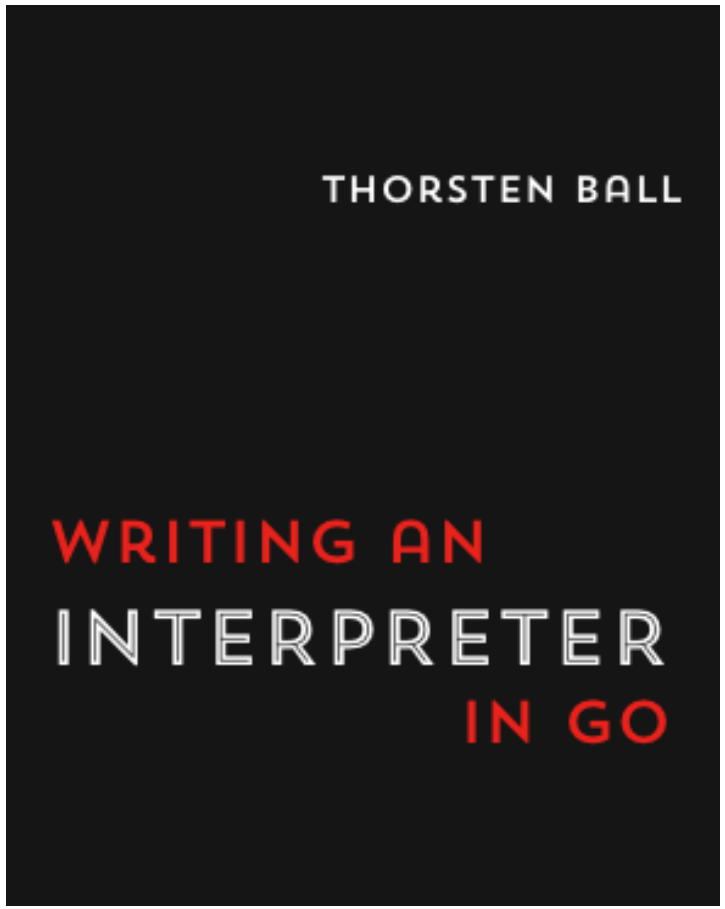


interpreterbook.com

Writing an Interpreter in ~~GO~~ C#

Tim Boyle

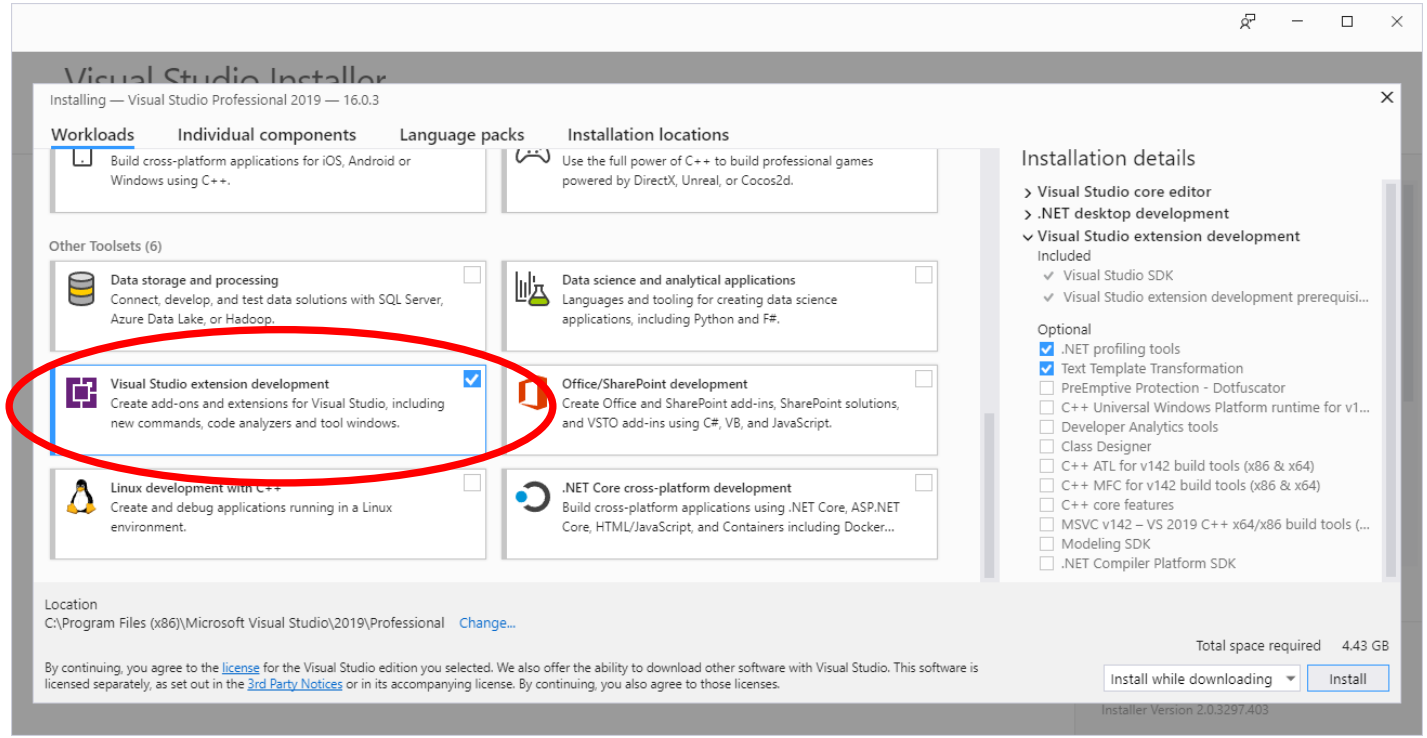
.Net Oxford

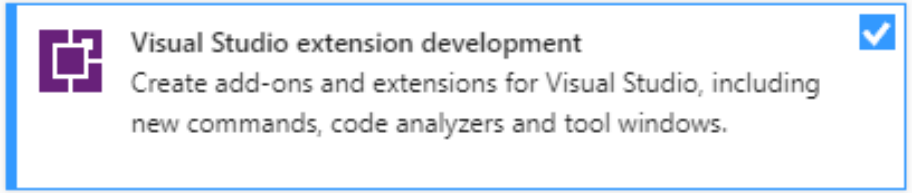
14th May 2019

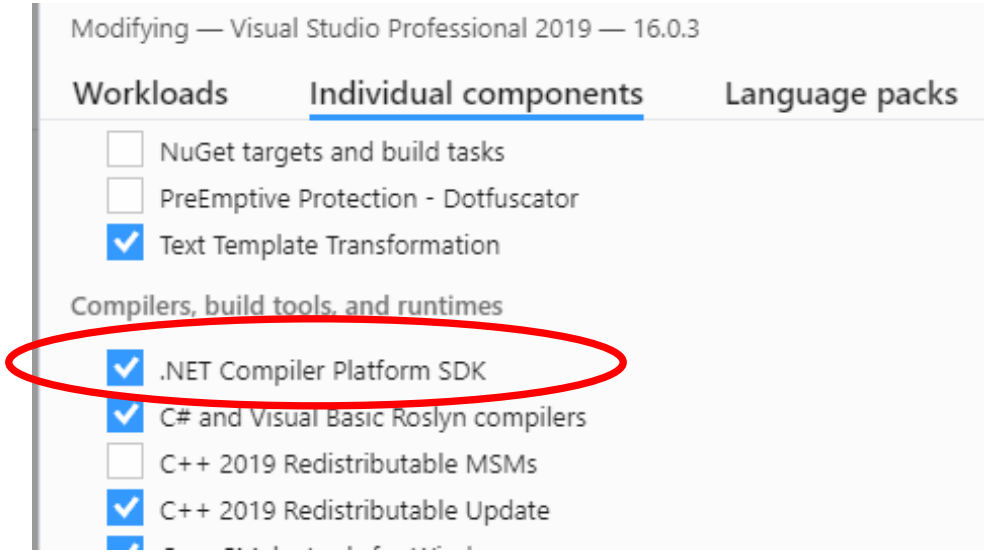
Overview

- Roslyn API Template
- Syntax Visualizer
- Simple Interpreter
- Code Fixes

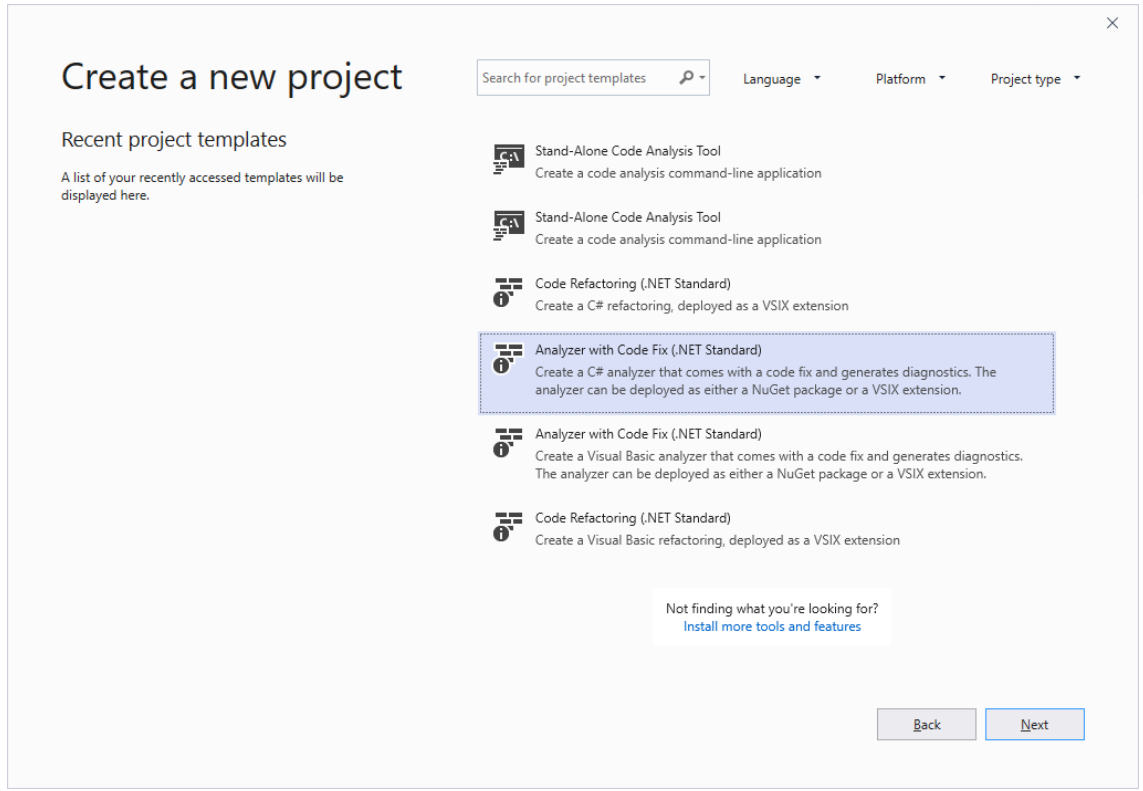
Roslyn SDK

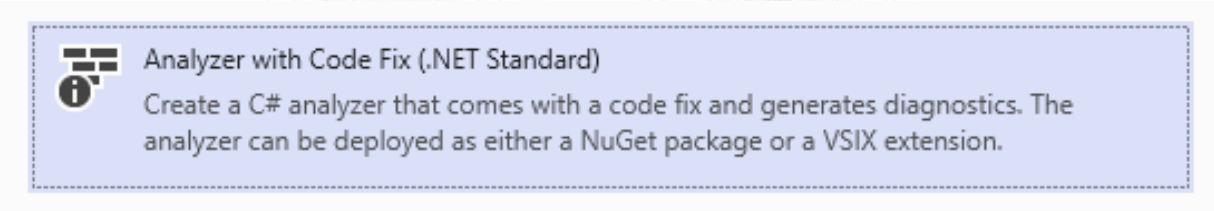

Visual Studio Installer
Installing — Visual Studio Professional 2019 — 16.0.3
Workloads
Individual components
Language packs
Installation locations
Build cross-platform applications for iOS, Android or Windows using C++.
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.
Other Toolsets (6)
Data storage and processing
Connect, develop, and test data solutions with SQL Server, Azure Data Lake, or Hadoop.
Data science and analytical applications
Languages and tooling for creating data science applications, including Python and F#.
Visual Studio extension development
Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.
Office/SharePoint development
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.
Linux development with C++
Create and debug applications running in a Linux environment.
.NET Core cross-platform development
Build cross-platform applications using .NET Core, ASP.NET Core, HTML/JavaScript, and Containers including Docker...
Installation details
Visual Studio core editor
.NET desktop development
Visual Studio extension development
Included
Visual Studio SDK
Visual Studio extension development prerequisi...
Optional
.NET profiling tools
Text Template Transformation
PreEmptive Protection - Dotfuscator
C++ Universal Windows Platform runtime for v1...
Developer Analytics tools
Class Designer
C++ ATL for v142 build tools (x86 & x64)
C++ MFC for v142 build tools (x86 & x64)
C++ core features
MSVC v142 – VS 2019 C++ x64/x86 build tools (...
Modeling SDK
.NET Compiler Platform SDK
Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional
Change...
Total space required 4.43 GB
By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.
Install while downloading
Install
Installer Version 2.0.3297.403

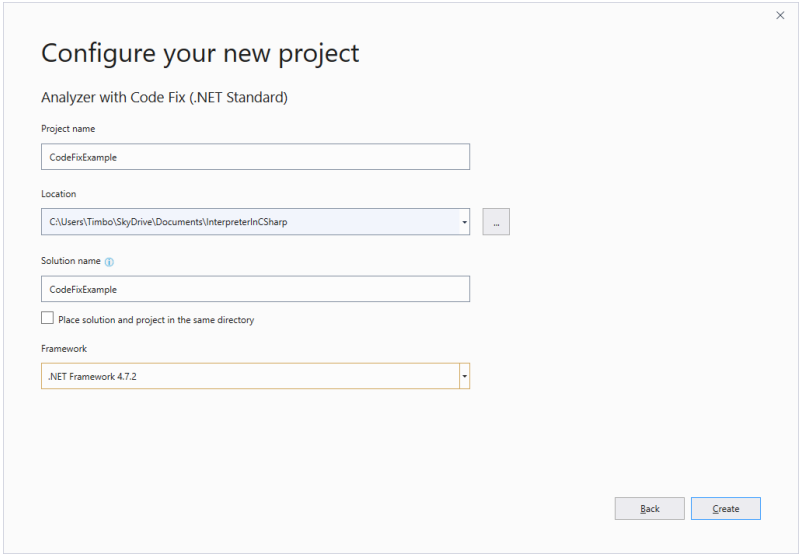

Visual Studio extension development
Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.


Modifying — Visual Studio Professional 2019 — 16.0.3
Workloads
Individual components
Language packs
NuGet targets and build tasks
PreEmptive Protection - Dotfuscator
Text Template Transformation
Compilers, build tools, and runtimes
.NET Compiler Platform SDK
C# and Visual Basic Roslyn compilers
C++ 2019 Redistributable MSMs
C++ 2019 Redistributable Update

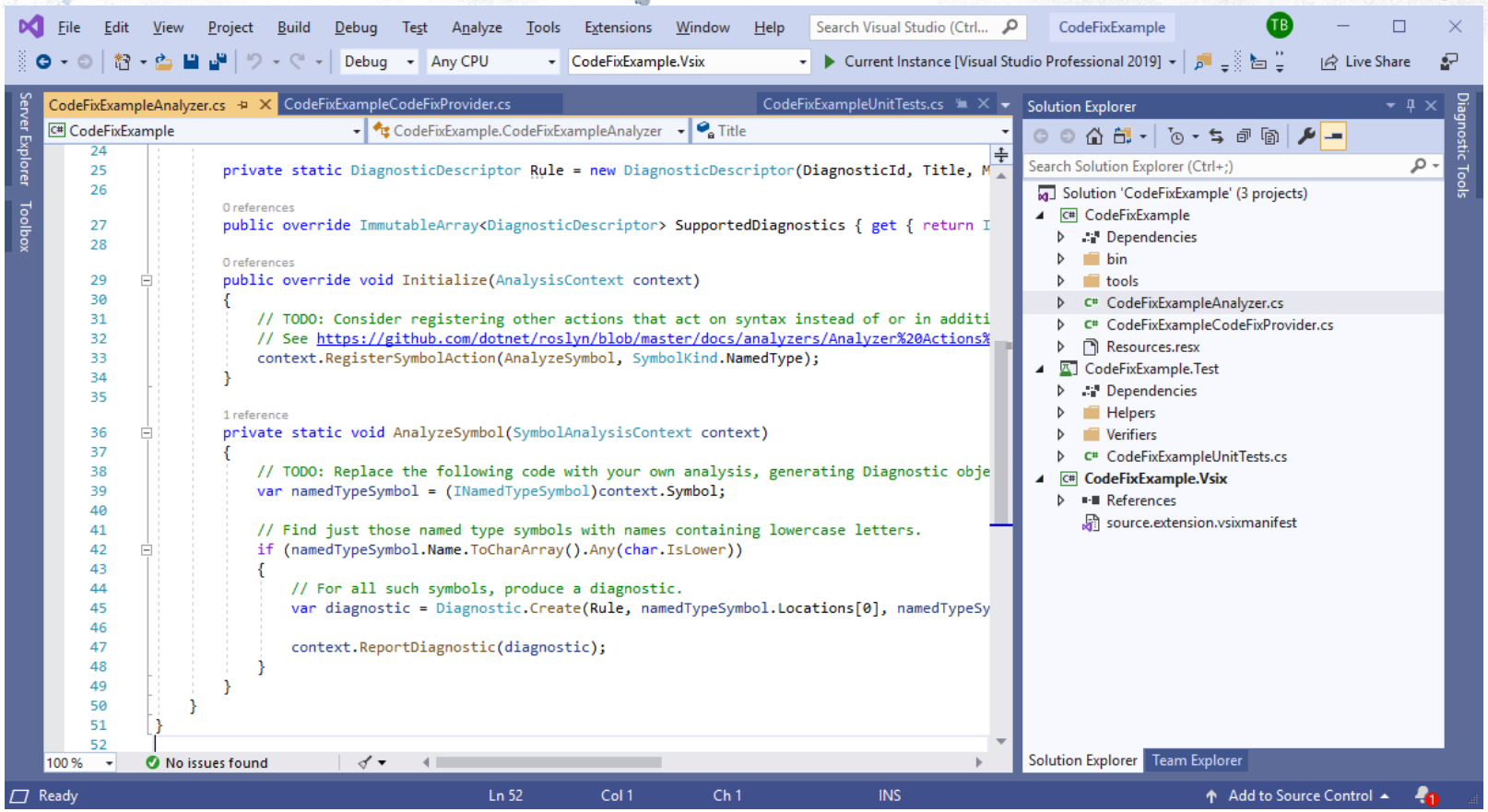
Template


Create a new project
Search for project templates
Language
Platform
Project type
Recent project templates
A list of your recently accessed templates will be displayed here.
Stand-Alone Code Analysis Tool
Create a code analysis command-line application
Stand-Alone Code Analysis Tool
Create a code analysis command-line application
Code Refactoring (.NET Standard)
Create a C# refactoring, deployed as a VSIX extension
Analyzer with Code Fix (.NET Standard)
Create a C# analyzer that comes with a code fix and generates diagnostics. The analyzer can be deployed as either a NuGet package or a VSIX extension.
Analyzer with Code Fix (.NET Standard)
Create a Visual Basic analyzer that comes with a code fix and generates diagnostics. The analyzer can be deployed as either a NuGet package or a VSIX extension.
Code Refactoring (.NET Standard)
Create a Visual Basic refactoring, deployed as a VSIX extension
Not finding what you're looking for?
Install more tools and features
Back
Next

