Iga Słotwińska

Projekt TKOM

Opis

Projekt ma umożliwiać działania na plikach w formacie svg (grafiki wektorowe). Stworzony język powinien zapewniać możliwość wyszukiwania pewnych obiektów, zapisywania ich jako nowe grafiki i zmieniania ich atrybutów (np. kolor). Jako błędne będą traktowane próby pracowania na plikach innych niż svg, np. xml. Zakładana jest poprawność podawanych atrybutów dla danego elementu.

Wymagania funkcjonalne

1. Wypisanie elementów z dokumentu
    1. Wypisanie wszystkich elementów
    2. Wypisanie elementów spełniających pewne wymogi (filtrowanie)
        1. Według tagu np. <rect …/>
        2. Według atrybutu np. <rect id=„2” …/>
2. Wypisanie atrybutów elementów z dokumentu
    1. Wszystkich elementów
    2. Przefiltrowanej grupy elementów (według tagu lub atrybutu)
3. Zmiana atrybutu danego elementu
    1. Zmiana wartości atrybutu
    2. Dodanie nowego atrybutu
    3. Usunięcie istniejącego atrybutu
4. Zapisanie elementów do pliku svg
    1. Utworzenie nowego pliku
    2. Nadpisanie istniejącego dokumentu

Tokeny

(,) , ” , { , } , , , ; , if , else , while , . , = , == , < , > , <= , >= , != , + , - , * , / , && , || , val, element , list

Operacje

- save(element, path-to-file) // stworzenie lub dodanie elementu do pliku
- save(list, path-to-file) // stworzenie lub dodanie elementu do pliku
- read(path-to-file) // wczytanie elementów pliku do kolekcji
- create() //stworzenie pustej kolekcji
- create(element) // stworzenie jednoelementowej kolekcji
- create(list) // stworzenie kolekcji z obiektami z istniejącej kolekcji
- create(tag-name) // stworzenie elementu o danym tagu
- (list).add(element) // dodanie elementu do kolekcji
- (list).add(list) // dodanie grupy elementów do kolekcji
- (list).remove(element) // usunięcie elementu z kolekcji
- (list).remove(list) // usunięcie grupy elementów z kolekcji
- (list).filterByTag(tag-name) // zwrócenie nowej przefiltrowanej kolekcji z zadanym tagiem
- (list).filterByAttr(attribute-name, value) // zwrócenie nowej przefiltrowanej kolekcji z zadanym atrybutem
- (element).setAttr(attribute-name, value) // ustawienie wartości atrybutu dla danego elementu
- (element).deleteAttr(attribute-name) // usunięcie atrybutu dla danego elementu
- (element).getAttr(attribute-name) // zwrócenie wartości atrybutu
- print(list) // wypisanie elementów w liście
- print(element) // wypisanie atrybutów elementu
- (list).size() // wypisanie wielkości listy

Budowa projektu

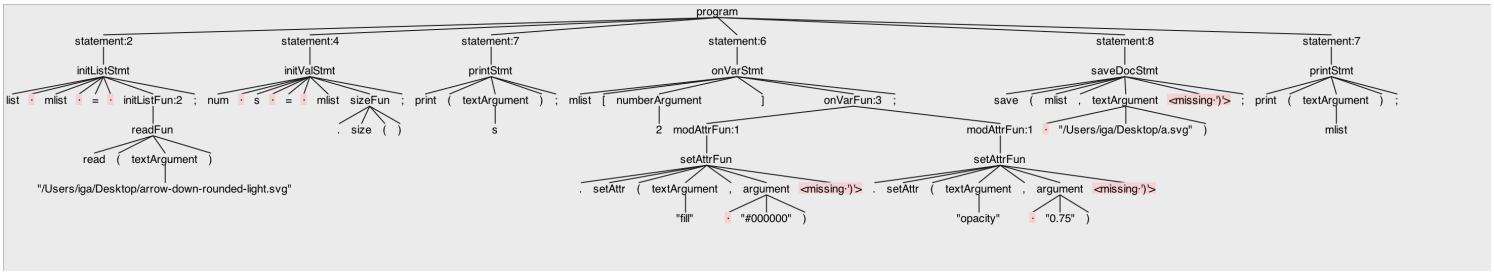
Projekt został napisany w języku Java z użyciem ANLTR, dzięki któremu zostały wygenerowane klasy leksera oraz parsera (na podstawie napisanej przeze mnie gramatyki). Logika interpretera jest zaimplementowana w klasie Visitor, która dziedziczy po klasie GramBaseVisitor. Parser przechodzi pomiędzy węzłami drzewa wyprowadzenia w sposób zdefiniowany w klasie Visitor. Zmienne przechowywane są w liście i dodatkowo nazwy zmiennych zadeklarowanych w danym bloku są przechowywane na stosie. Zmienne są instancjami klasy Variable, która trzyma nazwę zmiennej, jej typ oraz wartość. Do obsługi plików svg używana jest klasa MetaSVG, która zawiera metody do tworzenia, zapisywania, wczytywania i edycji plików svg. Do wczytywania skryptów służy klasa ScriptHandler, która czyta za pomocą Scannera standardowe wejście.

