**Start:**

<S> 🡪 <defst> { <class\_body\_main> <class\_body>} <defen>

<static\_abstract> 🡪 static | abstract |final|e

**Definition:**

<defst>🡪 public <defst1> <defst>|<defst1> <defst>| class ID

<defst1>🡪 static class ID <inheritance> { <class\_body> } | <static\_abstract> <defst1’’>

<defst1’>🡪 <inheritance> { <class\_body> }| ε

<defst1’’>🡪class ID <defst1’>

<defen>🡪 public <defen1> <defen> | <defen1> <defen> | ε

<defen1>🡪 <c\_modifier> class ID <inheritance> { <class\_body> }

<inheritance>🡪 extends ID <inh> | ε

<inh>🡪 ,ID|ε

**Class Body:**

<class\_body\_main> 🡪<c\_modifier> <class\_body2> | static <main1> |

<class\_body>🡪 <data\_type> ID <class\_body1><class\_body> | <c\_modifier>

<class\_body2> 🡪| <acc\_modifier1> <c\_modifier> class ID <inheritance>

{<class\_body> } | ID <class\_body1> <class\_body> | ε

<class\_body1>🡪 <initial> <list> | (<PL>) <body>}

<class\_body2>🡪 void <class\_body1> | class ID <inheritance> { <class\_body> }

<main1>🡪 void <main2> | class ID <inheritance> { <class\_body> }

<main2>🡪 Main { <body> } | <class\_body1> <main1>

**Body:**

<body>🡪 ; | {<MSt>}

<MST>🡪 <SSt><MST> | ε

**Multiline Statement:** <MSt>🡪ε| <SSt><MSt>

**Access Modifiers:**

<acc\_modifier>🡪 public | private | protected | ε

**Class Modifiers:**

<c\_modifier>🡪 static | abstract | final | ε

**Return Type:**

<ret\_type>🡪 void | <data\_type>

**Function:**

<function>🡪 static <ret\_type> ID ( <PL> ) <body>

**Parameters:**

<PL>🡪<OE> <PL2> |ε <PL2>🡪, <OE><PL1> |

**This:**

<this>🡪this .| ε

**Expression:**

<OE>🡪 <AE> <OE’>

<OE’>🡪 || <AE> <OE’> | ε <AE>🡪 <RE> <AE’>

<AE’>🡪 && <RE><AE’> | ε

<RE>🡪 <E> <RE’>

<RE’>🡪 ROP <E> <RE’> | ε

<E>🡪 <T> <E’>

<E’>🡪 PM <T> <E’> | ε

<T>🡪 <F> <T’>

<T’>🡪 MDM <F> <T’> | ε

<F>🡪 ID <F’>| <const> | (<OE>) | !<F> | <inc\_dec> <this> ID <Z2> |this.ID <Z>

;

<F’>🡪 <Z>; | ε

**While:**

<while>🡪 while (<OE>) <body1>

**Do While:**

<do\_while>🡪 do { <;> <MSt> <;> } while ( <OE> )

<;>🡪 ; <;> | ε

**If Else:**

<if\_else>🡪 if (<OE>) <body1> <else>

<else> 🡪 else <body1> | **ε**

**For:**

<for>🡪 for ( <c1> <c2>; <c3> ) <body1>

|  |  |
| --- | --- |
| <c1>🡪 | ; | <declare> | <assign\_st> |
| <c2>🡪 | <OE> | ε |
| <c3>🡪 | <this> ID <X>| <inc\_dec> <this>ID <X1> | ε |

<X1>🡪 . ID<X> | [intconst].ID <X> |(<PL>).ID<X>

<X>🡪 . ID<X> | <assign\_op> <OE> | [intconst].ID <X> |(<PL>).ID<X> | <inc\_dec>

**Constant:**

<const>🡪 int\_const | string\_const | double\_const

**Declaration:**

<declare>🡪 <data\_type> ID <initial> <list>

<initial>🡪 = <OE> | ε

<list>🡪 ; | , ID <initial> <list>

**Single Line Statement:**

<SSt>🡪 <while> | <for> | <if\_else>| <do\_while> ;| continue ; | break ;| return <OE> ;|<function>} | ID <obj>; | this.ID<K>;| <inc\_dec> <this> ID <Z2> ; | <data\_type> <1d- Array>

<obj>🡪 ID =new <c\_name> () | <Z>

<K>🡪 <Z2> | = <OE>

<Z>🡪 .ID <R1> | [ int\_const ].ID<R1> | (<PL>) <R2> | <X’>

<R1> 🡪.ID <Z> | [ int\_const ] .ID <Z> | (<PL>) .ID <Z> | ε

<R2> 🡪.ID <B> | ε

<B> 🡪 <Z2> | <X’>

<X’> 🡪 <assign\_op> <OE> | <inc\_dec>

<Z2> 🡪 .ID <R1’> | [int\_const] .ID <R1’> | (<PL>) .ID <R1’> | ε

<R1’> 🡪 .ID <Z2> | [int\_const] .ID <Z2> | (<PL>) .ID <Z2> | ε

**Object Declaration:**

<obj-dec>🡪 ID ID = new ID (<OE>);

**Assignment:**

<assign\_st>🡪 <this> ID <X1> <initial> <list>

<assign\_op>🡪 = | <c\_assign>

<c\_assign>🡪 += | -= | \*= | /=

<this> 🡪this.|super.|ε

<X>🡪 . ID<X> | <assign\_op> <OE> | [intconst].ID <X> |(<PL>).ID<X> |<inc\_dec>

**Increment Decrement:**

<inc\_dec> 🡪 ++ | --

**1d- Array:**

<1d- Array> 🡪 <data\_type> <ID> [<arr\_size>] <stm\_tr>

<arr\_size> 🡪 int\_const

<stm\_tr> 🡪 ;|, <ID> [<arr\_size>] <stm\_tr>