**Smart compiler**

**Compiler Construction**

Course Supervisor: Ms. Maryam Feroze

**B16101116-Muhammad Talib Waseem**

**B16101043-Hafiz Alqama Bin Shuja**

**Introduction:**

Smart Compiler is a simple compiler that will be made on javascript as backend. The compiler Language contain all the main concept of today’s famous programming languages.

**Feature:**

1. Int, float, char, string, bool, const (immutable), let, var
2. ArrayList (LinkedList), Stack, Queue, 1D-Array
3. Function, Higher Order Function (JS feature)
4. Logical Operation (if-else)
5. No terminator (optional)
6. No garbage, unassigned variable can’t be used.
7. No public, private all attribute will be private and function are public.
8. Classes, Object Oriented Programming
9. Operator Overloading, function overloading
10. Inheritance (single inheritance)
11. Only Public Inheritance
12. Protected can be used to access base class property to child
13. Spread Operator ( … )
14. Switch case
15. All 3 loop with range loop
16. Ternary Operator ( ? : )
17. Static variable, function

**Word Table:**

|  |  |
| --- | --- |
| **Words** | **Class** |
| int | Datatype |
| float | Datatype |
| string | Datatype |
| short | Datatype |
| long | Datatype |
| double | Datatype |
| const | Datatype |
| var | Datatype |
| while | while |
| do | Do |
| for | for |
| if | If |
| else | else |
| switch | switch |
| case | case |
| break | break |
| continue | continue |
| true | true |
| false | false |
| class | Class |
| protected | protected |
| static | static |
| arrayList | arrayList |
| Stack | Stack |
| Queue | Queue |
| function | function |
| return | return |

**Punctuators:**

1. , use to separate parameter, object attribute
2. ; statement terminator
3. : for inheritance
4. { } for object structure
5. ( ) for function
6. [ ] for 1D,2D array
7. . to access attribute of object

**Operators:**

AOP 🡺 = += -= \*= /= %=

ADDOP 🡺 + -

MOP 🡺 \*

DOP 🡺 / %

ROP 🡺 < > <= >= != ==

LAND 🡺 &&

LOR 🡺 ||

BAND 🡺 &

BOR 🡺 |

**Identifiers:**

∑ = { alphabet, digit, \_ }

alphabet = [ A - Z ], [ a – z ]

digit = [ 0 – 9 ]

1. ID start with either \_ or alphabet
2. First char can be followed by any char.

RE = (\_+ alphabet (^ + \_ + alphabet + digit

**Constants:**

1. Integer Constant:

∑ = { digit, +, - }

RE = ( + | - | ^)(digit

1. Float Constants**:**

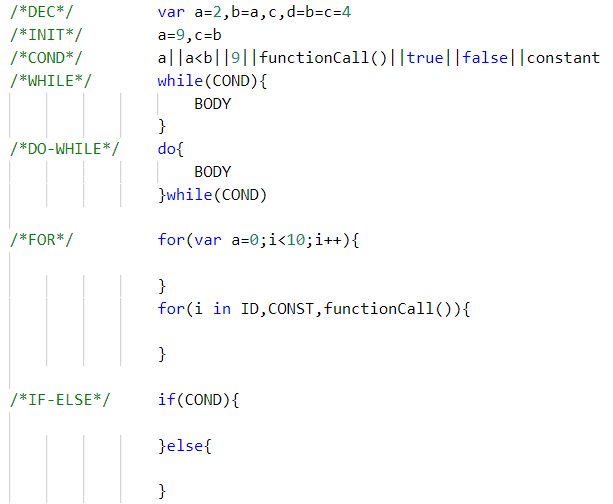
∑ = { +, -, . , digit }

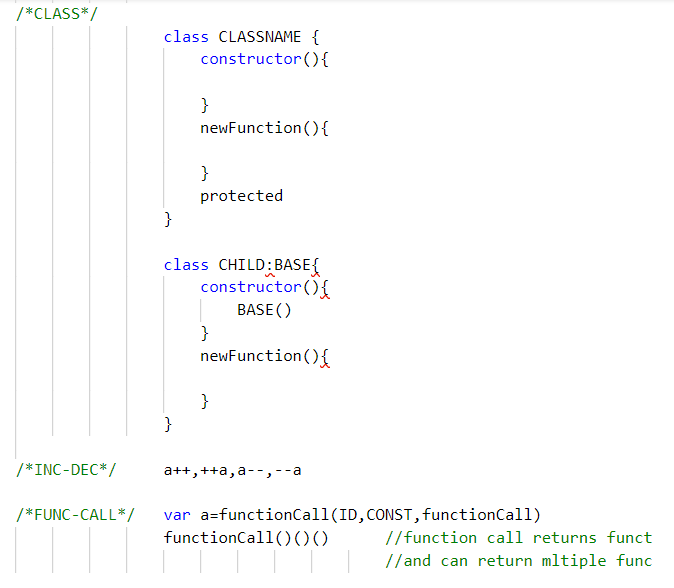
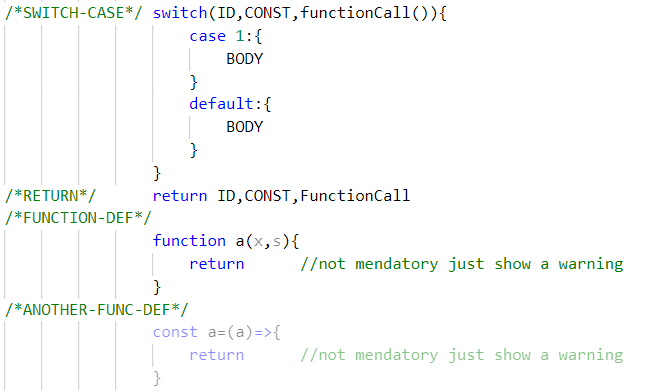
RE = ( + | - | +| ^ )(digit)\* . (digit

