|  |  |  |
| --- | --- | --- |
| **CFGS** | **FIRST SET** | **FOLLOW SET** |
| **START:**  <start> -> <defs> | Final , class , interface,abstract, E | {EOF} |
| **DEF:**  <defs> -> <class\_def> <defs>  <defs> -> <interface\_def> <defs>  <defs> -> <abs\_class\_def> <defs>  <defs> -> E | final , class ,interface , abstract ,E | {EOF} |
| **CLASS DEFINITION**  <class\_def> -> class ID <INH> { <CB> }  <class\_def> -> final class ID <INH> { <CB> } | final , class | Final , class , interface  ,abstract , E , EOF |
| **INTERFACE:**  <interface\_def> -> interface ID <int\_INH>{ <int\_B> } | Interface | Final , class , interface ,  abstract , E , EOF |
| **ABSTRACT CLASS:**  <abs\_class\_def> -> abstract class ID <INH> {  <abs\_B> } | abstract | Final , class , interface ,  abstract , E , EOF |
| <NAMF> -> final  <NAMF> -> E | final , E | Class , abstract , DT , def, |
| **INHERITENCE:**  <INH> -> extends ID <imp>  <INH> -> E  <INH> -> <imp> | extends , E , implements | { |
| **CLASS BODY**  <CB> -> <assign\_st> <CB>  <CB> -> E  <CB> -> <Tassign> <CB>  <CB> -> <AM> <NAMS> <NAMF> <MVC> <CB> | final , static , basic , DT , ID ,E , this , super , public , private , def | } |
| **EXTENDS:**  <int\_INH> -> extends ID <id Dash>  <int\_INH> -> E | extends , E | { |
| **INTERFACE BODY**  <int\_B> -> <int\_var> <int\_B>  <int\_B> -> <abs\_func\_def> <int\_B> <int\_B> -> E | DT , abstract , E | } |
| **ABSTRACT CLASS BODY**  <abs\_B> -> <assign\_def> <abs\_B>  <abs\_B> -> E  <abs\_B> -> <Tassign\_st> <abs\_B> | final , static , basics ,abstract ,public , def , E , private | } |

|  |  |  |
| --- | --- | --- |
| <abs\_B> -> <AM> <NAMS> <NAMF> <MVA>  <abs\_B> |  |  |
| I**MPLEMENTS**  <imp> -> implements ID <id Dash> <imp> -> E | Implements , E | { |
| **FUNCTION DEFINITION**  <func\_def> -> basic <RDT> def ID <Param> {  <MST> } | basic | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| **CONSTRUCTOR**  <construct\_def> -> def ID (<Param>) { <MST> } | def | Final , static , basic  ,abstract , public , def , E , private , final , static  ,basics,ID,DT,E,this,super,p ublic,private ,def , } |
| **DECLARATIOIN**  <Decl> -> DT ID <AR> < int1> <list1> | DT | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| **ASSIGNMENT:**  <assign\_st> -> ID <LAS> | ID | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| **BODY**  <R> -> (<args>) <FNA>  <R> -> [<exp>] <D2DA> <ref> <EEXP>  <R> -> . ID <R>  <R> -> = <exp>;  <R> -> inc\_dec | ( , [ , , , = , + + , | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| <FNA> -> ;  <FNA> -> . ID <R> | ; , . | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| <D2DA> -> [<exp>]  <D2DA> -> E | [ , E | . , = |
| <Ref> -> . ID <R> | . , E | = |

