

## 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 zadaniem tagiem
- (list).filterByAttr(attribute-name, value) // zwrócenie nowej przefiltrowanej kolekcji z zadaniem 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.

## Testowanie

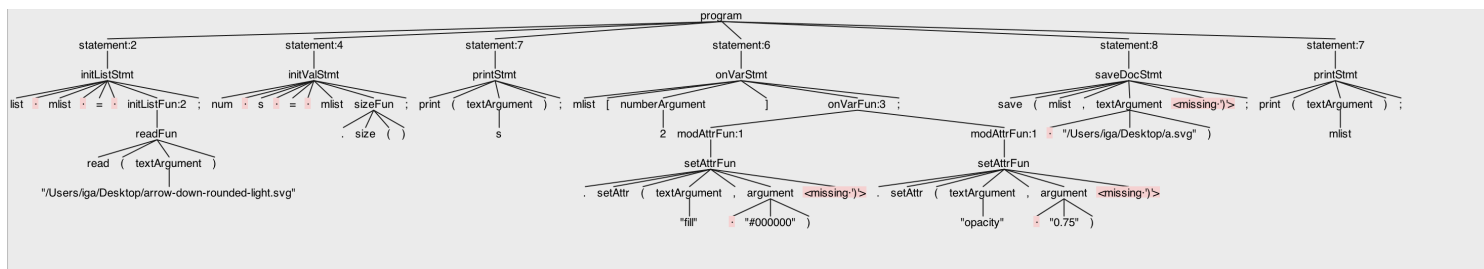
W celu sprawdzenia poprawności działania programu zostaną stworzone także testy automatyczne w JUnit 4. Na potrzeby demonstracji projektu logowane jest przechodzenie kolejnych węzłów drzewa wyprowadzenia. Testowany jest lekser, gdzie sprawdzana jest poprawność drzewa rozbioru. Sprawdzane są pętle, wyrażenia warunkowe oraz deklaracje zmiennych. W testach parsera sprawdzana jest poprawność typów zmiennych, przykładowo czy w odpowiednim miejscu rzucany jest wyjątek InvalidVariableType, czy zmienne mają unikalne nazwy oraz czy zostały zadeklarowane wcześniej w danym bloku, jeśli z nich korzystamy.

## Podsumowanie

Wszystkie założenia projektu i wymagania funkcjonalne zostały zaimplementowane, a także sama obsługa plików svg. Jedynie zmianie podległa gramatyka, gdzie dodane zostały typy zmiennych num oraz string, określające czy zmienna jest typu liczbowego czy tekstowego. Dzięki temu łatwiejsze jest sprawdzanie typów w funkcjach.

## 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);
```



```
00:03:56.613 [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  
wynikowy

plik ->



## Gramatyka

// Parser Rules

program: (statement)+ ;

statement: ifStmt | initListStmt | initElemStmt | initValStmt | mathStmt | onVarStmt | printStmt | saveDocStmt | whileStmt;

ifStmt: IF STARTNBACKET expression ENDNBACKET STARTCBACKET (statement)+ ENDCBACKET (elseFun)? ;

elseFun: ELSE STARTCBACKET (statement)+ ENDCBACKET;

whileStmt: WHILE STARTNBACKET expression ENDNBACKET STARTCBACKET (statement)+ ENDCBACKET ;

mathStmt: VARNAME ASSIGNOP ((numberArgument MATHOP numberArgument) | argument) SEMICOLON ;

saveDocStmt: SAVE STARTNBACKET VARNAME COMMA textArgument ENDNBACKET SEMICOLON ;

printStmt: PRINT STARTNBACKET textArgument ENDNBACKET SEMICOLON ;

initListStmt: LISTVAR VARNAME ASSIGNOP ( initListFun | VARNAME ) ( ( modListFun )+ | ( filterFun )+ )? SEMICOLON ;

onVarStmt: VARNAME (STARTSBACKET numberArgument ENDSBACKET)? onVarFun SEMICOLON ;

onVarFun: ( modListFun )+ | ( filterFun )+ | ( modAttrFun )+ | sizeFun ;

initListFun: createFun | readFun ;

createFun: CREATE STARTNBACKET ( VARNAME )? ENDNBACKET ;

readFun: READ STARTNBACKET textArgument ENDNBACKET ;

filterFun: filterAttrFun | filterTagFun ;

filterAttrFun: EXOP FILTERBYATTR STARTNBACKET textArgument COMMA argument ENDNBACKET ;

filterTagFun: EXOP FILTERBYTAG STARTNBACKET textArgument ENDNBACKET ;

modListFun: EXOP MODLISTOP STARTNBACKET VARNAME ENDNBACKET ;

sizeFun: EXOP SIZE STARTNBACKET ENDNBACKET;

initElemStmt: ELEMVAR VARNAME ASSIGNOP initElemFun ( setAttrFun )\* SEMICOLON ;

initElemFun: createElemFun | getElemFun ;

createElemFun: CREATE STARTNBACKET textArgument ENDNBACKET ;

```

getElemFun: VARNAME STARTSBRACKET numberArgument ENDSBRACKET ;
modAttrFun: setAttrFun | deleteAttrFun ;
setAttrFun: EXOP SETATTR STARTNBRACKET textArgument COMMA argument ENDNBRACKET ;
deleteAttrFun: EXOP DELETEATTR STARTNBRACKET textArgument ENDNBRACKET ;
initValStmnt :
    ( NUMBERTVAR | 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';
NUMBERTVAR: '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' ) {skip0};

```