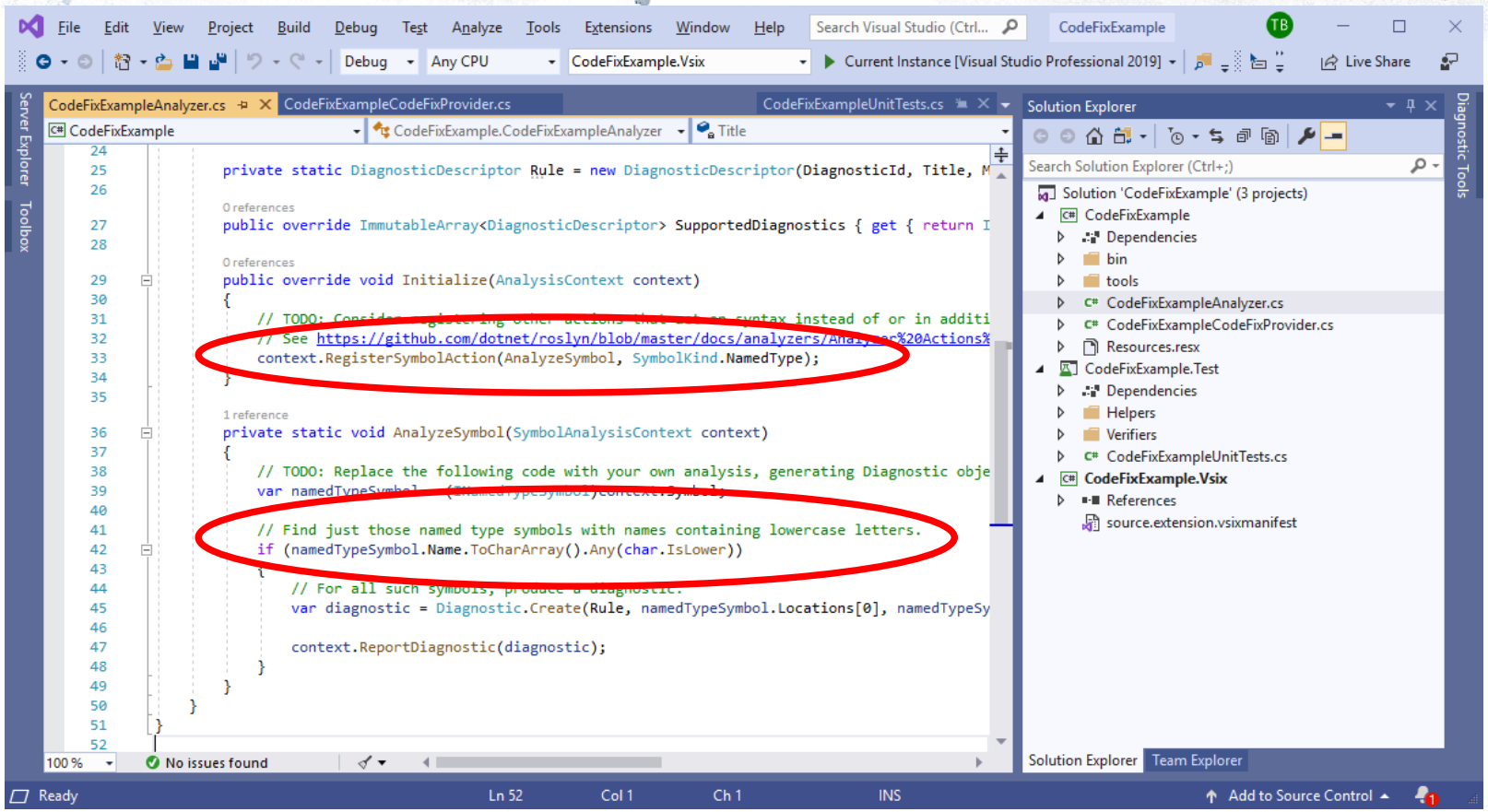

Analyzer with Code Fix (.NET Standard)
Create a C# analyzer that comes with a code fix and generates diagnostics. The analyzer can be deployed as either a NuGet package or a VSIX extension.


Configure your new project
Analyzer with Code Fix (.NET Standard)
Project name
CodeFixExample
Location
C:\Users\Timbo\SkyDrive\Documents\InterpreterInCSharp
...
Solution name
CodeFixExample
Place solution and project in the same directory
Framework
.NET Framework 4.7.2
Back
Create

