COS382 - Project: Domain-Specific Language

Small Groups (2–3); Due: [multiple]

Feb 20	**Assigned**
Mar 3	**Domain and Problem Selected *and* Approved**
Mar 17	**Grammar Completed**
Apr 7	**Functional Parser**
Apr 28	**Functional Translator**
May 5	**Documentation—Complete Project DUE**

Note: This is a major project with several milestones to complete over 2 months, and it counts as 20% of the course grade.

Overview

The purpose of this project is to design and implement a domain-specific language using ANTLR. The process should include:

- a language design
- creation of a grammar to recognize the language
- use of ANTLR to create a language recognizer
- use of ANTLR to create an appropriate translator
- documentation of the language with examples

Selecting a Domain

Domains for this project will come from a wide range of applications and will have some spread in difficulty level. Some examples and concepts are presented below.

In general, the goal of a DSL is to serve as a language for a human to express a problem, a solution, an action, etc., in a natural format. The particular "program" expressed in this language is then checked for correctness of expression (parsed) and given meaning appropriate to the domain. The purpose is to translate input that is typically human-readable into something useful in another context.

In order for this process to occur, a grammar must be created to define the legal sentences of the language. The grammar describes allowed input formats. The actions generated by a particular set of inputs specifies the semantics of the "program" and causes actions to occur. The form of output of programs defined in DSLs will vary greatly.

Possible Domains/Problems

Clearly, the list of possible ideas below is not exhaustive. Feel free to propose other languages. Creation and use of a DSL for a project in another class or a research project is encouraged. Design of a language to cause a physical action with hardware connected to a computer port would be fun.

- Configuration language. Allow expression in "natural terms", produce output in proper format for another program, system, etc.
- A "make" tool.
- Quiz generator. Given a set of questions in proper format with correct answers indicated,

generate a multiple choice style quiz and answer key.
- Survey generator. Parse a file in defined format and generate a *survey monkey* style survey.
- Generate HTML forms based on a simple syntax describing information to be received.
- Generate proper CSS styles based on descriptive input.
- Language to generate test scripts from a natural style input file.
- Mapping business rules to SQL.
- Generate some type of chart used in software engineering or systems design.
- Menu generator.
- Read data in CSV format, extract and manipulate values based on description of data and actions.
- You might be able to convince the professor that an assembler would fit into our definition of DSL.
- Turtle graphics. Interesting problem is how to represent the output: visual, coordinate position after each move, etc.
- Simple drawing language, postscript style?
- Robot control language.
- Some simple natural language text (with restricted grammar).
- GUI designer. Output could be difficult.

It may be possible to translate from a known DSL to meaningful output. However, if this approach is approved, the following conditions will apply:

- A correct ANTLR grammar must be defined.
- It is *not* acceptable to use an available ANTLR grammar.
- The maximum grade for the project is 80% (24).

Requirements

The general approach for creating a DSL should include the following steps.

Important note: Each of these steps must be documented as a `readme.html` in your submission.

1. Project selected and approved
    1. Define the domain.
    2. Define the problem. What is the purpose of the language?
    3. Create example "solutions". What is the expected output for some examples of valid input?
    4. Define functionality. How is the output determined for a given input?
    5. Talk with the professor, and get approval.
2. Grammar completed
    1. Create a grammar for the DSL.
    2. Test example solutions against grammar. Do you have to make corrections or changes?
3. Functional parser
    1. Use ANTLR to generate a parser for the language.
    2. Test the parser for agreement with the design.
4. Functional translator
    1. Use a visitor/listener to generate a translator based on the desired functionality and examples.
    2. Test the translator.
5. Documentation
    1. Write documentation for your project. Use the `readme.html` as a template (similar to a GitHub `readme.md`)
    2. Create a meaningful and non-trivial application example.
    3. Validate the output of your DSL for the application example.
    4. Discuss how the DSL compares to generating the same output with a general-purpose language or by hand.
    5. Clearly indicate which group member is responsible for each portion of the project.

Collaboration

Understand the problem:	Prof, TA, or Partners
Develop a solution:	Prof, TA, or Partners
Implement and debug:	Prof, TA, or Partners

This is a small group project (2 or 3). If you have questions about your general solution or about your particular implementation (e.g., your code), ask the prof or TA!

In other words, *do **not** consult with the internet.*

Readme Write-up

Details about the `readme.html` file are found here. The `readme.html` template is found at the following links: unformatted version and with Markdeep formatting.

Grade Template

3	Domain / Problem Approved with `readme.html` write-up
6	Interesting (not trivial/simple) and Unique Problem
3	Grammar Completed with `readme.html` write-up
3	ANTLR Parser with `readme.html` write-up
6	ANTLR Translator with `readme.html` write-up
6	Documentation with `readme.html` write-up
3	Examples and Testing

Late submissions will *not* be accepted.

Only one submission per group is required.

Submit

Submit the entire project including source code, compiled code, `readme.html` documentation, and evidence of your code working (screen snapshot) via Moodle as a Zip file.