1. String Constant:

RE = “ [ (C+A) + (\(A+B)) ] ”

**Basic Syntax:**

****

****

**CFG’s**

1. **<DEC>** 🡪 DT ID <DEC1><NEXTDEC>
   1. <DEC1> 🡪 AOR <DEC2> | €
   2. <DEC2> 🡪 ID **<INIT-VALUE>** <DEC1> |

**<INIT-VALUE-2>**

* 1. <NEXTDEC> 🡪 , ID <DEC1><NEXTDEC>|; |€
  2. **<INIT-VALUE>** 🡪 <N-ID>
     1. <N-ID> 🡪 € |. ID < INIT-VALUE> |

(**<CALLING-PARAMS>**) <N-ID> |

[ID <INIT-VALUE>] <N-ID>

1. **<INC-DEC-POST>** 🡪 <GTSWID>

<INC-DEC-ID-HANDLER>

* 1. <inc-dec-op>🡪 ++ | --
  2. <INC-DEC-ID-HANDLER> 🡪 **<INIT-VALUE>**

<inc-dec-op>

1. **<INC-DEC-PRE>** 🡪 <inc-dec-op> ID **<INIT-VALUE>**
2. **<CALLING-PARAMS>** 🡪 <CP-VALUE>
   1. <CP-VALUE> 🡪 € | <CP-VALUE2>
   2. <CP-VALUE2> 🡪 ID <INIT-VALUE>

<NEXT-CPVALUE>

* 1. <NEXT-CPVALUE> 🡪 € |, <CP-VALUE2>

1. **<CONST-DT>** 🡪 const ID <CONST-DT1>

<NEXT-CONST-DT>

* 1. <CONST-DT1> 🡪 AOR <CONST-DT2>
  2. <CONST-DT2> 🡪 new ID ( **<DEC-PARAM>** ) | **<INIT-VALUE-2> |** ID **<INIT-VALUE>**

1. **<DEC-PARAM>** 🡪**<SIMPLE-INIT>** <NEXT-DEC-PARAM>
   1. <NEXT-DEC-PARAM>🡪, <DEC-PARAM> | €

7. **<FUNCTION-DEC>** 🡪 function <FUNC-DEF-1>

7.1. <FUNC-DEF-1>🡪 ID **<DEC-PARAM> <BODY>**

8. **<INIT>** 🡪<GTSWID>

8.2. <INIT\_ID\_HANDLER>🡪<INIT-VALUE> AOR <VALUE>

8.1. <VALUE>🡪 ID <INIT-VALUE> <INIT-2> | <CONST>| <OBJECT>

8.2. <INIT-2>🡪 AOR <VALUE> | €

9**.<WHILE>** 🡪 while (<**COND>**) **<BODY>**

10.**<COND>** 🡪 <ID-CONST> <COND1>

10.2. <COND1>🡪 € | ROP <ID-CONST>

10.1. <ID-CONST>🡪 ID **<INIT-VALUE>**|**<CONST>**

11. **<BODY>** 🡪 ; | <SST> | {<MST>}

11.1. <SST> 🡪 <DEC> | <WHILE> | <FOR> | <DO-WHILE> | <FUNCTION>| <CONST-DT> | ~~<INIT> | < INC-DEC-POST> | <FUNTION-CALL>~~ ID <GTSWID> | <INC-DEC-PRE> | <IF-ELSE> | <SWITCH-CASE> | <RETURN>