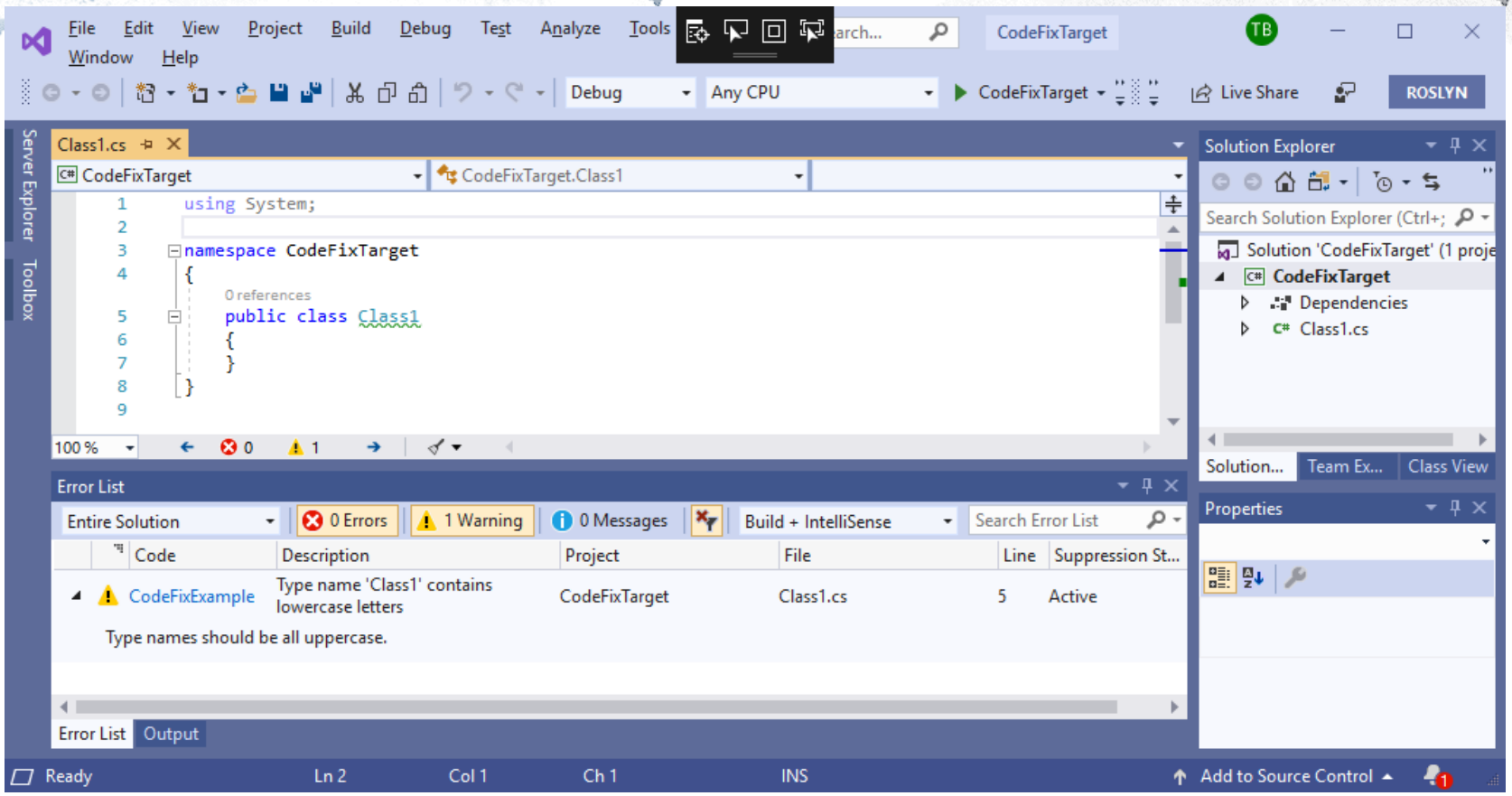
Code Fixes / Analyzers



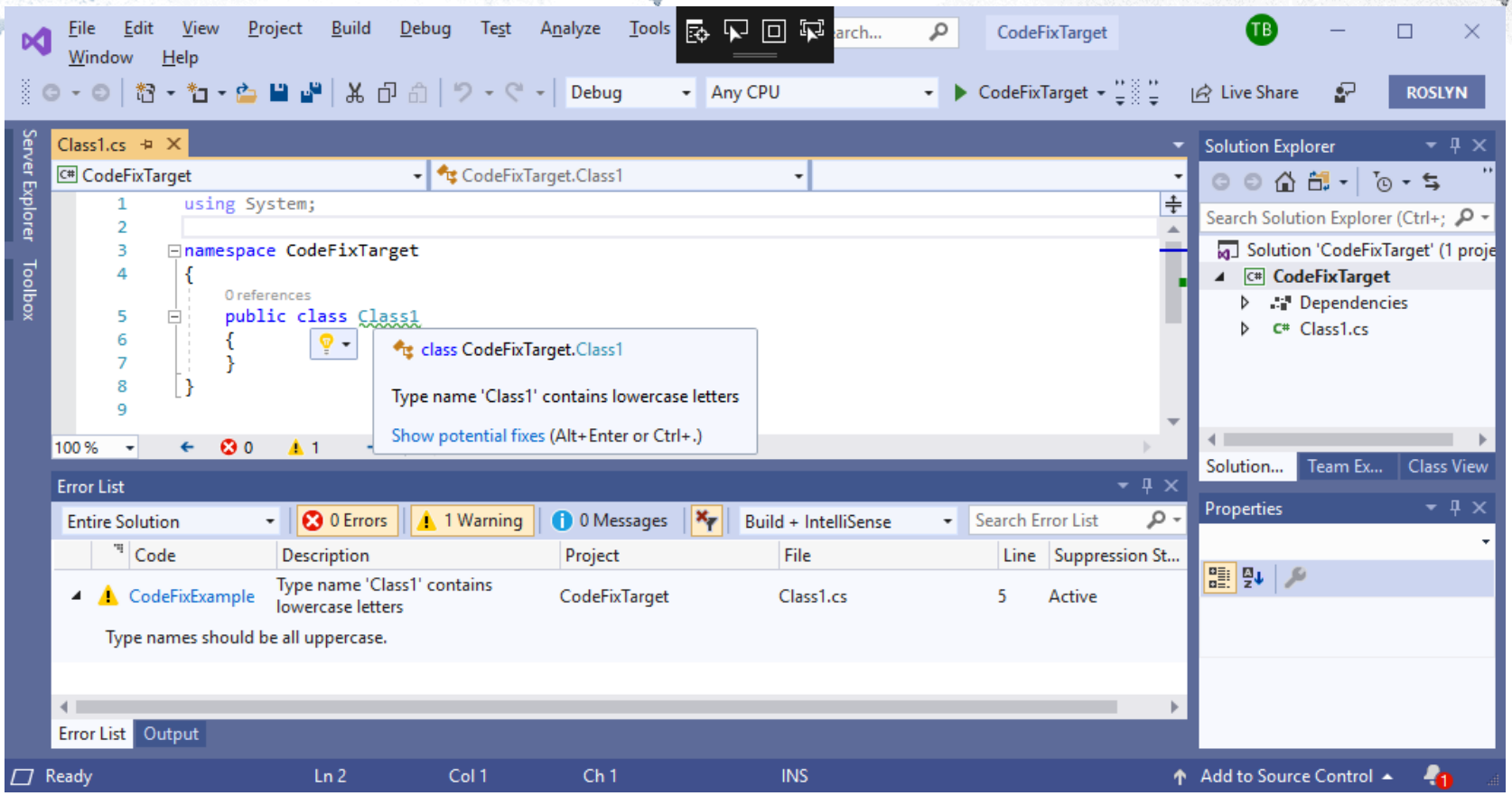
Code Fixes / Analyzers



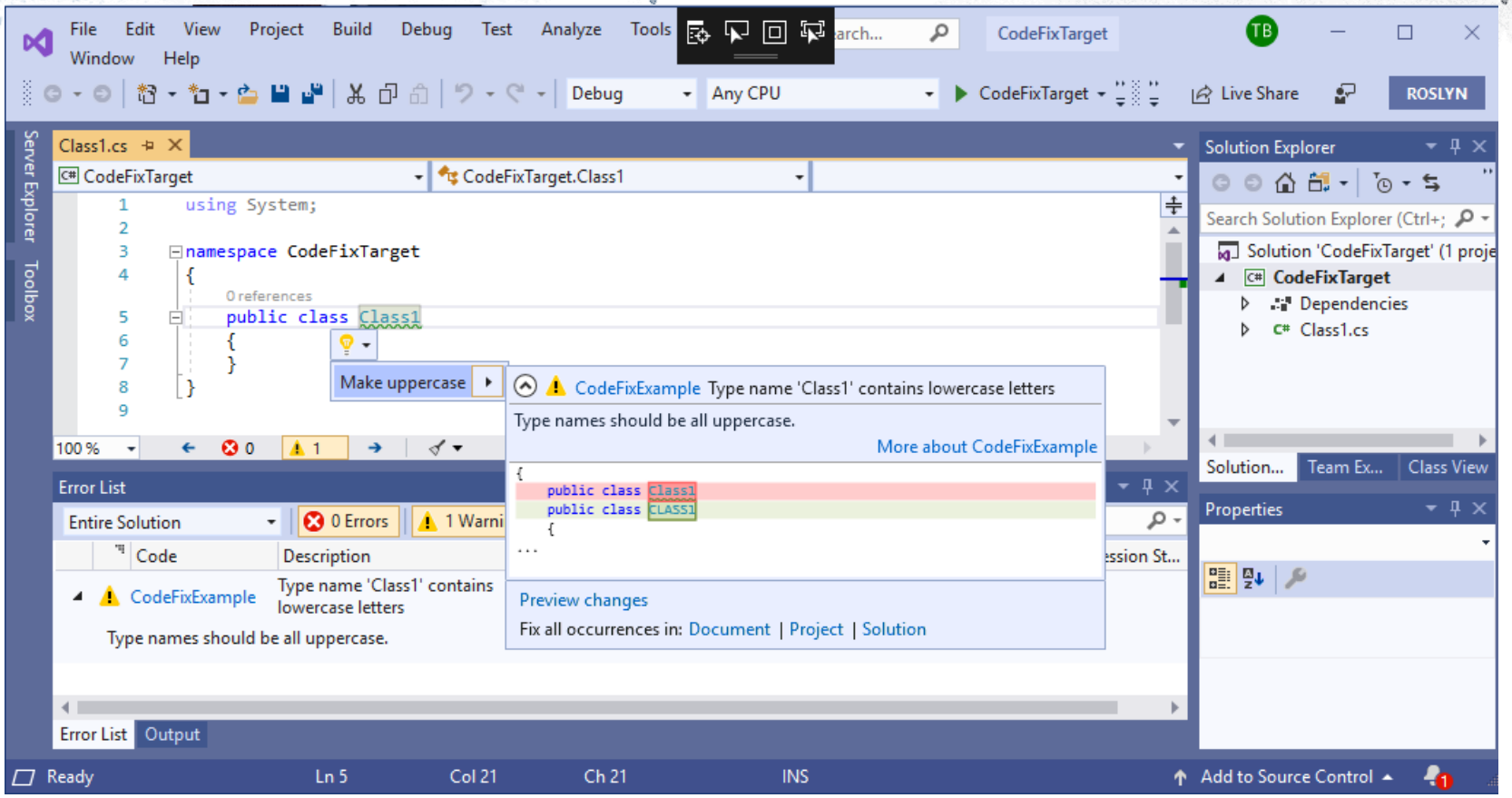
Code Fixes / Analyzers



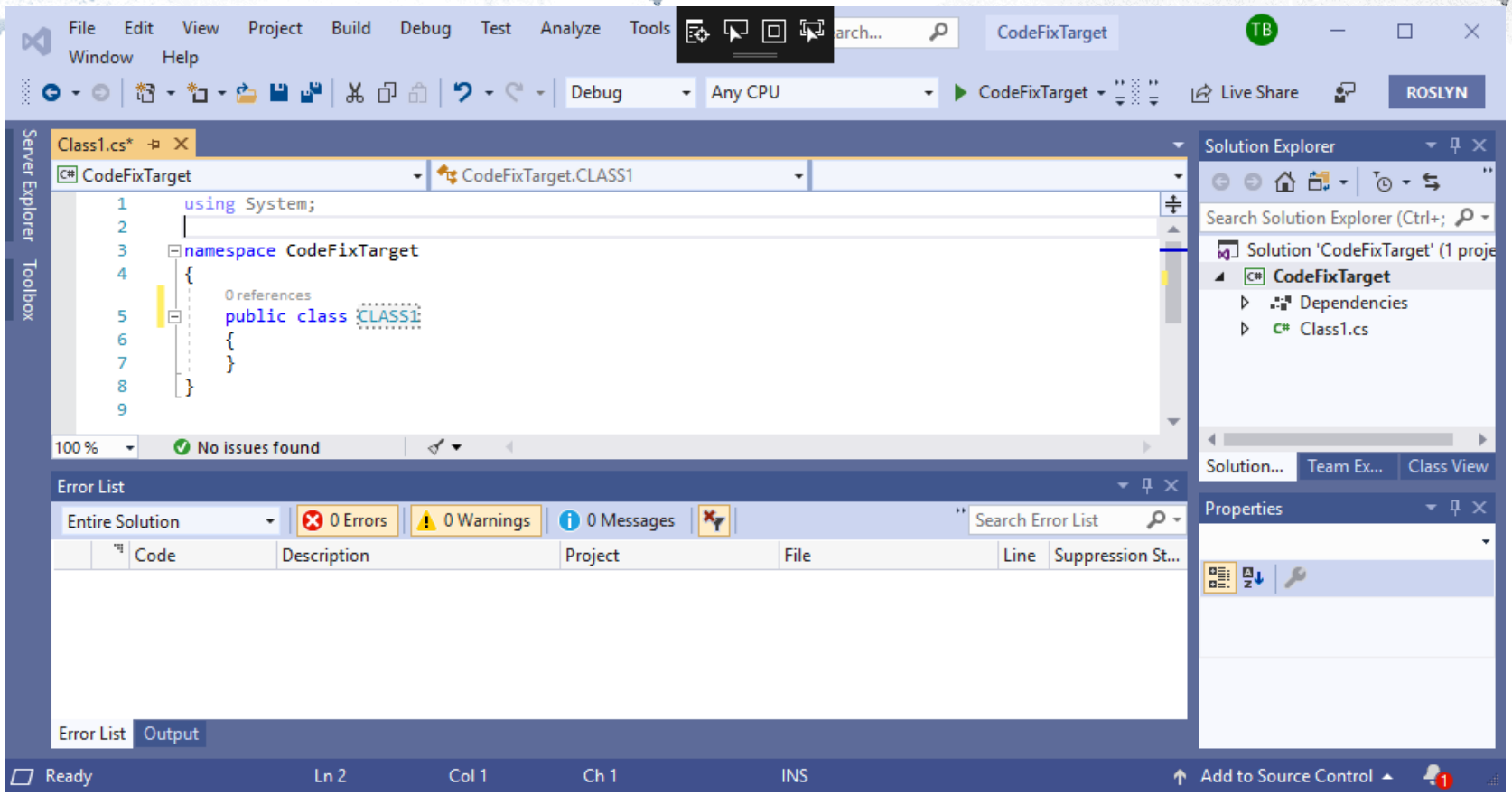
Code Fixes / Analyzers



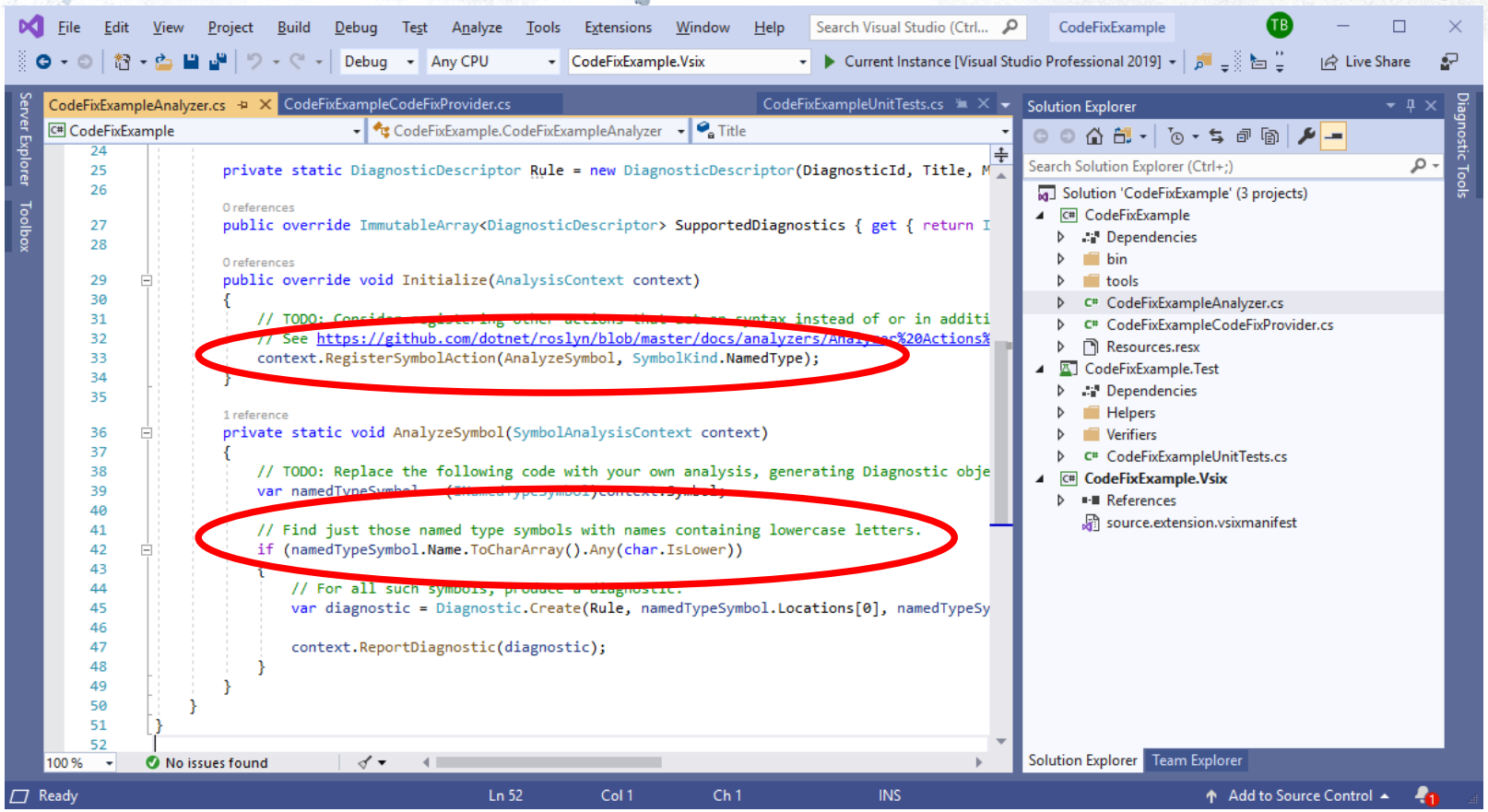
Code Fixes / Analyzers



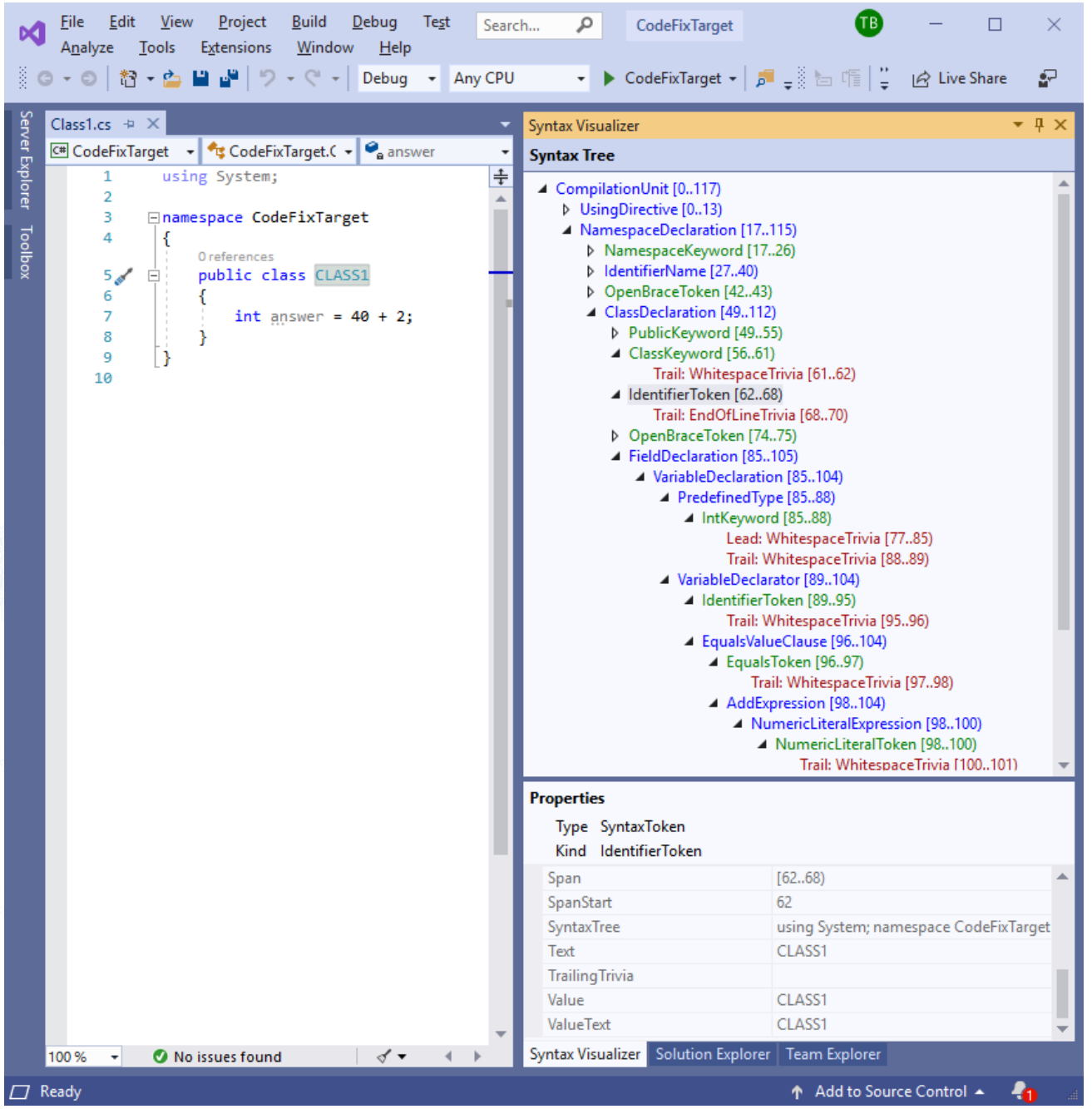
Code Fixes / Analyzers



Code Fixes / Analyzers



Syntax Visualizer

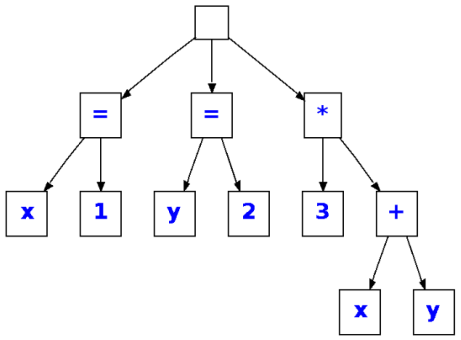


Interpreter

Source Code → Tokens → Abstract Syntax Tree (AST) → Evaluation

Lexical Analysis

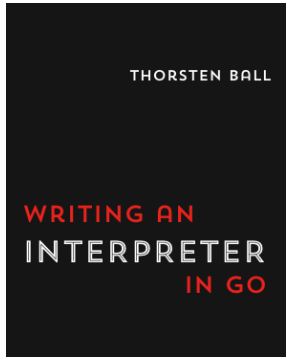
Parsing



Lexical Analysis

- Chop up text into pieces which are more meaningful to the parser
- These pieces are called **tokens**

Lexing



```
let answer = 40 + 2;
```

Lexing

```
let answer = 40 + 2;
```

Lexing

```
let answer = 40 + 2;
```

Lexing


```
let answer = 40 + 2;
```

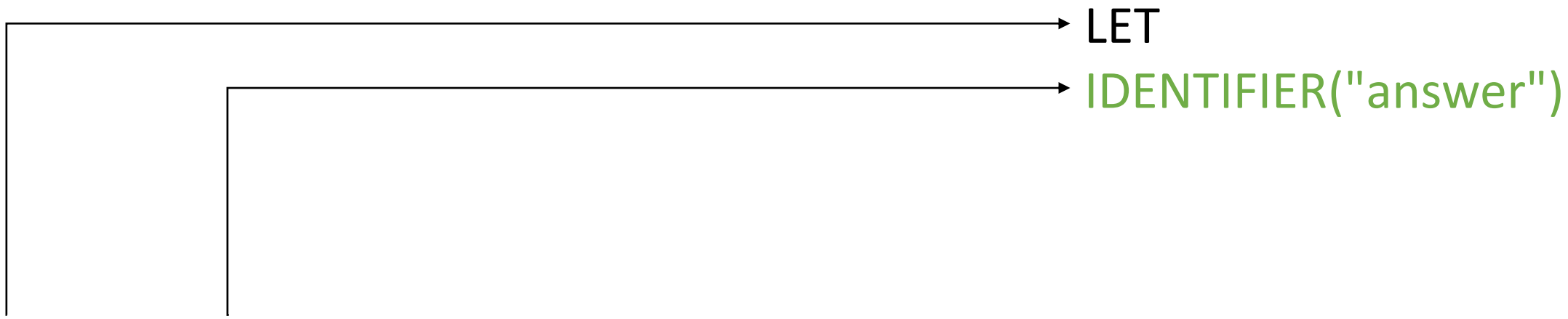
Lexing

let answer = 40 + 2;

Lexing

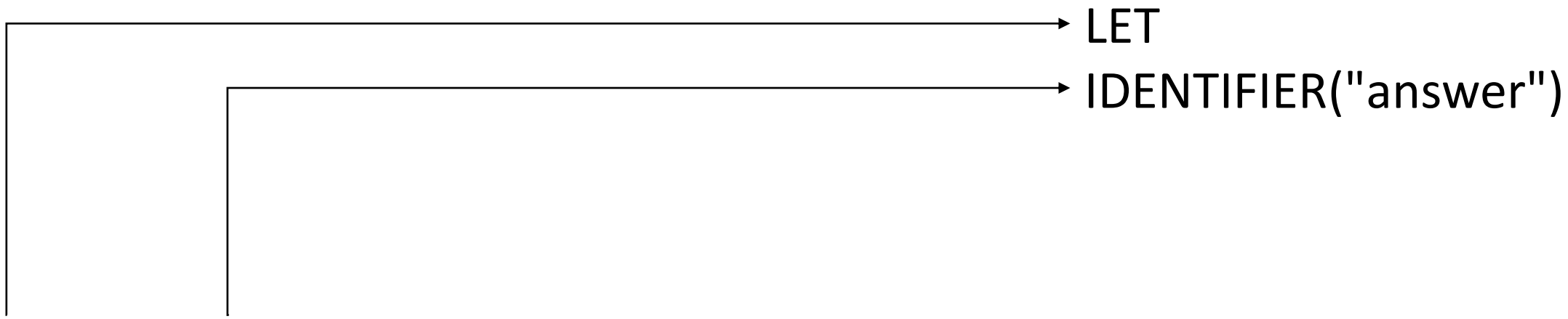
let answer = 40 + 2;