Proces analizy i wykonywania skryptów będzie odbywał się w następujących etapach:
1. Obsługa źródeł - wczytanie pliku ze skryptem
2. Analiza leksykalna - rozbicie tekstu wejściowego na tokeny, przekazywanie rozpoznanego tokenu do parsera
3. Analiza składniowa - sprawdzenie poprawności przychodzących tokenów ze zdefiniowaną gramatyką, tworzenie drzewa rozbioru
4. Analiza semantyczna - sprawdzenie poprawności programu utworzonego przez analizator składniowy drzewa i jego instrukcji

Dodatkowo, zdefiniowane zostaną moduły do obsługi błędów, które mogą występować na każdym z wyżej wymienionych etapów oraz moduł do parsowania plików svg.
Program może przyjąć na wejście plik tekstowy ze skryptem.
W celu sprawdzenia poprawności działania programu zostaną stworzone także testy automatyczne w JUnit 4. Na potrzeby demonstracji projektu logowane jest przechodzenie kolejnych węzłów drzewa wyprowadzenia.

Przykład z drzewem rozbioru i wynikiem

```
list mlist = read(„/Users/iga/Desktop/arrow-down-rounded-light.svg");
num s = mlist.size();
print(s);
mlist[2].setAttr("fill", "#000000").setAttr("opacity", „0.75");
save(mlist, „/Users/iga/Desktop/a.svg");
print(mlist);
```




```
00:03:56.357 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Starting program.
00:03:56.360 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting statement.
00:03:56.360 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Declaring list variable.
00:03:56.365 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Declaring list variable (function).
00:03:56.365 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Reading list from document (function).
00:03:56.605 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting statement.
00:03:56.605 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Declaring string or num variable.
00:03:56.606 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting statement.
00:03:56.606 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Printing.
4.0
00:03:56.609 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting statement.
00:03:56.610 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting function on variable statement.
00:03:56.611 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting argument (function).
00:03:56.613 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting argument (function).
00:03:56.614 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting statement.
00:03:56.616 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Saving document.
00:03:56.683 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Visiting statement.
00:03:56.683 [main] INFO  islotwin.tkom.gen.GramBaseVisitor - Printing.
LIST:
ID: 0 TAG: style ATTRS: {type: text/css; xml:space: preserve; }
ID: 1 TAG: title ATTRS: {}
ID: 2 TAG: path ATTRS: {fill: #000000; d: M0,0h24v24H0V0z; opacity: 0.75; }
ID: 3 TAG: polyline ATTRS: {class: st1; points: 20.8,8.6 11.8,15.4 2.7,8.6 ; }
```

