Creating SimpleLanguage Runtime

# Installing the Pieces

To build a C# .NET 8 program that parses and runs a simple language defined in a `.g4` file using ANTLR (ANother Tool for Language Recognition), we'll proceed incrementally. Here's the first phase breakdown:

1. Define the Grammar: Create the `.g4` file to define the language syntax.

2. Set Up the Project: Create a new C# .NET 8 project and configure it to use ANTLR.

3. Generate the Parser and Lexer: Use ANTLR to generate the necessary classes.

4. Implement the Interpreter: Write the code to interpret and execute the parsed language.

## Step 1: Define the Grammar

Create a file named `SimpleLang.g4` with the following content:

```g4

grammar SimpleLang;

prog: stat+ ;

stat: varDecl

| expr

| methodDecl

;

varDecl: type ID ('=' expr)? ';' ;

type: 'int'

| 'string'

| 'int' '[' dimensions ']'

| 'string' '[' dimensions ']'

;

dimensions: INT (',' INT)\* ;

methodDecl: type ID '(' argList? ')' block ;

argList: type ID (',' type ID)\* ;

block: '{' stat\* '}' ;

expr: expr ('\*'|'/') expr

| expr ('+'|'-') expr

| ID

| INT

| STRING

| ID '[' expr ']'

;

ID: [a-zA-Z\_][a-zA-Z\_0-9]\* ;

INT: [0-9]+ ;

STRING: '"' .\*? '"' ;

WS: [ \t\r\n]+ -> skip ;

```

## Step 2: Set Up the Project

1. Create a new .NET 8 Console Application:

```sh

dotnet new console -n SimpleLangInterpreter

cd SimpleLangInterpreter

```

2. Add the ANTLR NuGet package:

```sh

dotnet add package Antlr4.Runtime.Standard

```

3. Create a directory for the grammar file:

```sh

mkdir Grammar

```

4. Place the `SimpleLang.g4` file in the `Grammar` directory.

## Step 3: Generate the Parser and Lexer

1. Install the ANTLR tool:

```sh

dotnet tool install -g Antlr4BuildTasks

```

2. Generate the parser and lexer classes:

```sh

antlr4 -Dlanguage=CSharp -o Generated Grammar/SimpleLang.g4

```

This will generate the parser and lexer classes in the `Generated` directory.

3. Include the generated classes in your project:

Edit your `.csproj` file to include the generated files:

```xml

<ItemGroup>

<Compile Include="Generated\\\*.cs" />

</ItemGroup>

```

## Step 4: Implement the Interpreter

1. Create an interpreter class:

Create a new file named `SimpleLangInterpreter.cs` and implement a basic interpreter:

```csharp

using Antlr4.Runtime;

using Antlr4.Runtime.Tree;

using System;

class SimpleLangInterpreter

{

public static void Main(string[] args)

{

var input = "int x = 5; string y = \"hello\";"; // Example input

var inputStream = new AntlrInputStream(input);

var lexer = new SimpleLangLexer(inputStream);

var commonTokenStream = new CommonTokenStream(lexer);

var parser = new SimpleLangParser(commonTokenStream);

var context = parser.prog();

var visitor = new SimpleLangVisitor();

visitor.Visit(context);

}

}

class SimpleLangVisitor : SimpleLangBaseVisitor<object>

{

public override object VisitVarDecl(SimpleLangParser.VarDeclContext context)

{

var type = context.type().GetText();

var id = context.ID().GetText();

var expr = context.expr() != null ? context.expr().GetText() : "null";

Console.WriteLine($"{type} {id} = {expr}");

return base.VisitVarDecl(context);

}

public override object VisitExpr(SimpleLangParser.ExprContext context)

{

Console.WriteLine($"Expression: {context.GetText()}");

return base.VisitExpr(context);

}

}

```

2. Run the project:

```sh

dotnet run

```

This basic setup allows you to parse and process simple variable declarations and expressions. We'll extend the interpreter to handle arrays and methods in subsequent phases.

**After a runtime error, Jarvis suggested doing this:**

The error you're encountering suggests there might be an issue with the directory structure or the specific version of the tool. Let's address this step by step. Here are a few alternative steps you can follow:

### Step 1: Install ANTLR via NuGet

Instead of using the `antlr4` tool directly, you can install the ANTLR NuGet package and use it within your project:

1. Install the ANTLR4.Runtime.Standard NuGet package:

```sh

dotnet add package Antlr4.Runtime.Standard

```

### Step 2: Use the ANTLR Tool Directly

Download the ANTLR tool manually and use it to generate the parser and lexer classes:

1. Download the ANTLR tool:

Download the ANTLR tool JAR file from the official [ANTLR website](https://www.antlr.org/download.html).