Lexing



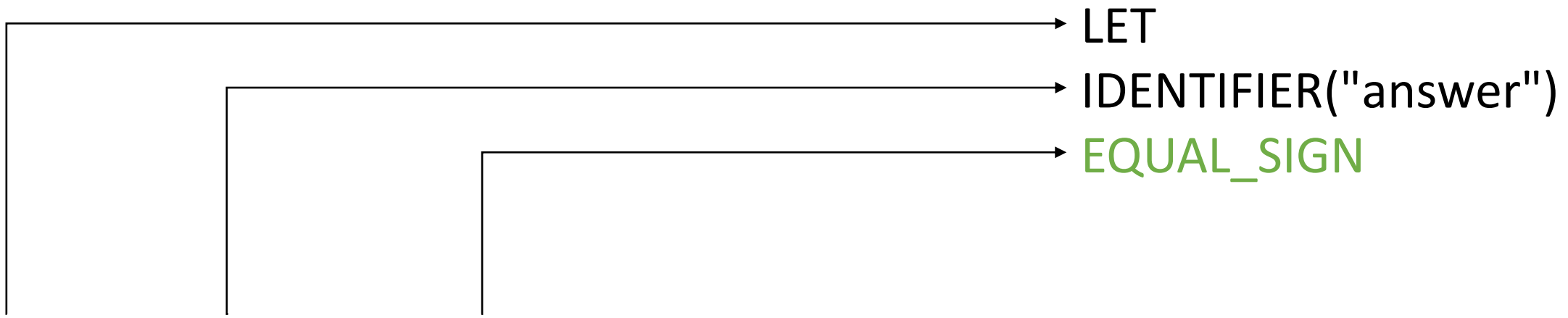

```
let answer = 40 + 2;
```

Lexing



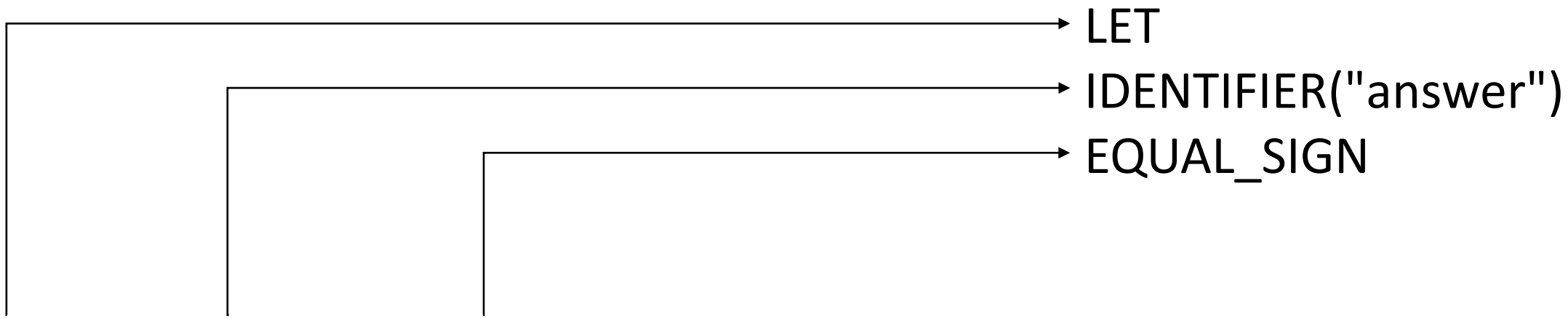

```
let answer = 40 + 2;
```

Lexing



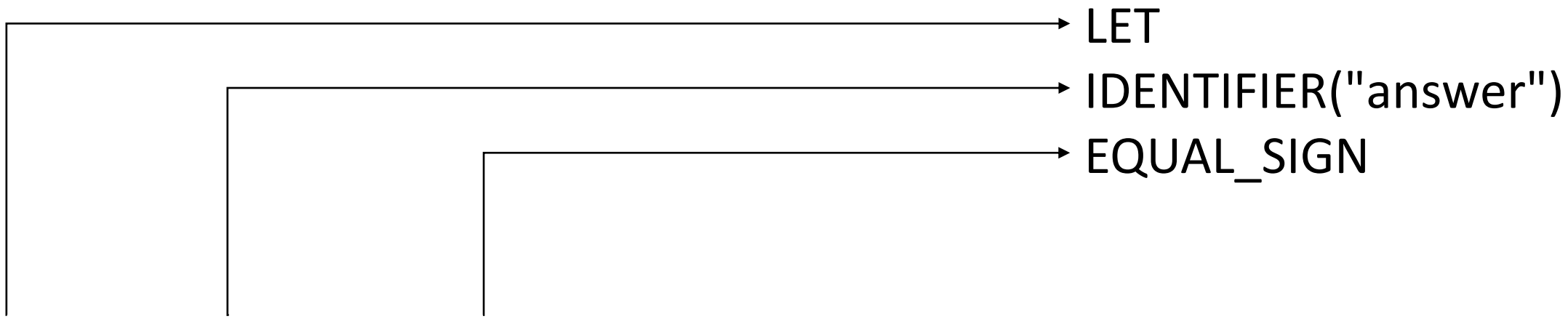
let answer = 40 + 2;

Lexing



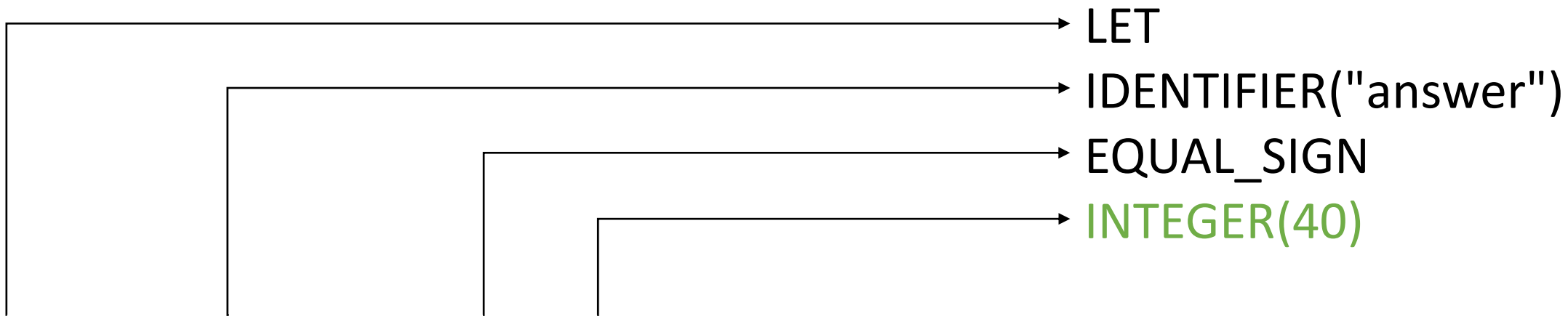
let answer = 40 + 2;

Lexing



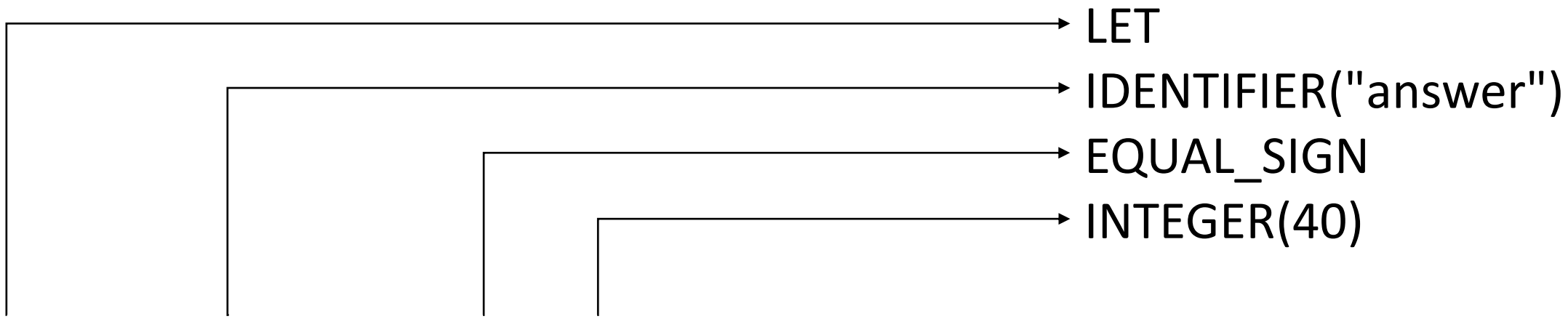

```
let answer = 40 + 2;
```

Lexing



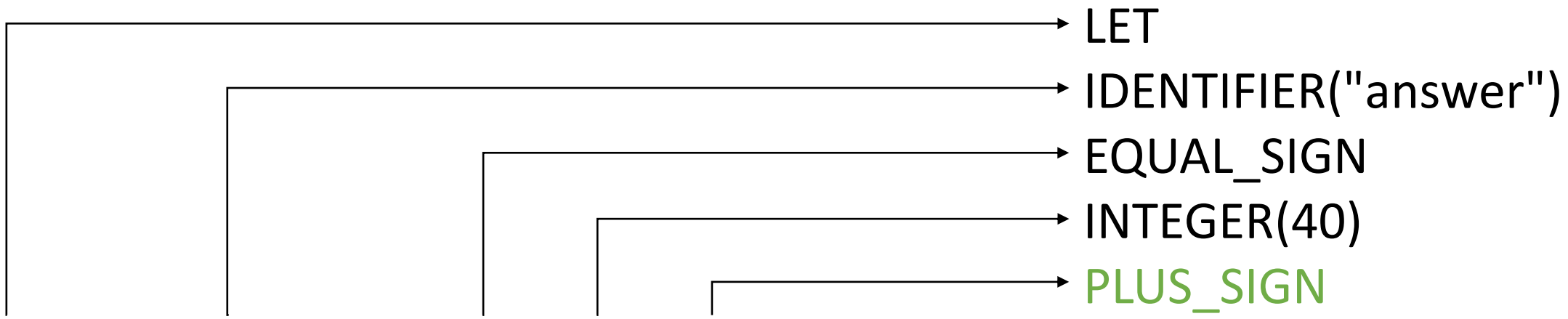
let answer = 40 + 2;

Lexing



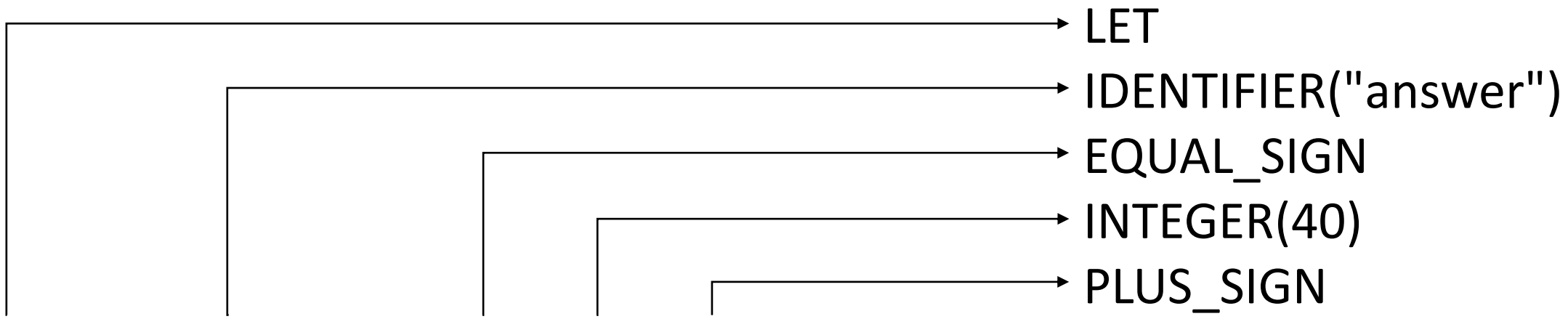

```
let answer = 40 + 2;
```

Lexing



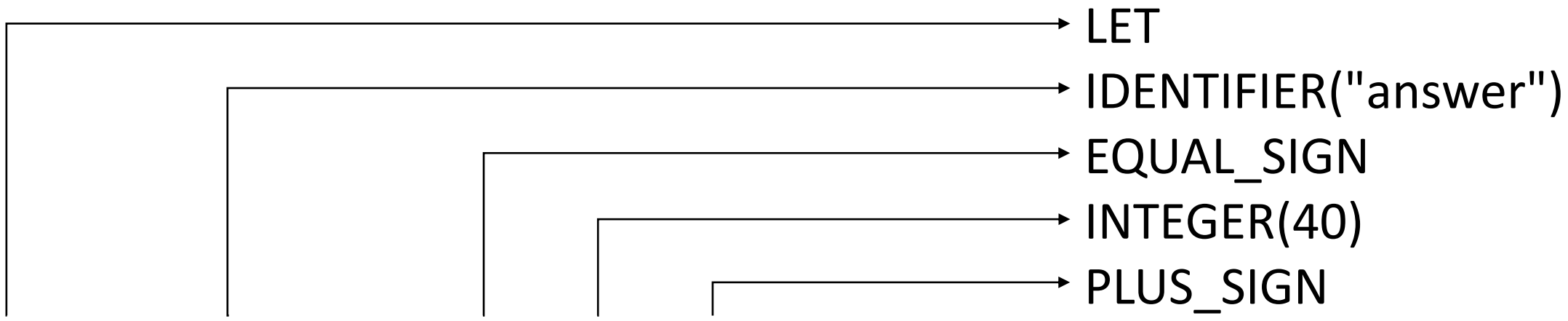

```
let answer = 40 + 2;
```

Lexing



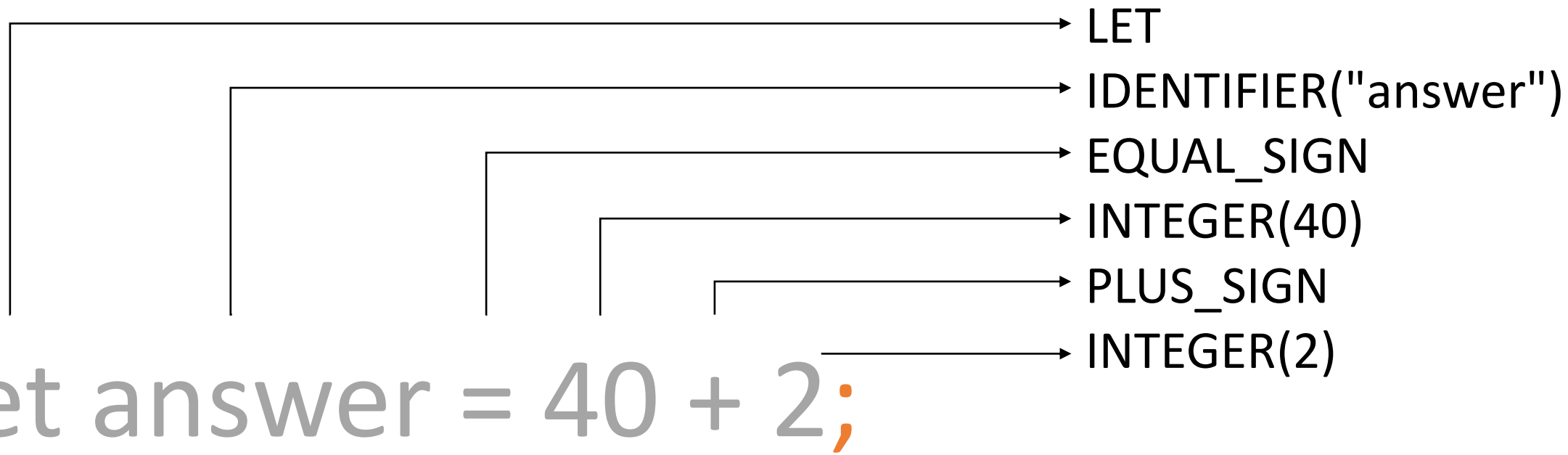

```
let answer = 40 + 2;
```

Lexing

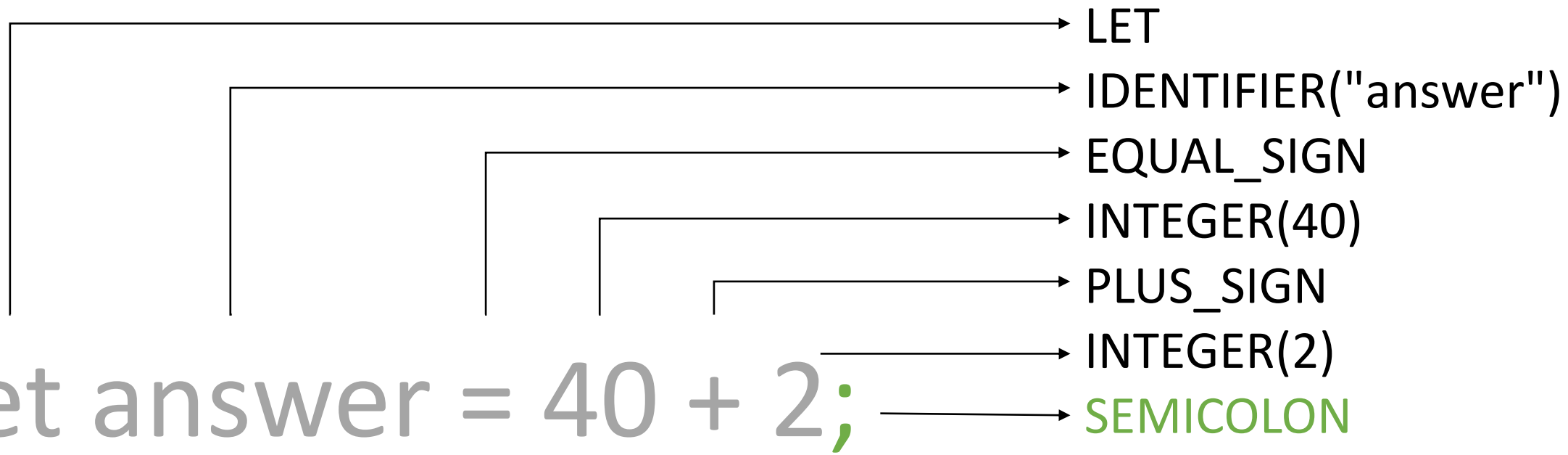


let answer = 40 + 2;

