma policy has changed.
    writer.WriteToken(Roles.Comma, ",");
    isAfterSpace = false;
    Space(!noSpaceAfterComma && policy.SpaceAfterBracketComma);
    // TODO: Comma policy has changed.
}
```

ILSpy



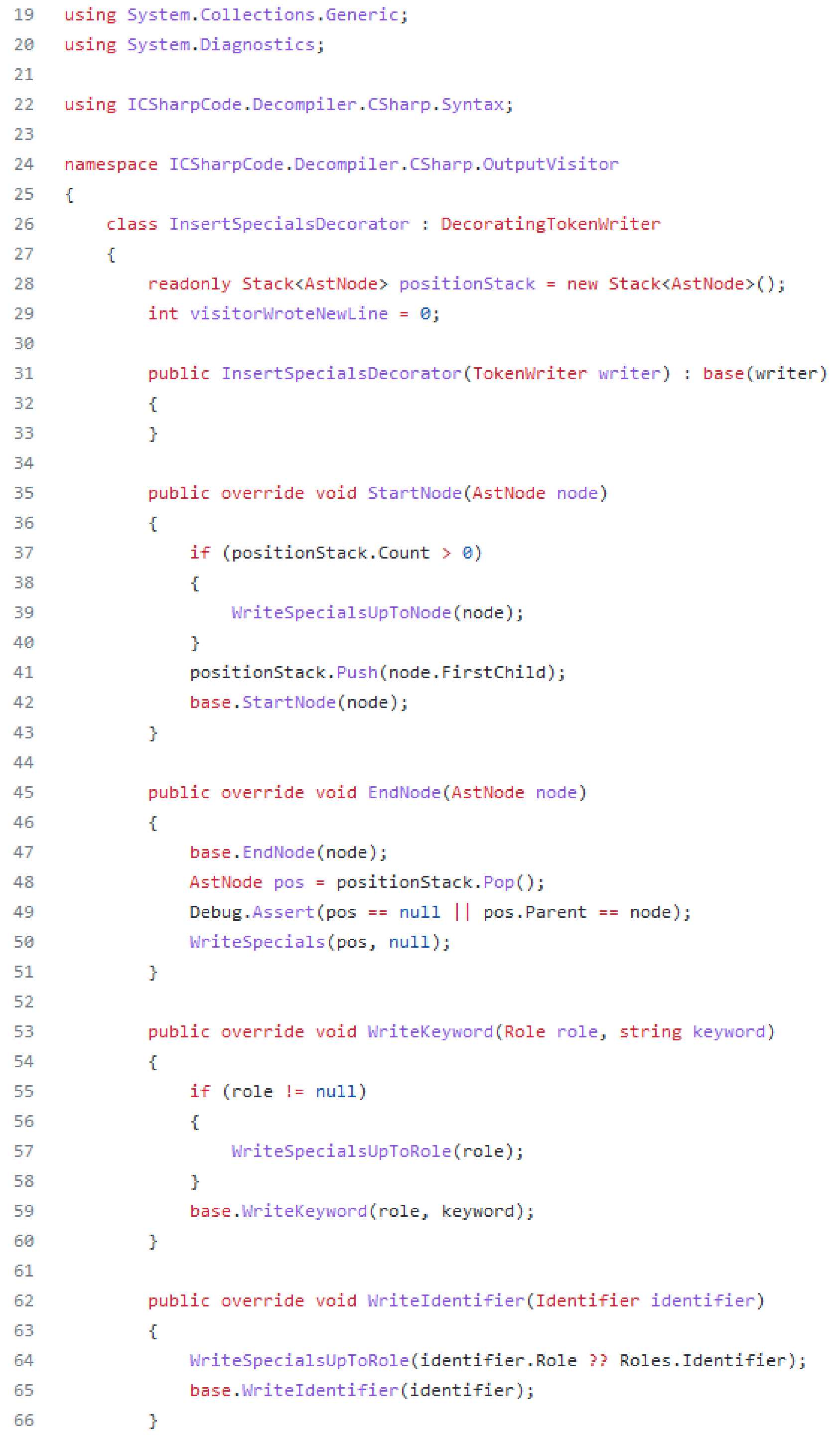
```
public virtual void VisitSwitchExpression(SwitchExpression switchExpression)
{
    StartNode(switchExpression);
    switchExpression.Expression.AcceptVisitor(this);
    Space();
    WriteKeyword(SwitchExpression.SwitchKeywordRole);