12. **<FOR>** 🡪 for (<FOR-PARAM>) <BODY>

12.1. <FOR-PARAM> 🡪 ID <FOR-PARAM-2> | <DEC>;<C2>;<C3> |; <C2>;<C3>

12.2. <FOR-PARAM-2> 🡪 <INIT>;<C2>;<C3> | <RANGE-FOR>

12.3. <RANGE-FOR> 🡪 in ID <INIT-VALUE>

12.4. <C2>🡪<COND> | €

12.5. <C3> 🡪 € | <INIT> | **<INC-DEC-POST> |**

**<INC-DEC-PRE>**

1. **<DO-WHILE>**🡪do <BODY> while ( <COND> )
2. **<IF-ELSE>** 🡪 if ( <COND> ) <BODY> <ELSE>
   1. <ELSE>🡪 € | else <BODY>
3. **<CLASS>** 🡪 class ID <CLASS-STRUCT>
   1. <CLASS-STRUCT> 🡪 <CLASS-BODY> | :ID <CLASSBODY>
   2. <CLASSBODY>🡪 { <CLASS-ST> <CLASS-MST> }
   3. <CLASS-ST> 🡪 <CLASS-FUNC> | <DEC> | <PROTECTED> | €
   4. <CLASS-MST> 🡪 € | <CLASS-ST> <CLASS-MST>
   5. <CLASS-FUNC> 🡪 ID **<DEC-PARAM> <BODY>**
   6. <PROTECTED>🡪{ <PRO-BODY>} | <PRO-BODY>
   7. <PRO-BODY>🡪 <CLASS-FUNC> |<DEC>
4. **<TERNARY>** 🡪 **<COND>**? **<BODY>**: **<BODY>**

1. **<OBJECT>** 🡪 {<PROP>}
   1. <PROP>🡪 <OB JECT1> <NEXT-PROP>
   2. <OBJECT1> 🡪 ID:<ATTR-VALUE> | <SPREAD>
   3. <NEXT-PROP>🡪, <PROP>| €
   4. <SPREAD> 🡪 … ID <INIT-VALUE>

1. **<ARRAY>** 🡪 [ <ARRAY-INNER>] <NEXT-ARRAY>
   1. <ARRAY-INNER> 🡪 € | <INIT-VALUE> | <ARRAY-VALUES>
   2. <ARRAY-VALUES> 🡪<ARRAY-VAL><NEXT-VAL>
   3. <ARRAY-VAL> 🡪 **<INIT-VALUE-2>** | <INIT-VALUE> | … ID <INIT-VALUE>
   4. <NEXT-VAL>🡪 , <ARRAY-VALUES>
   5. <NEXT-ARRAY>🡪 € | [ID <INIT-VALUE>]

1. **<INIT-VALUE-2> 🡪** <CONST> | <ARRAY> | <OBJECT>
2. **<SIMPLE-INIT> 🡪** ID <S-I-1>
   1. <S-I-1> 🡪 AOR <D-P-VALUE1> |€
   2. <D-P-VALUE1> 🡪 ID **<INIT-VALUE>** <D-P-VALUE2>| **<INIT-VALUE-2>**
   3. <D-P-VALUE2> 🡪 AOR < D-P-VALUE1>|

1. **<FUNTION-CALL>** 🡪 <GTSWID>
   1. **<FUN-C-P> 🡪** ( **<CALLING-PARAMS>**)

<NEXT-FUN-CP>

* 1. <NEXT-FUN-CP> 🡪 € |**<FUN-C-P>**

1. **<SWITCH-CASE>** 🡪 switch (<OPT>) <SW-BODY>
   1. <OPT> 🡪 <INIT-VALUE>
   2. <SW-BODY> 🡪 {<CASES> <DEFAULT>}
   3. <CASES>🡪 case <CASE-VALUE>: <BODY>
   4. <CASE-VALUE> 🡪 <INNER-CASE-VAL> | (<INNER-CASE-VAL>)
   5. <INNER-CASE-VAL> 🡪 **<INIT-VALUE>** |**<INIT-VALUE-2>**
   6. <DEFAULT> 🡪 default : <BODY>
2. <GTSWID> 🡪 <TG>
   1. <TG> 🡪 <INIT\_ID\_HANDLER>| **<INC-DEC-POST>** | <FUN-C-P>

|  |  |  |
| --- | --- | --- |
| # | Non-Terminals | First Set |
| 1 | DEC | * DT |
| 2 | DEC1 | * € * AOR |
| 3 | DEC2 | * ID * Fof (EXP)= |
| 4 | DEC3 | * Fof (MERGEDINIT)={ € . ( [ } |
| 5 | MERGED \_INIT | * € * Fof(M\_O\_V) = { . ( } * [ |
| 6 | M\_O\_V | * . * ( |
| 7 | MERGED | * Fof (OP) = {+ - \* / || && % } * Fof (MERGED\_INIT) = { € . ( [ } |
| 8 | OP | * + * - * \* * / * || * && * ! * % |
| 9 | MER-N-ARR |  |
| 10 |  |  |
|  |  |  |
|  |  |  |