# SYNTAX ANALYZER

- **START:**
  **<s>** → <C_I>class Main { Main() {<non_func_MST>} }<C_I>
  **<C_I>** → <class><C_I> | <interface><C_I> | €

- **CLASS:**
  **<class>**→AM <sealed> class ID <inherit_interface> {<non_class_MST>}
  **<sealed>**→sealed|€
  **<inherit_interface>**→: ID <multi_interface>| €
  **<multi_interface>**→,ID<multi_interface>|€

- **INTERFACE:**
  **<interface>**→ interface ID { <func_interface> }
  **<func_interface>**→<ret_T> ID (<parameter>); <func_interface>|€

- **DECLARATION:**
  **<list>** → <list1> | = <L'>
  **<list1>** → ; | ,ID<list>
  **<L'>** →<OE> <list>

- **ARRAY:**
  **<A'>** → ,<OE>] = {<2D_cond>}; | ] = {<condition>};
  **<2D_cond>**→{<condition>}<2D_cond1>
  **<2D_cond1>**→,<2D_cond> | €

- **OBJECT:**
  **<object>** → ID ID = new ID(<condition>);

- **FUNCTION:**
  - **FUNCTION DEFINITION:**
    **<ret_T>** → void | DT<const_array>

    **<const_array>** →<C'>|€
    **<C'>** →[ <C">

**<C”>** → ,] | ]

- **LOOPS:**
  - **FOR LOOP:**
    **<for_loop>** → for(<c1><c2>;<c3>){<non_func_MST>}
    **<c1>** → DT ID <list> | <c3>;
    **<c2>** → <OE> | €
    **<c3>** → <Asg_Op_loop> |€
  - **WHILE LOOP:**
    **<while_loop>**→while(<OE>){<non_func_MST>}

    ➢ **Assignment Operator For Loop**

    **<Asg_Op_loop>** → this. <Asg_Op’> | <Asg_Op’> | DI ID <ID_array> <AsgOp_Comma>
    **<Asg_Op’>** → ID<ID><ID_array> <Asg_Op’’>
    **<Asg_Op’’>** → , <Asg_Op_loop><Asg_Op_loop1>| <Asg_Op_loop1> | DI <AsgOp_Comma>

    **<Asg_Op_loop1>** → <AsgOp_Equal> <OE><Asg_Op_loop1><AsgOp_Comma>|€
    **<AsgOp_Comma>** → ,<Asg_Op_loop> |€
    **<AsgOp_Equal>** → AsgOp | equal

- **IF-ELSE:**
  **<if-else>**→ if(<OE>){<non_func_MST>}<else>
  **<else>** → else <else_if> | €
  **<else_if>** → {<non_func_MST>} | <if-else>

- **TRY , CATCH , FINALLY:**
  **<TCF>** → try{ <non_func_MST> }catch {<non_func_MST>} finally {<non_func_MST>}

- **OBJECT:**
  **<object>** → ID ID = new ID(<condition>);

- **EXPRESSION:**

  **<OE>**→<A_OE><OE’>

<OE'>→OR<A_OE><OE'>|€
<A_OE>→<R_OE><A_OE'>
<A_OE'>→AND <R_OE><A_OE'>|€
<R_OE>→<E_OE><R_OE'>
<R_OE'>→ROP<E_OE><R_OE'>|€
<E_OE>→<T_OE><E_OE'>
<E_OE'>→PM<T_OE><E_OE'>|€
<T_OE>→<F_OE><T_OE'>
<T_OE'>→MDM<F_OE><T_OE'>|€
<F_OE>→<const>|!(<OE>)|DI ID<ID_array>| this. <F'>|ID <F''>
<F'>→ID<F''>
<F''>→.<F'> | [<OE><arr_call>] <Dec_inc> | DI| €

- **GENERALIZED NON-TERMINAL:**

<Dec_inc> → DI | €

<ID_array> → [<OE><arr_call>] | €
<arr_call>→,<OE>|€
<return> → return <return'>
<return'>->OE>; | €

<base>→:base(<argument>)|€
<condition> → <argument> | €
<argument> → <OE><a>
<a> →,<argument> | €

<parameter>→DT ID<add_para> |€
<add_para>→,<parameter> | €

<ID>→.ID<ID>|€

<non_func_MST> → <SST_F><non_func_MST> | €
<SST_F> →DT ID <SST_F''> | this. <SST_F1'>|ID <SST_F1_obj> | <for_loop>
| <while_loop> | <if-else> | <TCF> | DI ID <ID_array> | break; |
continue; | <return>

<SST_F''>→<list>|[<OE><A'>
<SST_F1'>→ID<ID><SST_F1''>
<SST_F1_obj>-> ID = new ID(<condition>);| <ID><SST_F1''>

<SST_F1’’>→[<OE><arr_call>] <ID_arr>|  <Asg_Op_MST>  | DI;
|(<condition>);
<ID_arr> → <Asg_Op_MST> | DI ;
<Asg_Op_MST> →  <AsgOp_Equal> <OE> <Asg_Op_MST>|;

<non_class_MST> → <SST_C><non_class_MST> | null
<SST_C>→AM <SST_C’>|<object>
<SST_C’> → Static DT ID <list_attributes>|DT < SST_C’’>| void ID
(<parameter>){<non_func_MST>}| ID (<parameter>)<base>{<MST_constructor>}
< SST_C’’>->  ID<SST_C’’’>|<C’ > ID (<parameter>){<non_func_MST>}
< SST_C’’’>-><SST_F’’>|(<parameter>){<non_func_MST>}

<MST_constructor> → <MST_constructor’><MST_constructor>|null
<MST_constructor’> →DT ID <SST_F’’> | this. <SST_F1’>|ID <SST_F1_obj> |
<for_loop> | <while_loop> | <if-else> | <TCF> | DI ID <ID_array>|return;

<list_attributes> → <list_attributes1>| = <L_A’>
<list_attributes1> → ; | ,ID<list_attributes>
<L_A’>→<OE_class_MST> <list_attributes>


<OE_class_MST>→<A_OE_class_MST ><OE_class_MST’>
<OE_class_MST’>→OR<A_OE_class_MST ><OE_class_MST’>|€
<A_OE_class_MST >→<R_OE_class_MST ><A_OE_class_MST’>
<A_OE_class_MST’>→AND <R_OE_class_MST ><A_OE_class_MST’>|€
<R_OE_class_MST >→<E_OE_class_MST ><R_OE_class_MST’>
<R_OE_class_MST’>→ROP<E_OE_class_MST ><R_OE_class_MST’>|€
<E_OE_class_MST >→<T_OE_class_MST ><E_OE_class_MST’>
<E_OE_class_MST’>→PM<T_OE_class_MST ><E_OE_class_MST’>|€
<T_OE_class_MST >→<F_OE_class_MST ><T_OE_class_MST’>
<T_OE_class_MST’>→MDM<F_OE_class_MST ><T_OE_class_MST’>|€
<F_OE_class_MST >→<const>|!(<OE_class_MST >)