    OpenBrace(policy.ArrayInitializerBraceStyle);
    foreach (AstNode node in switchExpression.SwitchSections)
    {
        node.AcceptVisitor(this);
        Comma(node);
        NewLine();
    }
    CloseBrace(policy.ArrayInitializerBraceStyle);
    EndNode(switchExpression);
}
```

ILSpy



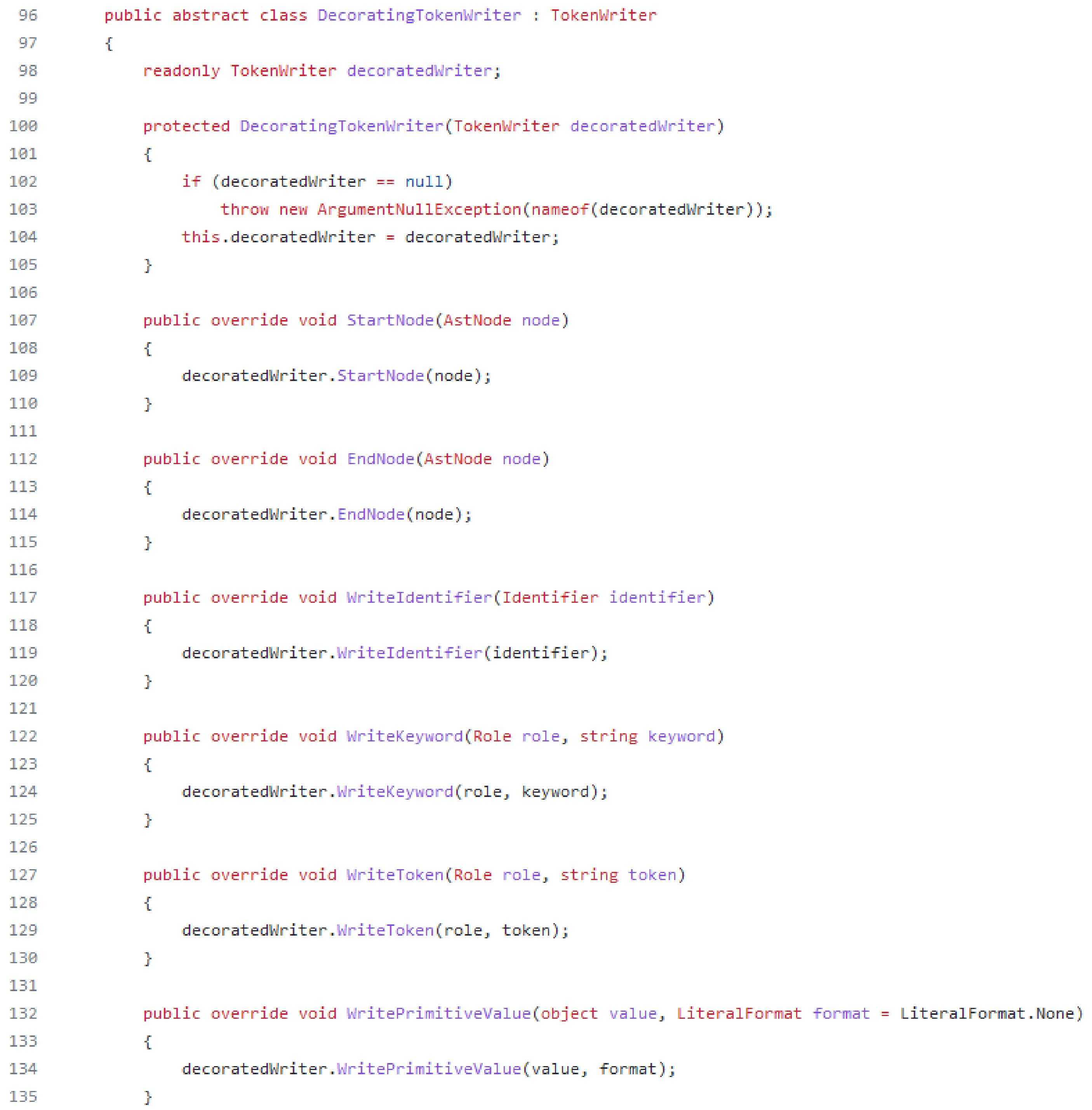
```
public abstract class TokenWriter
{
    public abstract void StartNode(AstNode node);
    public abstract void EndNode(AstNode node);