2. Generate the parser and lexer:

Use the downloaded JAR file to generate the parser and lexer classes. Assuming you downloaded `antlr-4.9.3-complete.jar`, you can use the following command:

```sh

java -jar antlr-4.9.3-complete.jar -Dlanguage=CSharp -o Generated Grammar/SimpleLang.g4

```

### Step 3: Include the Generated Files in Your Project

Make sure the generated files are included in your project:

1. Add the generated files:

Copy the generated files into your project directory if they are not already there.

2. Update the .csproj file:

Edit your `.csproj` file to include the generated files:

```xml

<ItemGroup>

<Compile Include="Generated\\\*.cs" />

</ItemGroup>

```

### Step 4: Implement the Interpreter

Follow the steps provided previously to implement the interpreter in C#.

**Example of the Updated .csproj File**

Here is an example of what your `.csproj` file should look like after including the generated files:

```xml

<Project Sdk="Microsoft.NET.Sdk">

<PropertyGroup>

<OutputType>Exe</OutputType>

<TargetFramework>net8.0</TargetFramework>

</PropertyGroup>

<ItemGroup>

<PackageReference Include="Antlr4.Runtime.Standard" Version="4.9.3" />

</ItemGroup>

<ItemGroup>

<Compile Include="Generated\\\*.cs" />

</ItemGroup>

</Project>

```

### Running the Project

After setting up everything, you should be able to run your project without encountering the directory not found error:

```sh

dotnet run

```

Here’s a screenshot of the project along with the NuGet Packages

A screenshot of a computer program

Description automatically generated

# Developing the Code

The workflow is:

* Update the G4 file
* Generate the code
* Build/Run the Interpreter

Under the Grammar folder is a (readme).txt file. Here are the contents:

When the grammar changes, we must modify the .g4 and regenerate all the code

located in the folder 'Generated'

This is done with a command line like this:

Move to where the Grammar folder is and run ANTLR (which was placed within the Grammar folder)

cd c:\Users\dan\_h\source\repos\SimpleLangInterpreter\Grammar

And then move the generated code up the our solutions 'Generated' folder using File Explorer,

thus replacing the files under Generated. Of course, you can organize files as you wish.

I just found this to be the best for me.

The change to .g4 may change what the Visitor code needs to be, so then alter the 'Custom'

visitor code, such as 'SimpleLangCustomVisitor.cs'

\*NEVER\* manually alter the generated code!

java -version => java version "1.8.0\_411"

Java(TM) SE Runtime Environment (build 1.8.0\_411-b09)

From a command line I ran the java program as show above which placed my generated files in a “Generated” folder underneath my Grammar.

Directory of C:\Users\dan\_h\source\repos\SimpleLangInterpreter\Grammar\Generated

05/21/2024 08:44 AM <DIR> .

06/05/2024 05:04 AM <DIR> ..

06/05/2024 05:38 AM 5,709 SimpleLang.interp

06/05/2024 05:38 AM 352 SimpleLang.tokens

06/05/2024 05:38 AM 15,146 SimpleLangBaseListener.cs

06/05/2024 05:38 AM 12,350 SimpleLangBaseVisitor.cs

06/05/2024 05:38 AM 6,743 SimpleLangLexer.cs

06/05/2024 05:38 AM 4,760 SimpleLangLexer.interp

06/05/2024 05:38 AM 352 SimpleLangLexer.tokens

06/05/2024 05:38 AM 11,144 SimpleLangListener.cs

06/05/2024 05:38 AM 50,125 SimpleLangParser.cs

06/05/2024 05:38 AM 7,158 SimpleLangVisitor.cs

10 File(s) 113,839 bytes

2 Dir(s) 138,371,670,016 bytes free

This is my preferred way, as it gives me the opportunity to look at the generated files before moving them up to the Generated folder in my solution.

# Errors and Resolution

Exception in thread "main" java.lang.UnsupportedClassVersionError: org/antlr/v4/Tool has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0

In my case, I had more than one Java installed. Doing a “java -version” revealed version 1.8.0\_411

A screenshot of a computer

Description automatically generated

Restarted the command window and now my version is “22.0.1” (16Apr2024)

Now my command line “java -jar antlr…” worked without error.