Logo Parser / Translator

BTI7064 – Automata and Formal Languages – Spring Term 2015

# Introduction

We were tasked with writing a parser/translator in javaCC(Logo.jj), which translated/parsed given commands in a “.logo” file, using an EBNF grammar which was provided, into a java class. Which, when executed, drew the described picture in the “.logo” file.

For example:

The “square.logo” file (see figure 1) will be output as beautiful square (see figure 2)

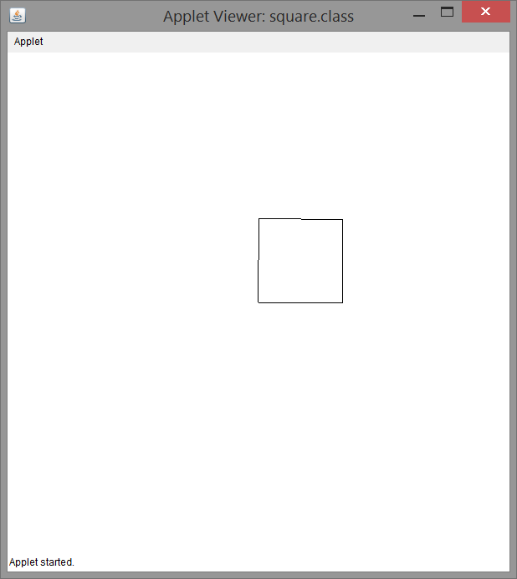


Figure 2: Result of square.logo file

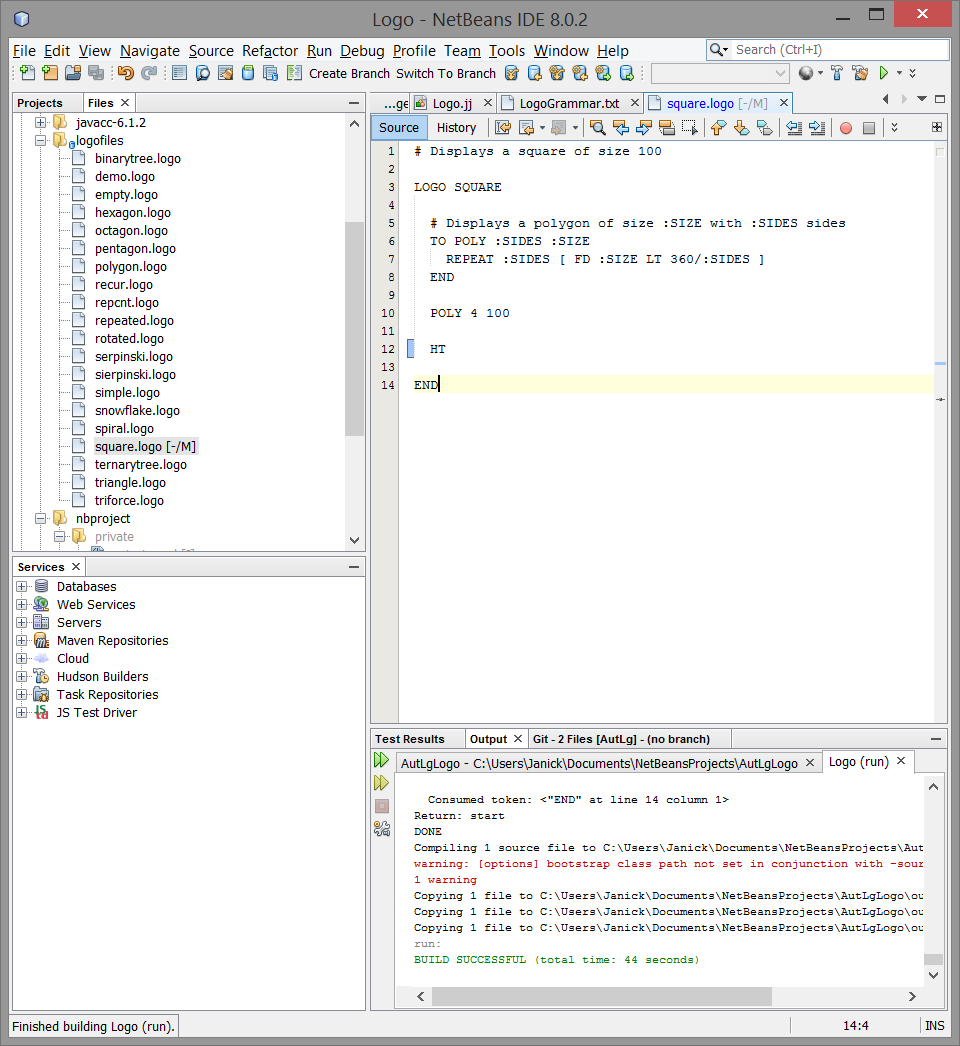


Figure : square.logo file

In order for the parser to successfully interpret and parse the given “.logo” our parser needed to understand the used grammar and commands. Therein lay the challenge of this “mini-project”. For example: Commands like REPEAT :SIDES [ FD :SIZE LT 360/:SIDES ] needed to be understood as well as parsed into a normal javaclass as:

for (int i = 1; i <= sides; i++) {

logo.fd(size);

logo.lt(360 / sides);

}

# Grammar

We slightly altered the provided Grammar to complete our project. The changes that were made are highlighted in bold.

Program = "LOGO" Identifier { Subroutine } { Statement } "END"

Subroutine = "TO" Identifier { Parameter } { Statement } "END"

Statement = "CS" | "PD" | "PU" | "HT" | "ST"

| "FD" NExpr | "BK" NExpr | "LT" NExpr | "RT" NExpr

| "WAIT" NExpr

| "REPEAT" NExpr "[" { Statement } "]"

| "IF" BExpr "[" { Statement } "]"

| "IFELSE" BExpr "[" { Statement } "]" "[" { Statement } "]"

| Identifier { NExpr }

NExpr = NTerm { ( "+" | "-" ) NTerm }

We eliminated the redundancy in the “NFactor” routine and created a second one called “NFactorPos“

NTerm = NFactor { ( "\*" | "/" ) NFactor }

NFactor = "-" NFactorPos | NFactorPos

**NFactorPos = Number | REPCOUNT | Parameter | "(" NExpr ")"**

BExpr = BTerm { "OR" BTerm }

BTerm = BFactor { "AND" BFactor }

BFactor = "TRUE" | "FALSE" | "NOT" "(" BExpr ")"

| NExpr ( "==" | "!=" | "<" | ">" | "<=" | ">=" ) NExpr

Comments start with "#" with scope until the newline

Numbers are real numbers

Identifiers start with a letter followed by letters or digits

Parameters are ":" followed by Identifier

Identifiers, parameters, keywords in uppercase only

# Solution

Description of your solution. How you have managed certain aspects, in particular the translation of the REPCOUNT reserved identifier, the error/warning you might have encountered during the project, etc.

# Test

How you have verified that your solution works well, i.e. the test examples you have used

# Limitations

Description of the limitations of your approach, discuss which software engineering principles studied in the Software Engineering module are not well respected in your project and how you could you improve your solution