Wczytany plik -> plik wynikowy

Gramatyka

```
// Parser Rules
program: (statement)+ ;
statement: ifStmt | initListStmt | initElemStmt | initValStmt | mathStmt | onVarStmt | printStmt | saveDocStmt
| whileStmt;
ifStmt: IF STARTNBRACKET expression ENDNBRACKET STARTCBRACKET (statement)+ ENDCBRACKET (
elseFun )? ;
elseFun: ELSE STARTCBRACKET (statement)+ ENDCBRACKET;
whileStmt: WHILE STARTNBRACKET expression ENDNBRACKET STARTCBRACKET (statement)+
ENDCBRACKET ;
mathStmt: VARNAME ASSIGNOP ((numberArgument MATHOP numberArgument) | argument)
SEMICOLON ;
saveDocStmt: SAVE STARTNBRACKET VARNAME COMMA textArgument ENDNBRACKET SEMICOLON ;
printStmt: PRINT STARTNBRACKET textArgument ENDNBRACKET SEMICOLON ;
initListStmt: LISTVAR VARNAME ASSIGNOP ( initListFun | VARNAME ) ( ( modListFun )+ | ( filterFun )+ )?
SEMICOLON ;
onVarStmt: VARNAME (STARTSBRACKET numberArgument ENDSBRACKET)? onVarFun SEMICOLON ;
onVarFun: ( modListFun )+ | ( filterFun )+ | ( modAttrFun )+ | sizeFun ;
initListFun: createFun | readFun ;
createFun: CREATE STARTNBRACKET ( VARNAME )? ENDNBRACKET ;
readFun: READ STARTNBRACKET textArgument ENDNBRACKET ;
filterFun: filterAttrFun | filterTagFun ;
filterAttrFun: EXOP FILTERBYATTR STARTNBRACKET textArgument COMMA argument ENDNBRACKET ;
filterTagFun: EXOP FILTERBYTAG STARTNBRACKET textArgument ENDNBRACKET ;
modListFun: EXOP MODLISTOP STARTNBRACKET VARNAME ENDNBRACKET ;
sizeFun : EXOP SIZE STARTNBRACKET ENDNBRACKET;
initElemStmt: ELEMVAR VARNAME ASSIGNOP  initElemFun ( setAttrFun )* SEMICOLON ;
initElemFun: createElemFun | getElemFun ;
createElemFun: CREATE STARTNBRACKET textArgument ENDNBRACKET ;
getElemFun: VARNAME STARTSBRACKET numberArgument ENDSBRACKET ;
modAttrFun: setAttrFun | deleteAttrFun ;
setAttrFun: EXOP SETATTR STARTNBRACKET textArgument COMMA argument ENDNBRACKET ;
deleteAttrFun: EXOP DELETEATTR STARTNBRACKET textArgument ENDNBRACKET ;
initValStmt :
   ( NUMBERVAR | STRINGVAR) VARNAME ASSIGNOP (
       argument
      | ( numberArgument MATHOP numberArgument )
      | VARNAME sizeFun
   ) SEMICOLON
         ;
expression: numberArgument EXPRESSIONOP numberArgument ( ANDOP numberArgument
EXPRESSIONOP numberArgument )* ;
textArgument: VARNAME | TEXT ;
```

```
numberArgument: VARNAME | NUMBER ;
argument: VARNAME | TEXT | NUMBER ;

// Lexer Rules
IF: 'if';
ELSE: 'else';
WHILE: 'while';
STARTCBRACKET: '{';
ENDCBRACKET: '}';
STARTNBRACKET: '(';
ENDNBRACKET: ')';
STARTSBRACKET: '[';
ENDSBRACKET: ']';
ASSIGNOP: '=';
EXPRESSIONOP: '<' | '>' | '<=' | '>=' | '==' | '!=';
MATHOP: '+' | '-' | '*' | '/' ;
ANDOP: '&&' | '||';
MODLISTOP: 'add' | 'remove' ;
EXOP: '.';
COMMA: ',';
SEMICOLON: ';';
GETATTR: 'getAttr';
SETATTR: 'setAttr';
DELETEATTR: 'deleteAttr';
SAVE: 'save';
READ: 'read';
PRINT: 'print';
CREATE: 'create';
SIZE: 'size';
FILTERBYATTR: 'filterByAttr';
FILTERBYTAG: 'filterByTag';
NUMBERVAR: 'num';
STRINGVAR: 'string';
LISTVAR: 'list';
ELEMVAR: 'elem' ;
NUMBER: ('0'|'-'?[1-9][0-9]*);
VARNAME: [a-zA-Z]([a-zA-Z]|NUMBER)*;
TEXT: '"'.+?'"' ;

WHITESPACE :  (' ' | '\t' | '\r' | '\n') {skip();};
```