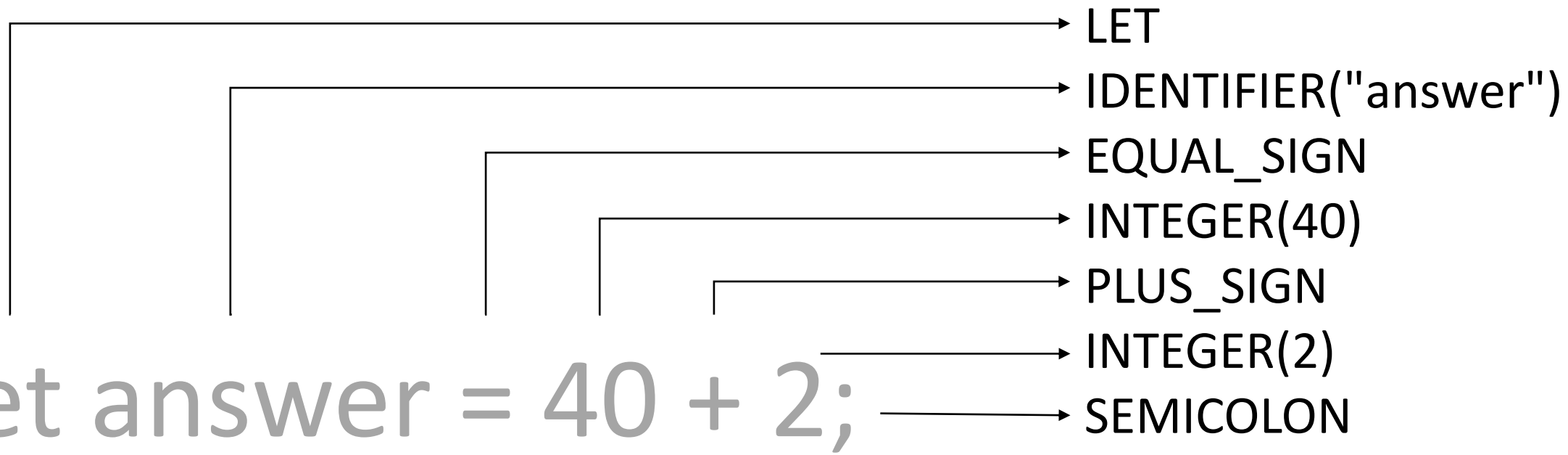
Lexing


LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
let answer = 40 + 2;

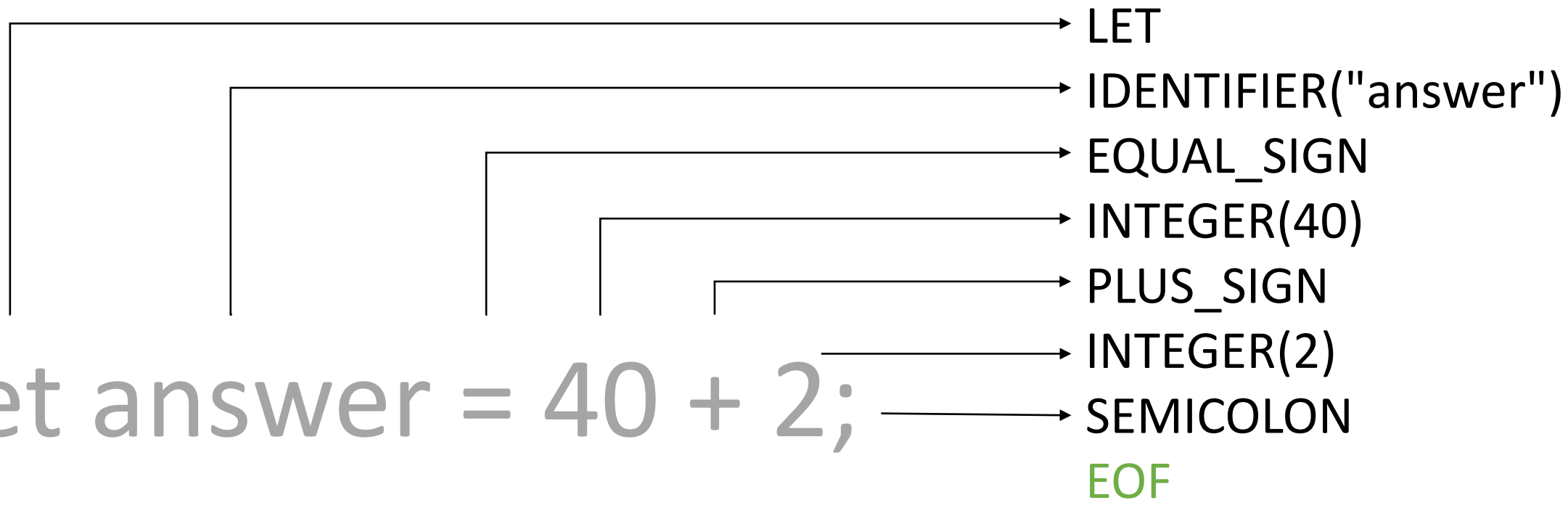
Lexing


LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
let answer = 40 + 2;

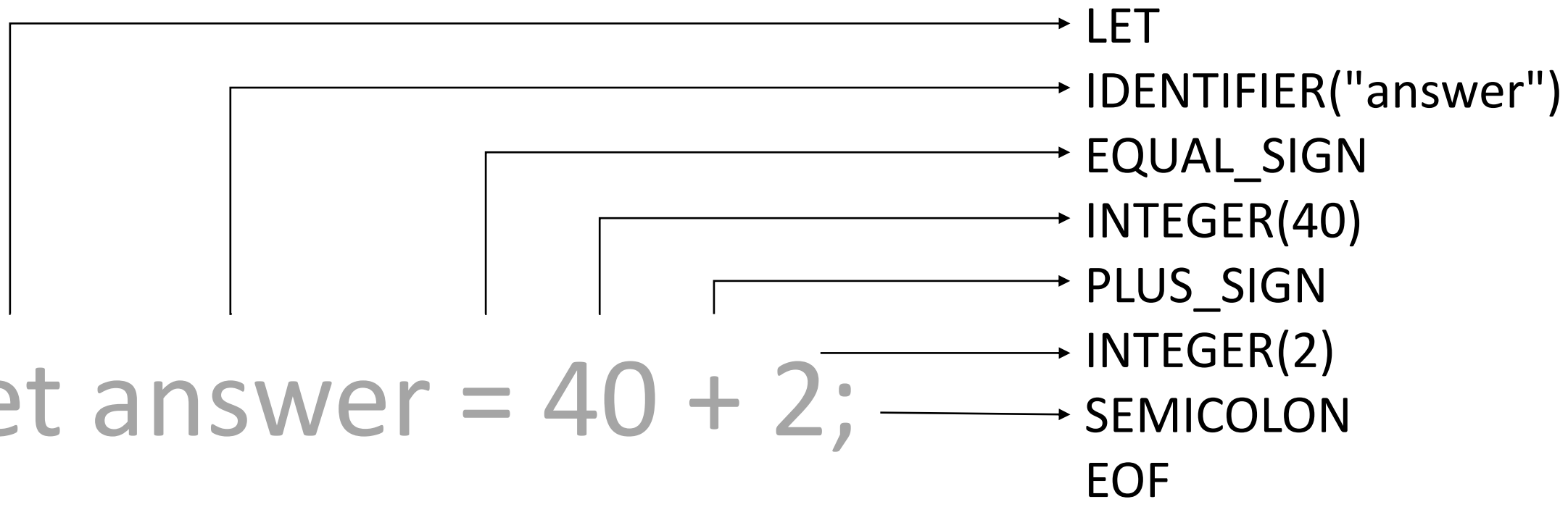
Lexing


let answer = 40 + 2;
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON

Lexing


let answer = 40 + 2;
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF

Lexing


let answer = 40 + 2;
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF

Parsing

Parser takes input data and builds a data structure giving a structural representation of the input, checking for correct syntax in the process.

Wikipedia

Parsing

```
let answer = 40 + 2;
```

Parsing

let answer = 40 + 2;

LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF

Parsing

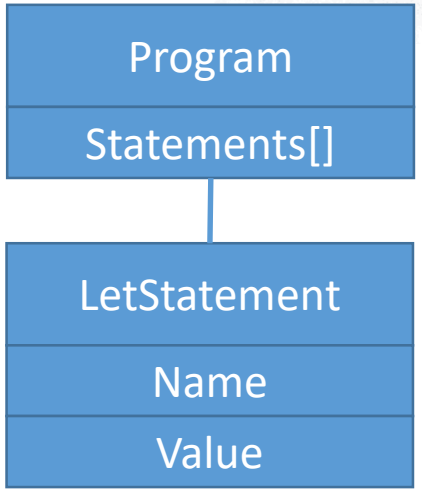
let answer = 40 + 2;

LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF

Parsing

let answer = 40 + 2;

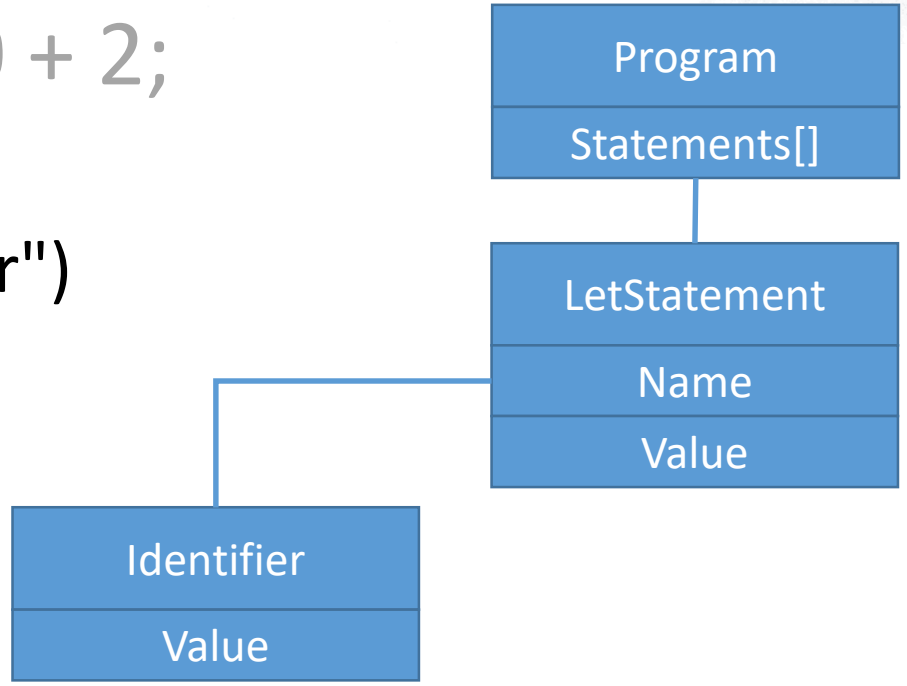
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

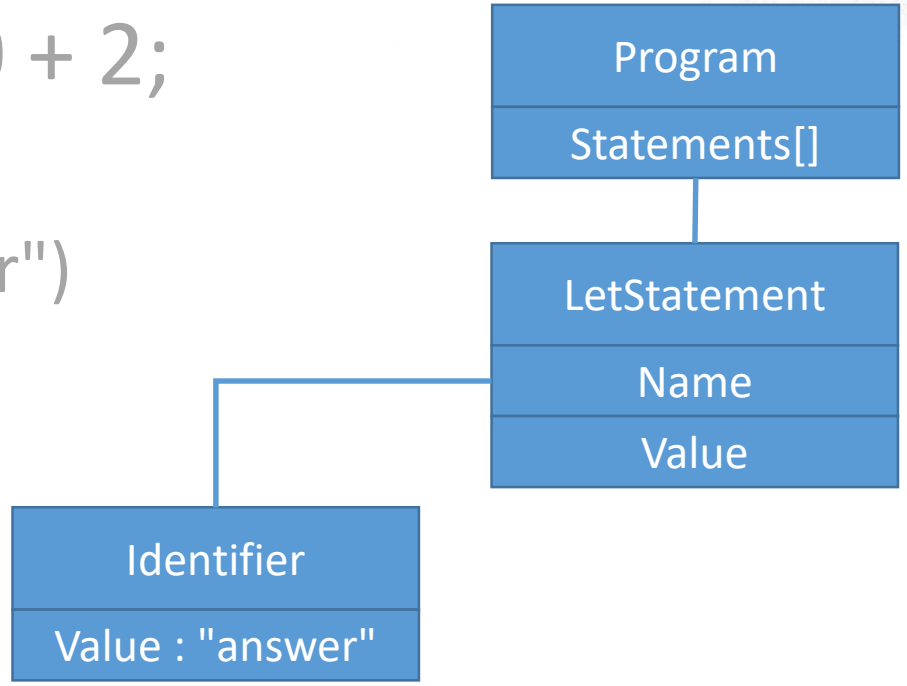
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

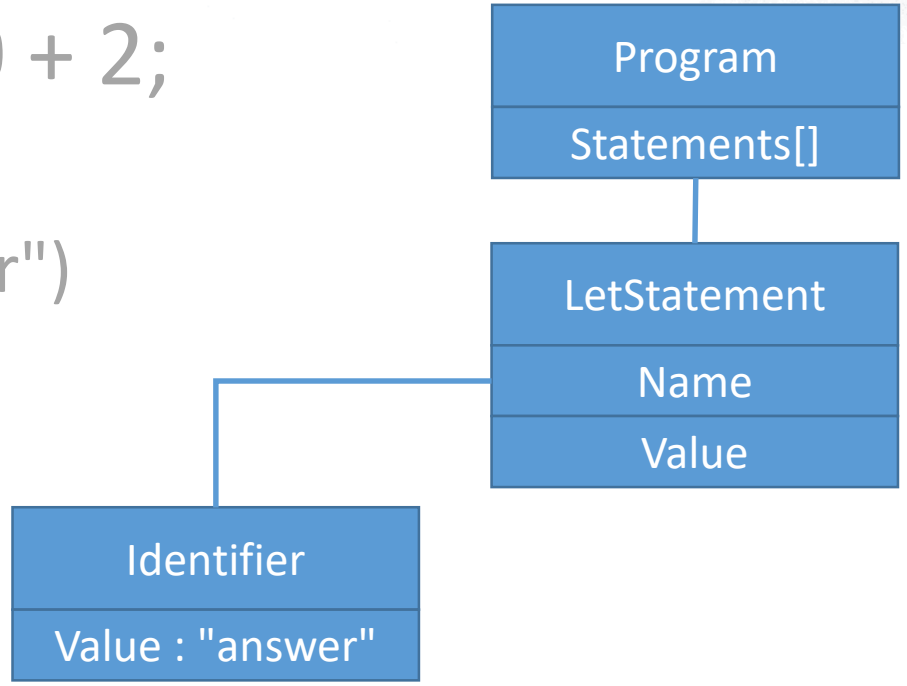
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

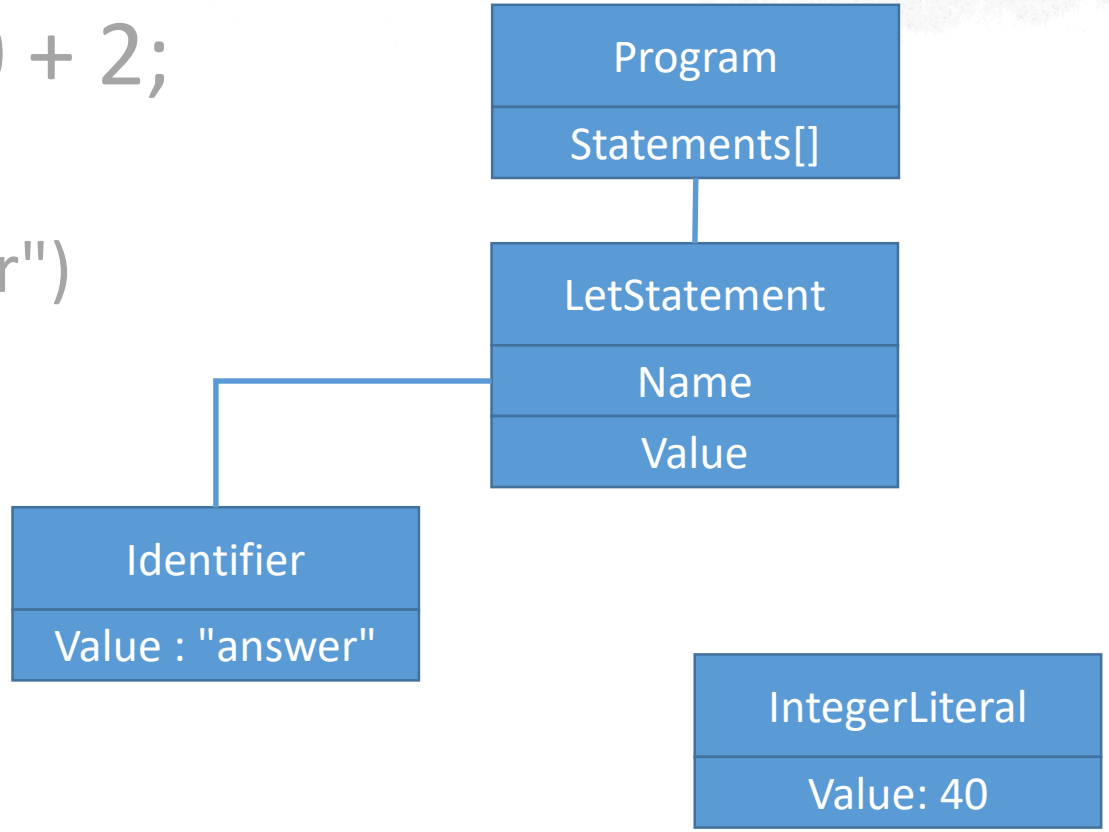
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