    /// <summary>
    /// Writes an identifier.
    /// </summary>
    public abstract void WriteIdentifier(Identifier identifier);

    /// <summary>
    /// Writes a keyword to the output.
    /// </summary>
    public abstract void WriteKeyword(Role role, string keyword);

    /// <summary>
    /// Writes a token to the output.
    /// </summary>
    public abstract void WriteToken(Role role, string token);

    /// <summary>
    /// Writes a primitive/literal value
    /// </summary>
    public abstract void WritePrimitiveValue(object value, LiteralFormat format = LiteralFormat.None);

    public abstract void WritePrimitiveType(string type);

    /// <summary>
    /// Write a piece of text in an interpolated string literal.
    /// </summary>
    public abstract void WriteInterpolatedText(string text);

    public abstract void Space();
    public abstract void Indent();
    public abstract void Unindent();
    public abstract void NewLine();

    public abstract void WriteComment(CommentType commentType, string content);
    public abstract void WritePreProcessorDirective(PreProcessorDirectiveType type, string argument);

    public static TokenWriter Create(TextWriter writer, string indentation = "\t")
    {
        return new InsertSpecialsDecorator(new InsertRequiredSpacesDecorator(new TextWriterTokenWriter(writer) { IndentationString = indentation }));
    }

    public static TokenWriter CreateWriterThatSetsLocationsInAST(TextWriter writer, string indentation = "\t")
    {
        var target = new TextWriterTokenWriter(writer) { IndentationString = indentation };
        return new InsertSpecialsDecorator(new InsertRequiredSpacesDecorator(new InsertMissingTokensDecorator(target, target)));
```

ILSpy



```
using System.Collections.Generic;
using System.Diagnostics;

using ICSharpCode.Decompiler.CSharp.Syntax;

namespace ICSharpCode.Decompiler.CSharp.OutputVisitor
{
    class InsertSpecialsDecorator : DecoratingTokenWriter
    {
        readonly Stack<AstNode> positionStack = new Stack<AstNode>();
        int visitorWroteNewLine = 0;

        public InsertSpecialsDecorator(TokenWriter writer) : base(writer)
        {
        }

        public override void StartNode(AstNode node)
        {
            if (positionStack.Count > 0)
            {
                WriteSpecialsUpToNode(node);
            }
            positionStack.Push(node.FirstChild);
            base.StartNode(node);
        }

        public override void EndNode(AstNode node)
        {
            base.EndNode(node);
            AstNode pos = positionStack.Pop();
            Debug.Assert(pos == null || pos.Parent == node);
            WriteSpecials(pos, null);
        }

        public override void WriteKeyword(Role role, string keyword)
        {
            if (role != null)
            {
                WriteSpecialsUpToRole(role);
            }
            base.WriteKeyword(role, keyword);
        }

        public override void WriteIdentifier(Identifier identifier)
        {
            WriteSpecialsUpToRole(identifier.Role ?? Roles.Identifier);
            base.WriteIdentifier(identifier);
        }
```

ILSpy



```
public abstract class DecoratingTokenWriter : TokenWriter
{
    readonly TokenWriter decoratedWriter;

    protected DecoratingTokenWriter(TokenWriter decoratedWriter)
    {
        if (decoratedWriter == null)
            throw new ArgumentNullException(nameof(decoratedWriter));
        this.decoratedWriter = decoratedWriter;
    }

    public override void StartNode(AstNode node)
    {
        decoratedWriter.StartNode(node);
    }

    public override void EndNode(AstNode node)
    {
        decoratedWriter.EndNode(node);
    }

    public override void WriteIdentifier(Identifier identifier)
    {
        decoratedWriter.WriteIdentifier(identifier);
    }

    public override void WriteKeyword(Role role, string keyword)
    {
        decoratedWriter.WriteKeyword(role, keyword);
    }

    public override void WriteToken(Role role, string token)
    {
        decoratedWriter.WriteToken(role, token);
    }

    public override void WritePrimitiveValue(object value, LiteralFormat format = LiteralFormat.None)
    {
        decoratedWriter.WritePrimitiveValue(value, format);
    }
```

Спасибо за внимание!