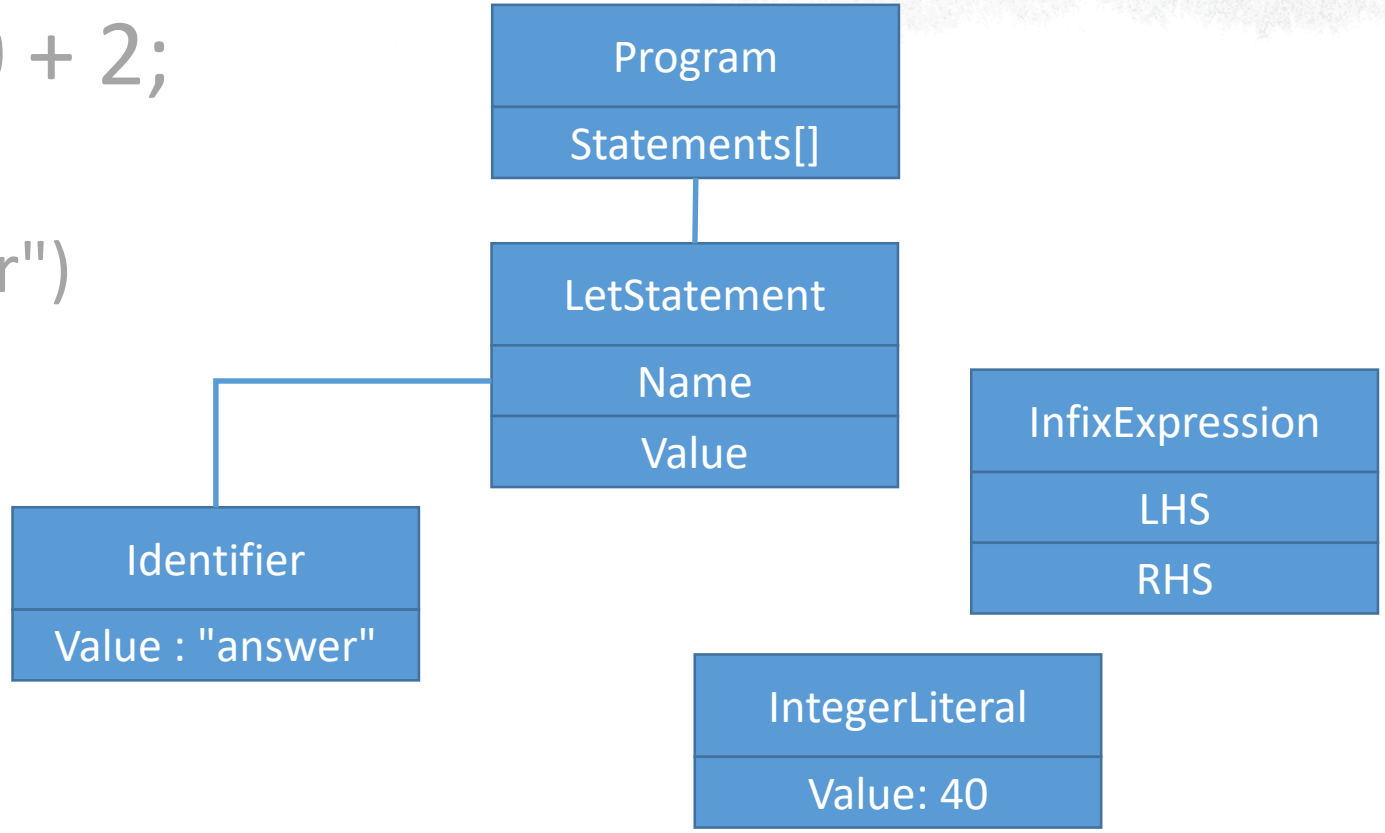
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

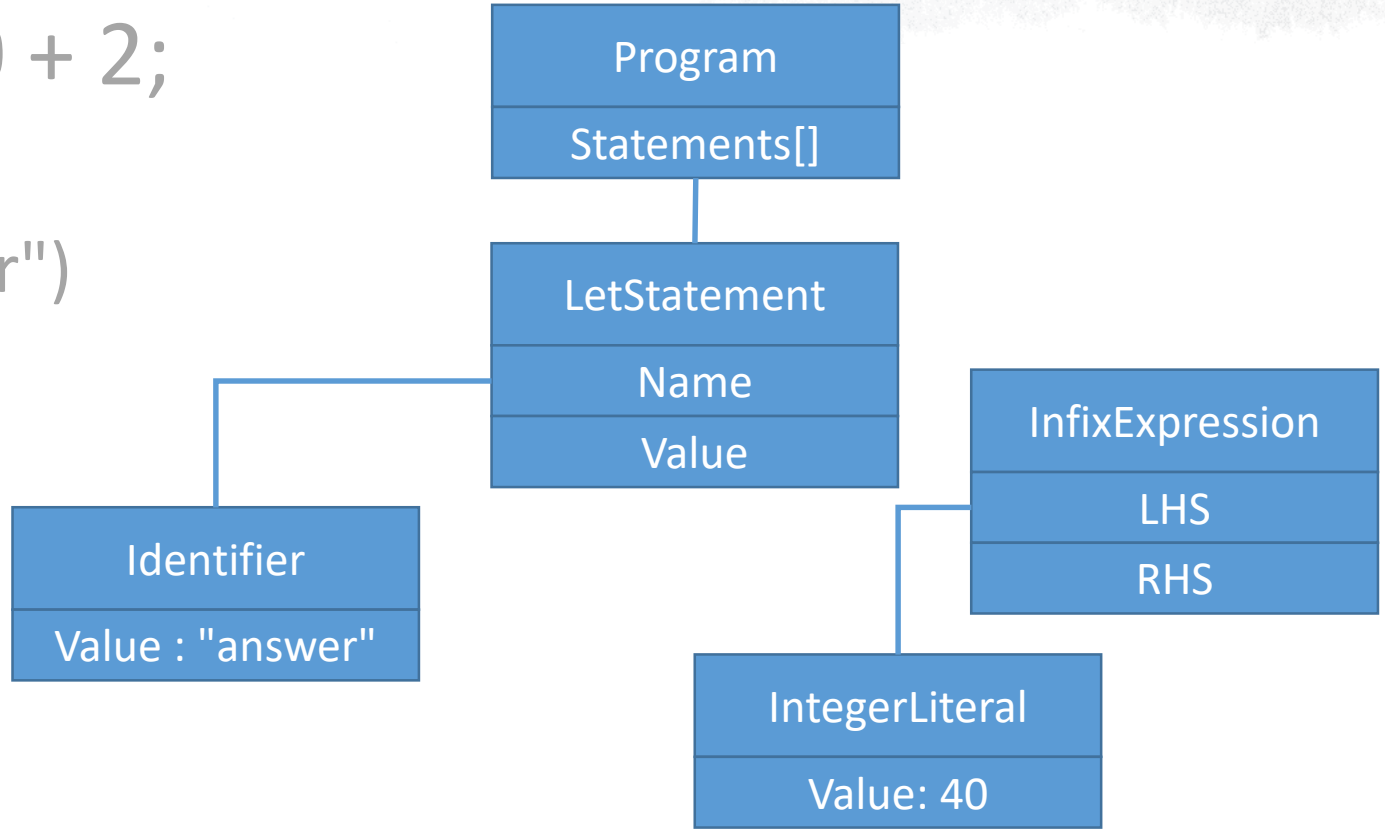
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

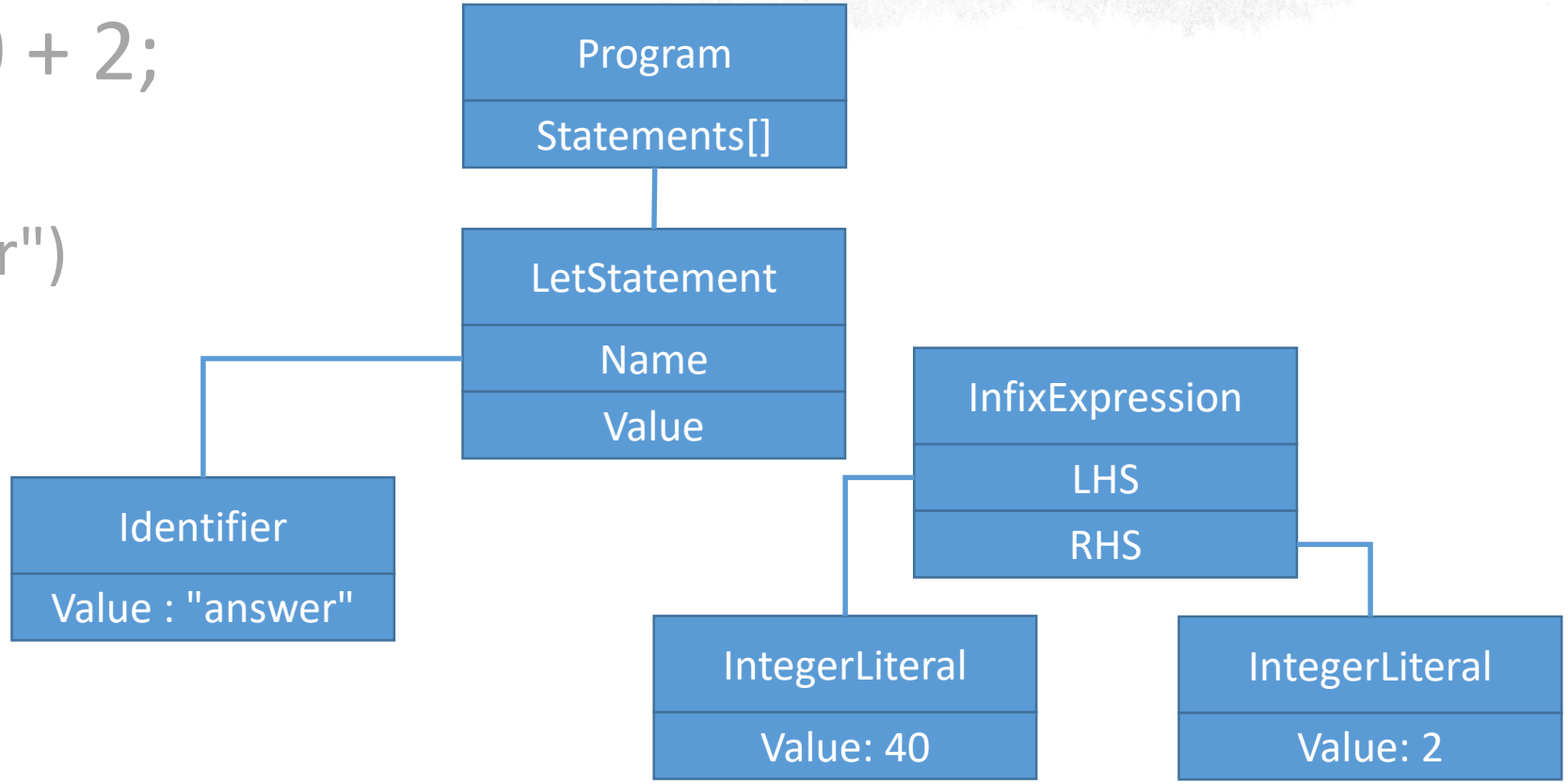
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

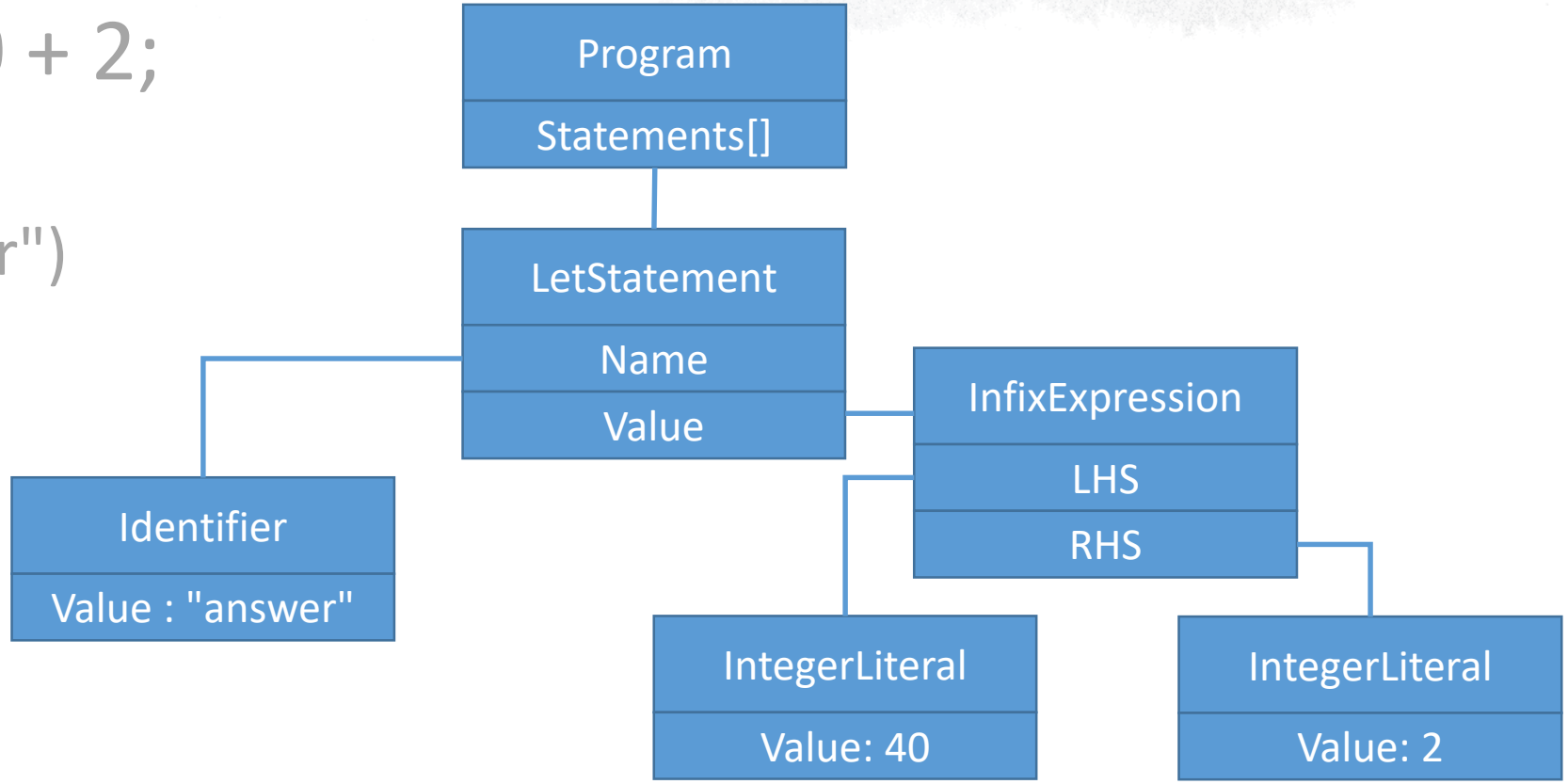
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

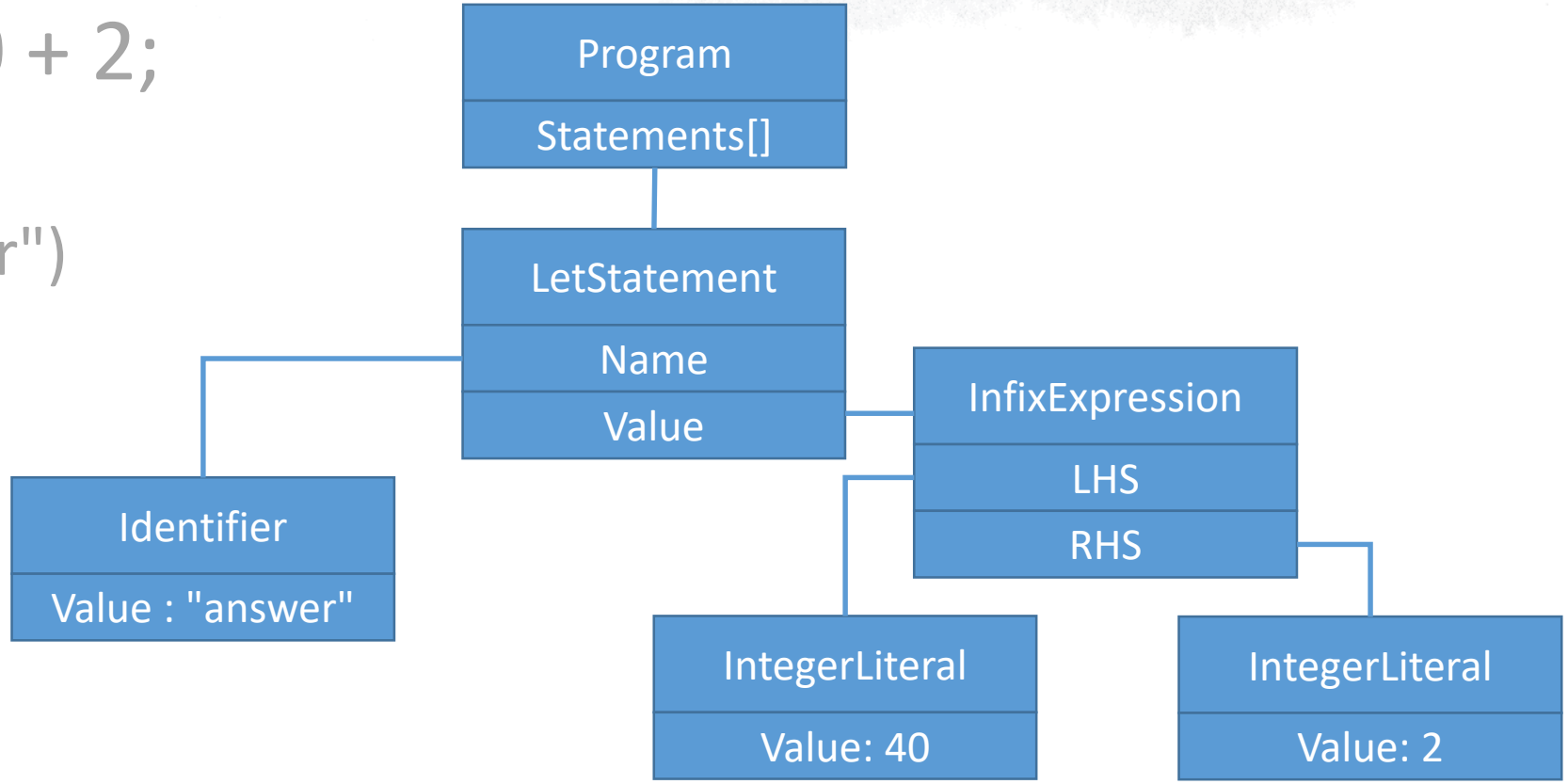
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

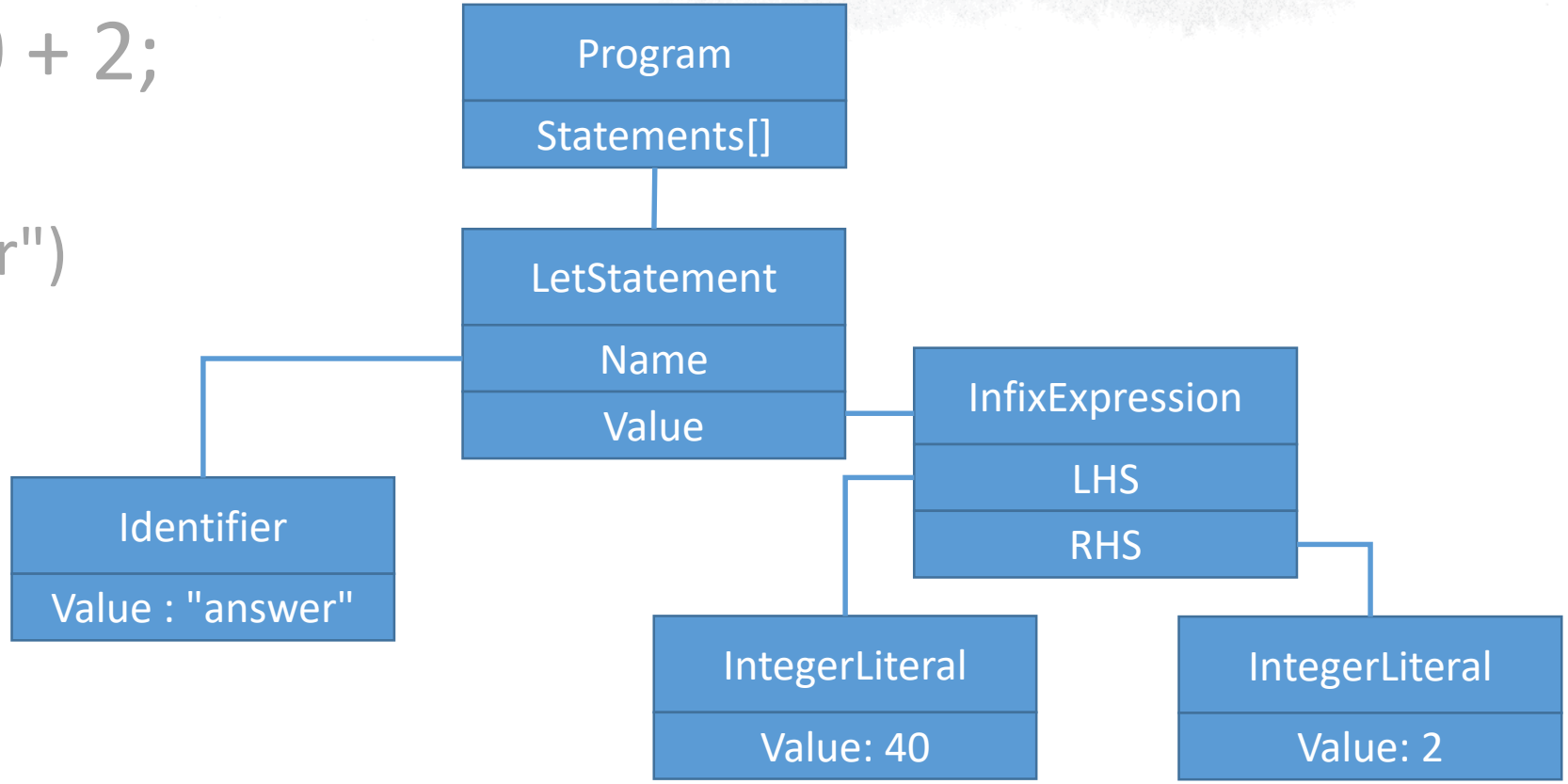
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Parsing

let answer = 40 + 2;

LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Roslyn

Syntax Visualizer

let answer = 40 + 2;

LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF

Syntax Visualizer

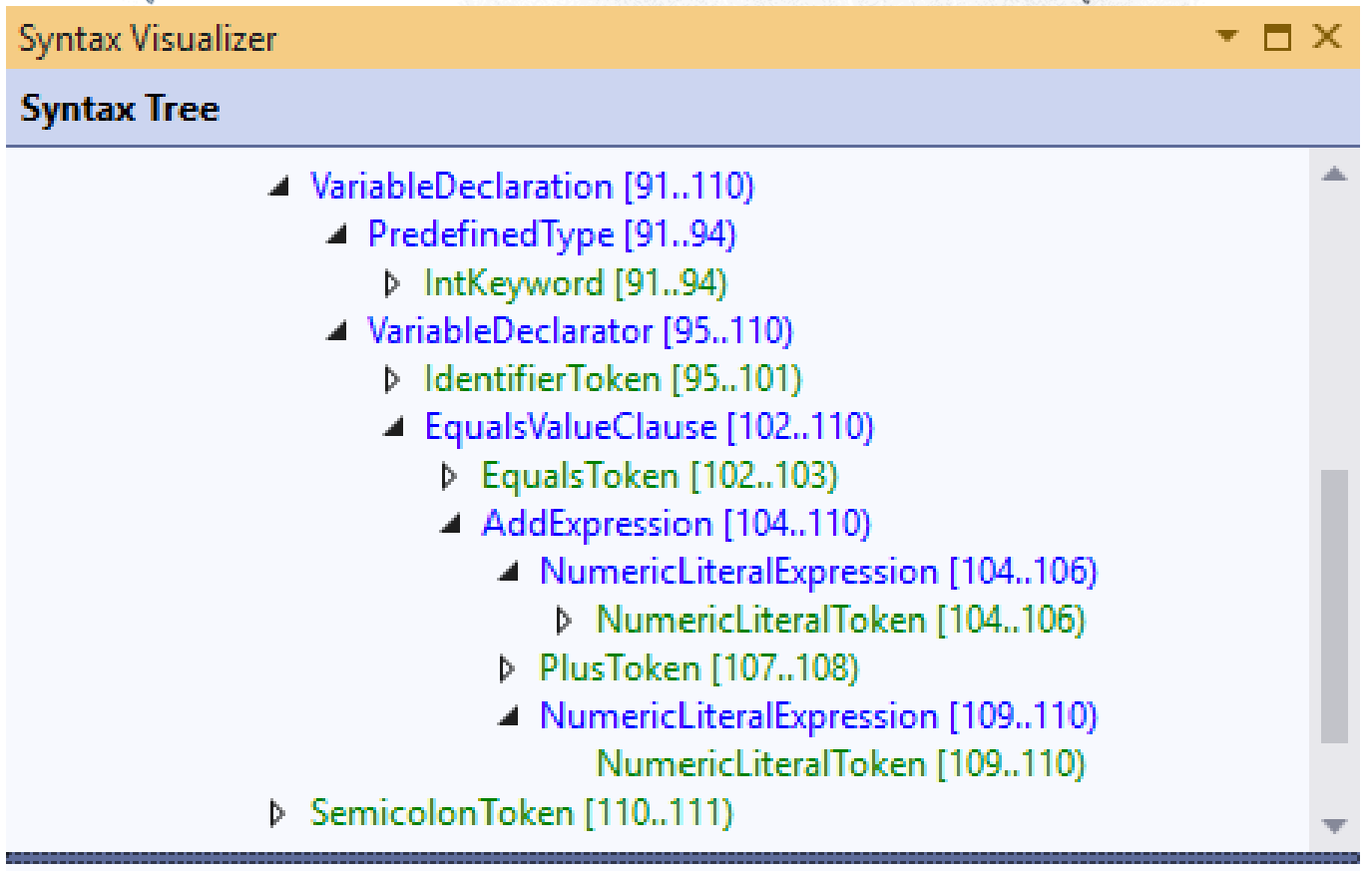
```
int answer = 40 + 2;
```

INT
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF

Visual Studio

```
int answer = 40 + 2;
```

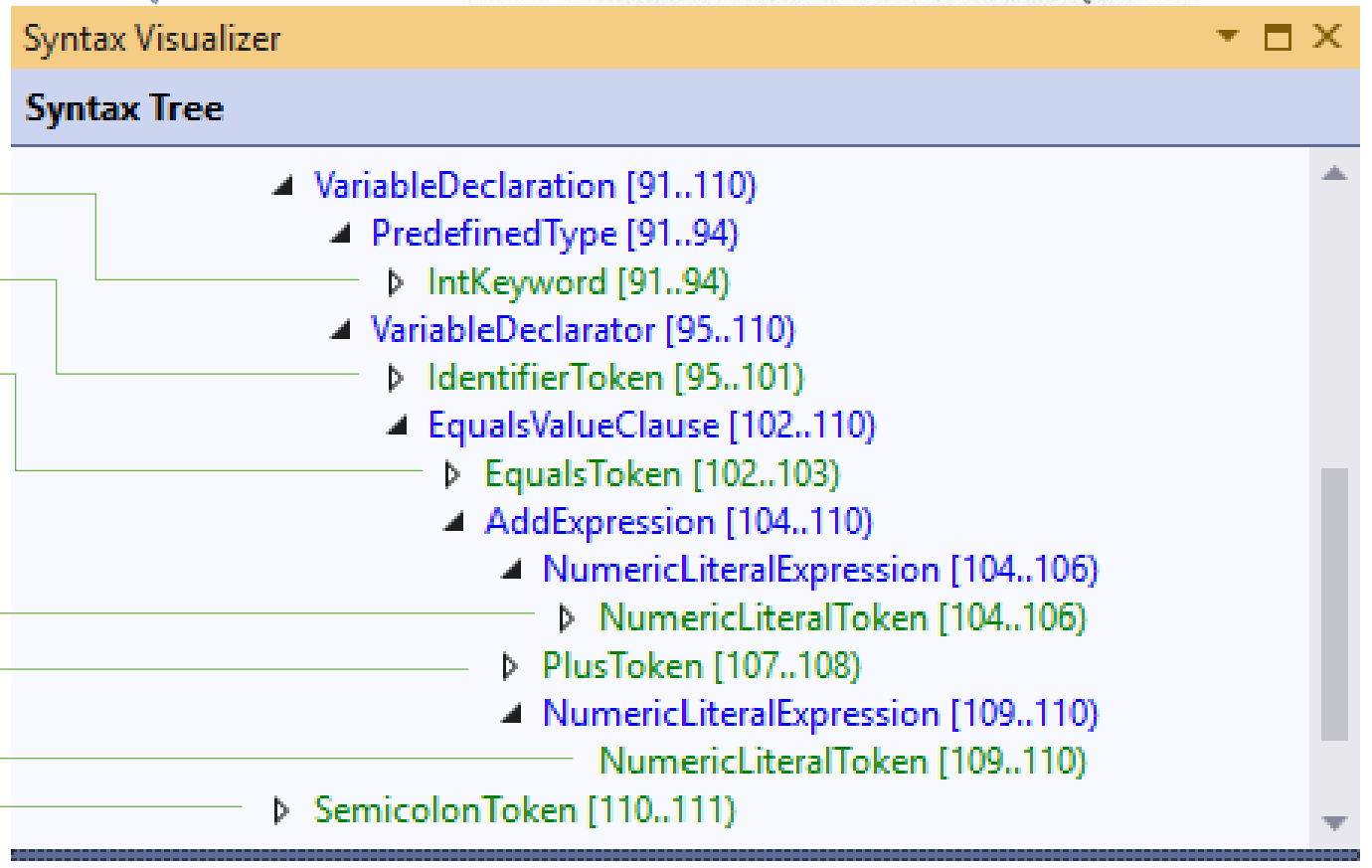
INT
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Visual Studio

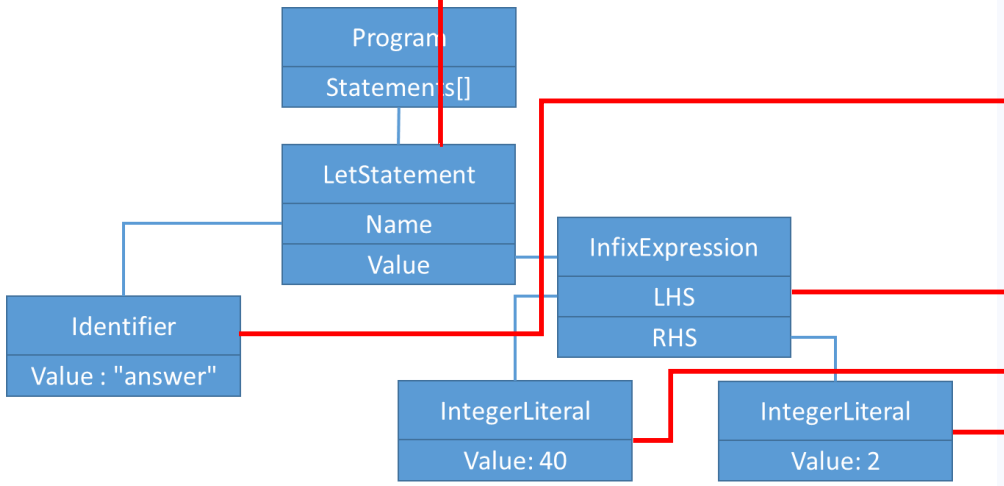
```
int answer = 40 + 2;
```

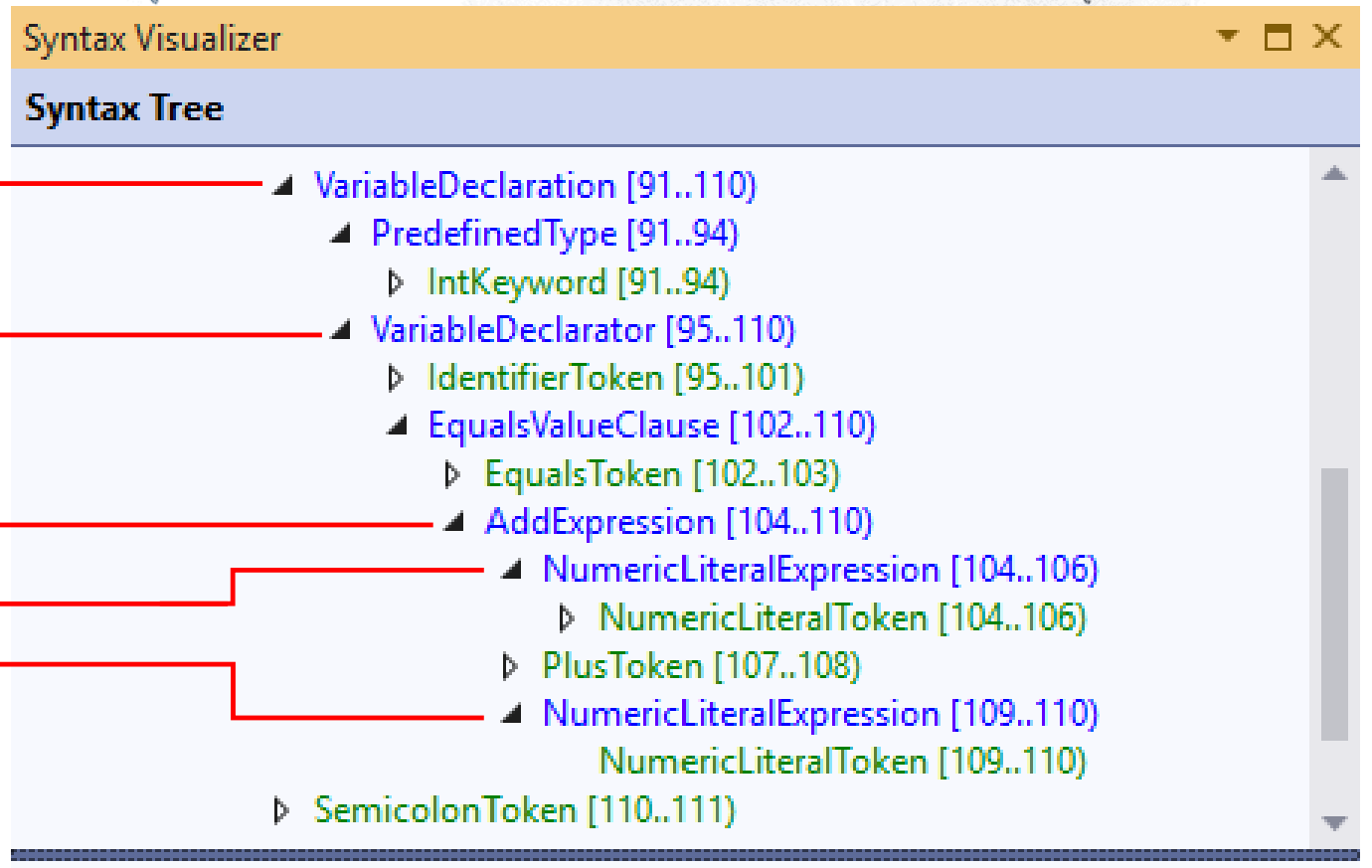
INT
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



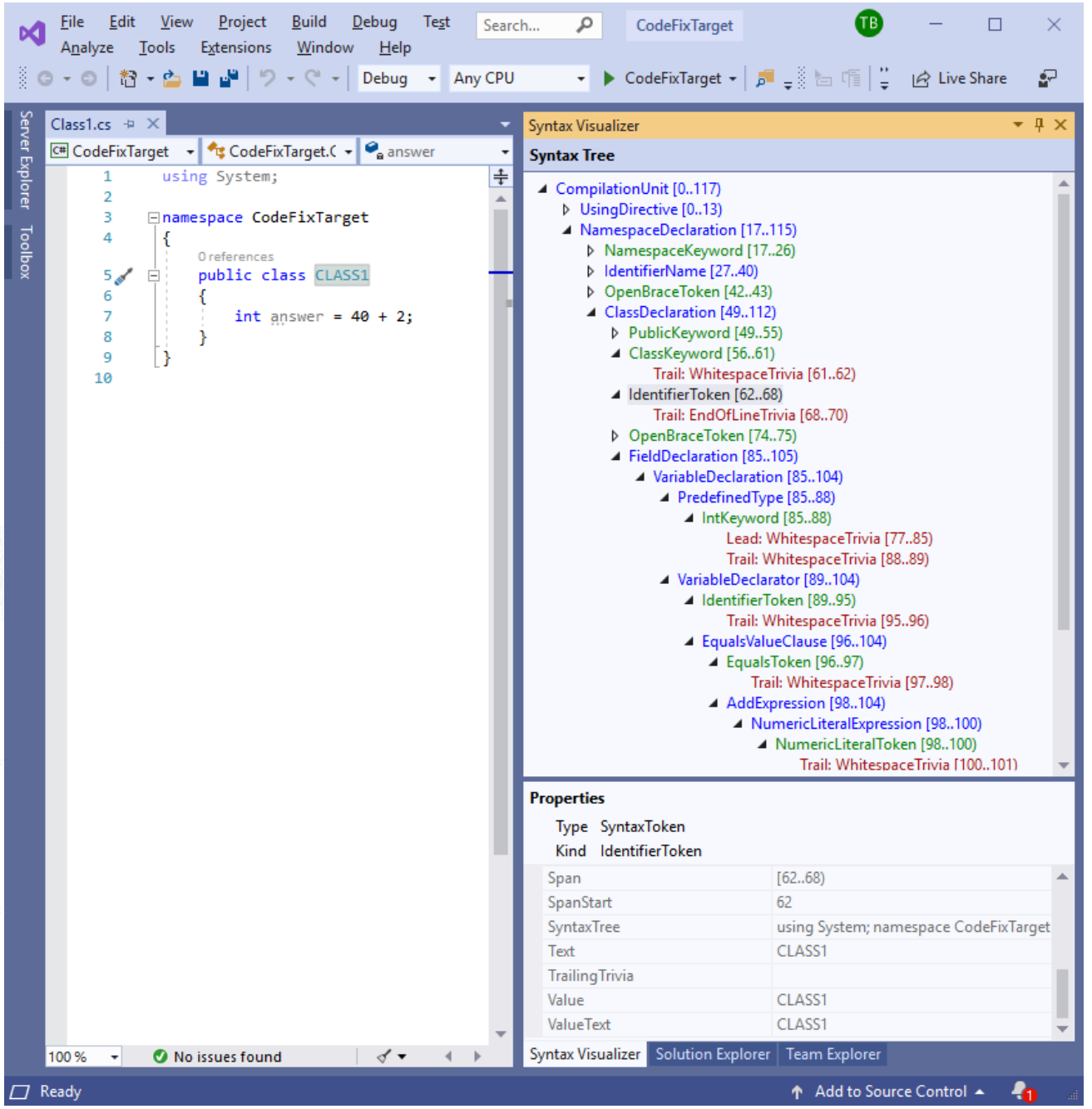
Visual Studio

```
int answer = 40 + 2;
```





Syntax Visualizer

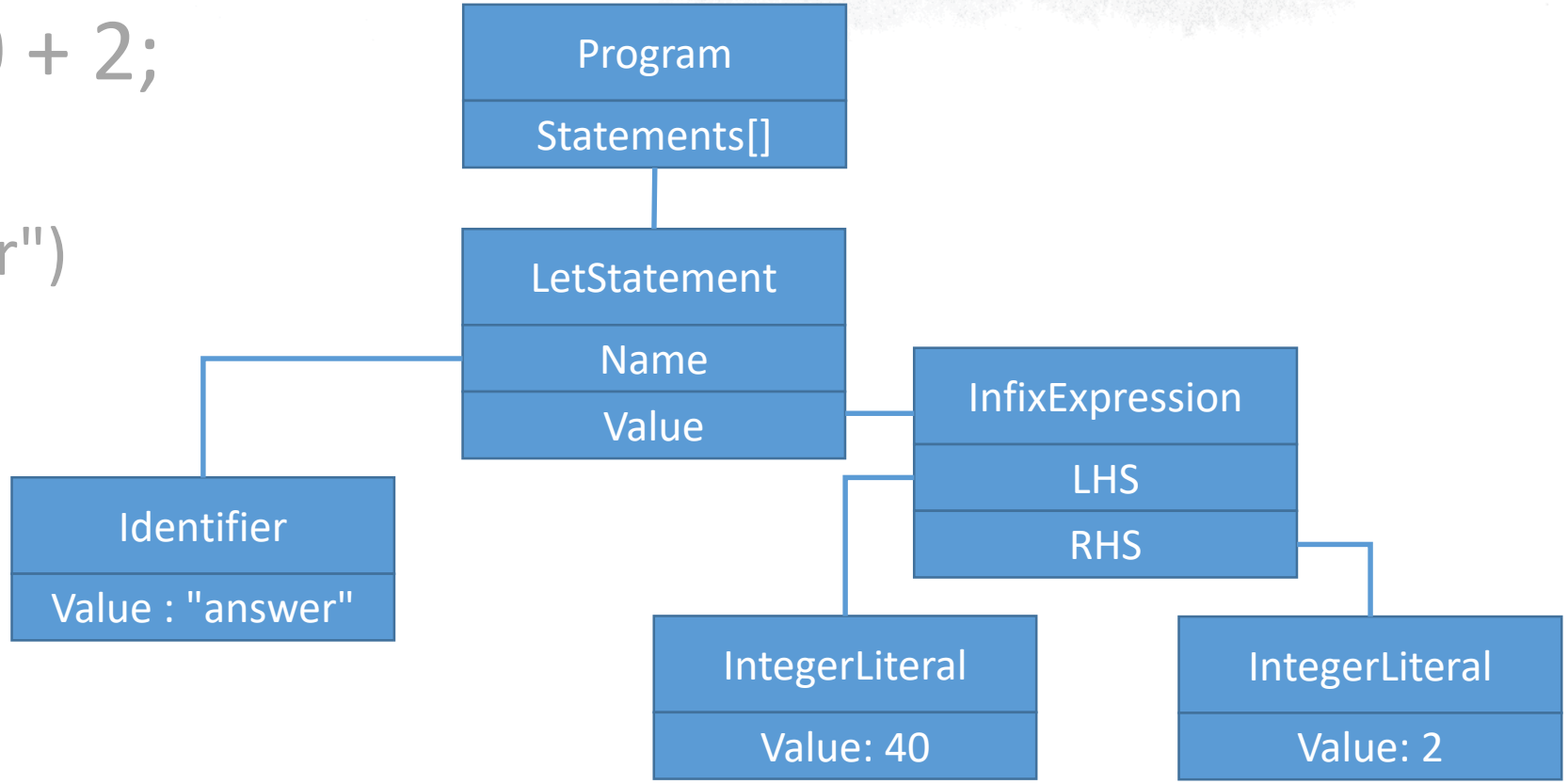


Code Fixes

Abstract Syntax Tree

let answer = 40 + 2;

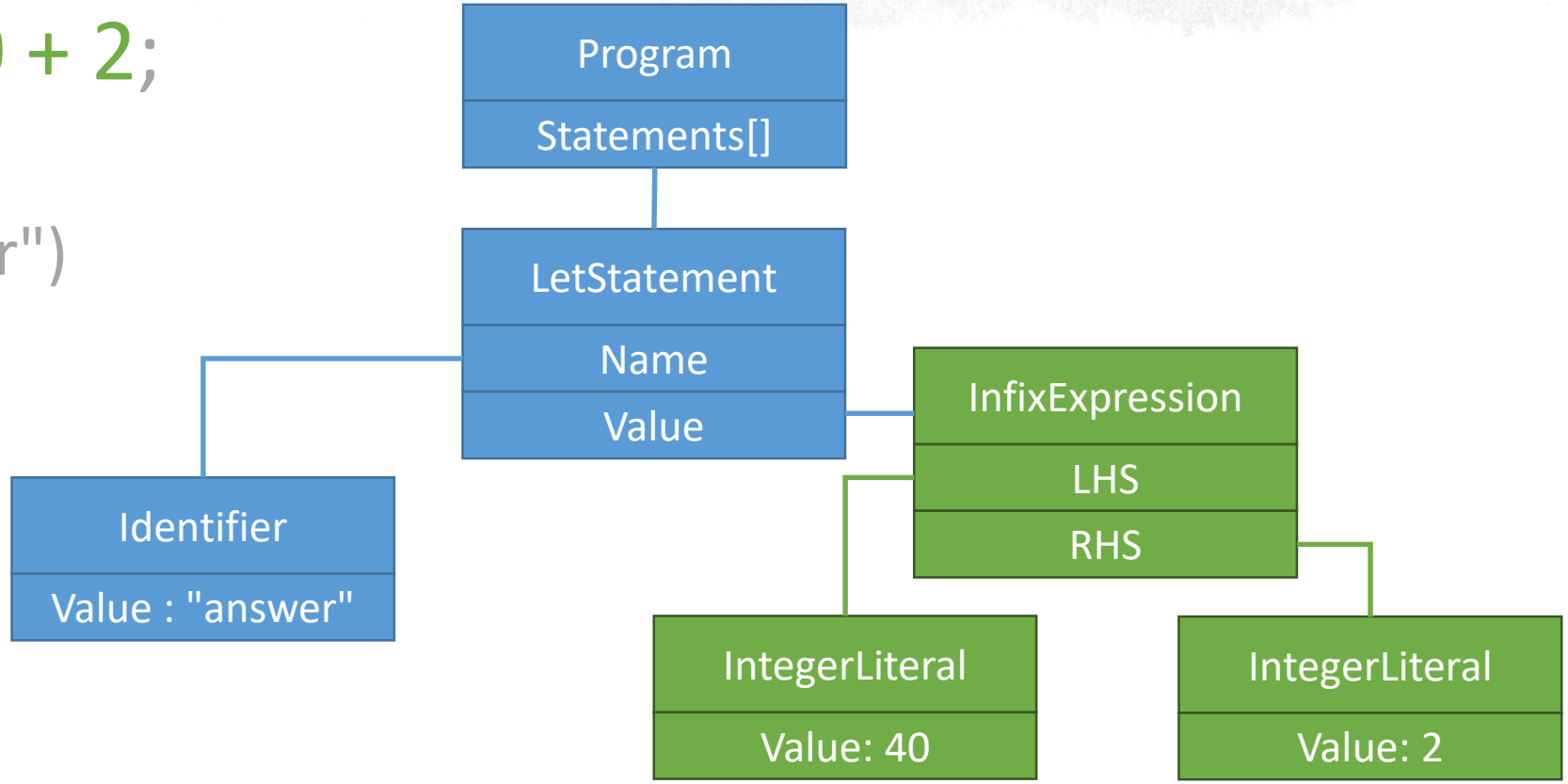
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



Abstract Syntax Tree

let answer = 40 + 2;

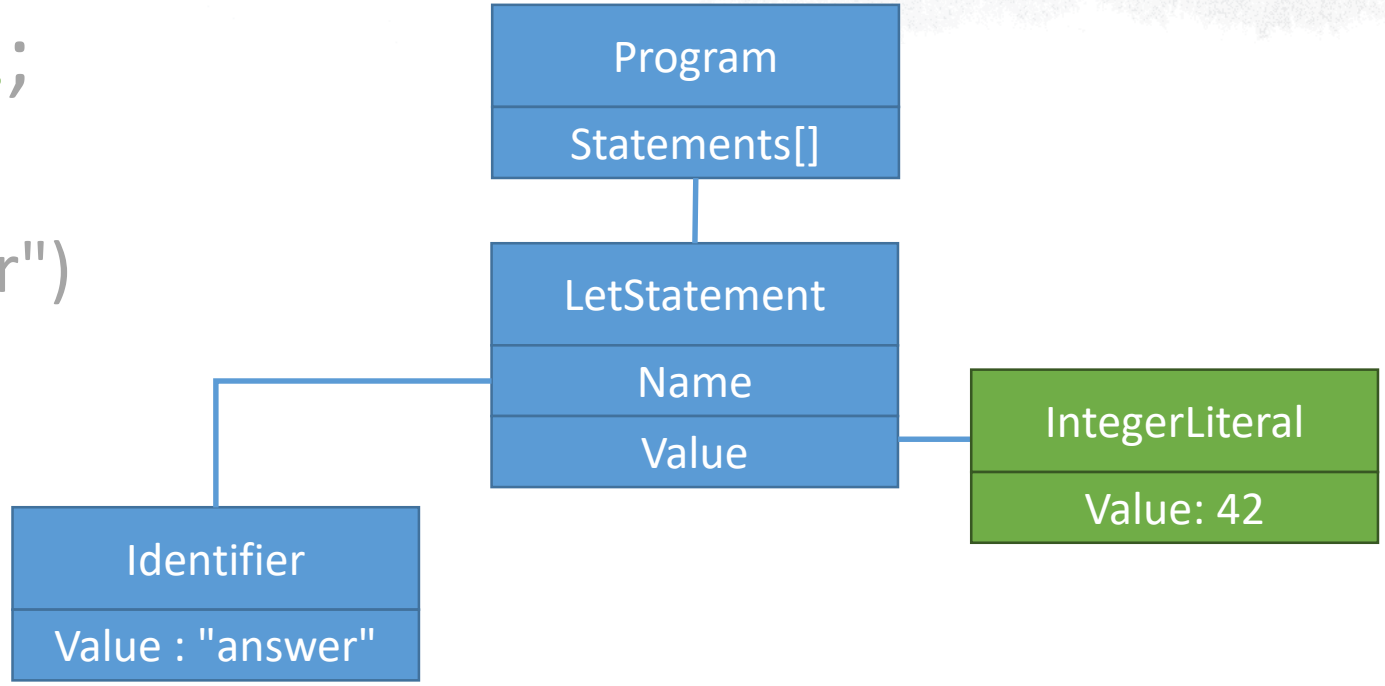
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(40)
PLUS_SIGN
INTEGER(2)
SEMICOLON
EOF



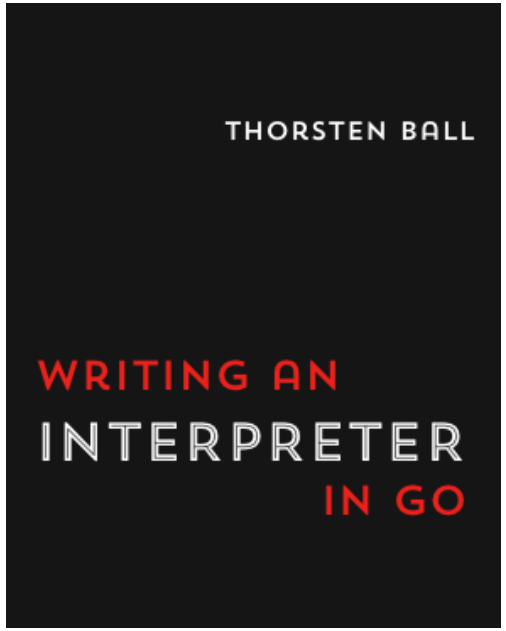
Abstract Syntax Tree

let answer = 42;

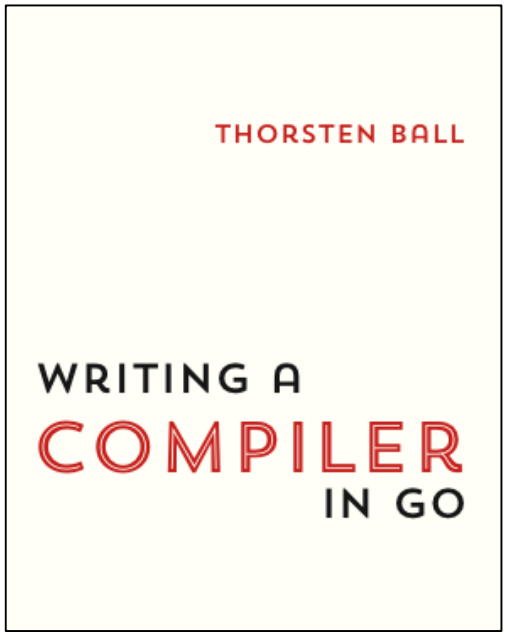
LET
IDENTIFIER("answer")
EQUAL_SIGN
INTEGER(42)
SEMICOLON
EOF



Thank You



interpreterbook.com



compilerbook.com

pluralsight.com