|  |  |  |
| --- | --- | --- |
| <Ref> -> E |  |  |
| <EEXP> -> = <exp> ; | = | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| <LAS> -> = <exp> ;  <LAS> -> . ID <R> <LAS> -> inc\_dec  <LAS> -> ID <obj R>  <LAS> -> (<args>) <FNA>  <LAS> -> [<exp>] <D>DA> <Ref> <EEXP> | = , . , ID ( , [ , + + , - - | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| <obj R> -> = new ID (<args>) ;  <obj R> -> ; | ; , = | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| <id Dash> -> , ID <id Dash>  <id Dash> -> E | , , E | { |
|  |  |  |
| <abs\_func\_def> -> abstract <RTD> def ID (<Param>)  ; | abstract | DT , abstract , } |
| <m StringConst> -> , StringConst <m StringConst>  <m StringConst> -> E | , , E | } |
| **RETURN DATA TYPE:**  <RTD> -> void  <RTD> -> DT <AN>  <RTD> -> ID | void , DT , ID | def |
| <AN> -> [ ]  <AN> -> E | [ , E | def |
| **PARAMETER**  <Param> -> <dec> <Param1>  <Param> -> <obj> <Param1>  <Param> -> E | DT , ID , E | ) |
| <Param1> -> , <dec\_obj> <Param1>  <Param1> -> E | , , E | ) |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| <dec> -> DT ID <AN> | DT | , , ) |
| <obj> -> ID ID | ID | , , ) |
| <dec\_obj> -> <dec>  <dec\_obj> -> <obj> | DT , ID | , , ) |
| **MST**  <MST> -> <SST> <MST>  <MST> -> E | E , while , break , for  ,if , continue  , return , ID , DT ,try | } |
| <MAS> -> (<arg>) <Ref1>  <MAS> -> [<exp>] <D2DA> <Ref1>  <MAS> -> . ID <MAS>  <MAS> -> E  <MAS> -> inc\_dec | ( , E , , , E | [ , = , ; , ) |
| <Ref1> -> . ID <MAS>  <Ref1> -> E | . , E | [ , = , = , , |
| **ARRAY:**  <AR> -> [ <expr> ] <D2DB>  <AR> -> E | [ , E | = , ; , , |
| <init1> -> = <init1>  <init1> -> E | = , E | ; , ; |
| <list1> -> ;  <list1> -> , ID <AR> <init1> <list1> | ; , , | Final , static , basic , abstract , public, def , E , private , final , static ,basic , ID , DT , E ,this , super ,  public , private , def, } |
| <init D> -> <exp> <init1>  <init D> -> [ <args> ] | ID , | ; , , |
| <args> -> <exp> <argsD>  <args> -> E | ( , ! , ID , const | ) |
| <argsD> -> , <exp> <argsD>  <argsD> -> E | , , E | ) |
| <exp> -> <AND OP> <expD> | ( , ! , ID , const | ; , ) , ] |
| <expD> -> ll <AND OP> <expD> | ll , E | ; , ) , ] |

|  |  |  |
| --- | --- | --- |
| -> E |  |  |
| <AND OP> -> <ROPOP> <ANDOPD> | ( , ! , ID , const | ll , ; ) , ] |
| <ANDOPD> -> && <ROPOP> <ANDOPD>    <ANDOPD> -> E | & & , E | ll , : , ) , ] |
| <ROPOP> -> <E> <ROPOPD>      <ROPOPD> -> ROP <E> <ROPRPD>  <ROPOPD> -> E | ( , ! , ID , const      ROP , E | && , ll , ; ) , ]      && , ll , ; ) , ] |
| <E> -> <T> <ED> | ( , ! , ID , const | ROP , && , ll , ; , ) , ] |
| <ED> -> PM <T> <ED>  <ED> -> E | PM , E | ROP , && , ll , ; , ) , ] |
| <T> -> <F> <TD> | ( ,! ,ID , const | PM , ROP , && ll , ; , ) , ] |
| <TD> -> MDM <F> <TD>  <TD> -> E | MDM , E | PM , ROP , && ll , ; ) , ] |
| <F> -> (<exp>)  <F> -> <const>  <F> -> ! <F>  <F> -> <TRAS>  <F> -> [ <args> ]  <F> -> <inc\_dec> | ( , ! , ID , const , [ ,+  + , - - | MDM , PM , ROP , && ,  ll , ) , ; , ] |
| <SST> -> <white\_st>  <SST> -> <break\_st>  <SST> -> <for\_st>  <SST> -> <if\_elif\_else>  <SST> -> <continue\_st>  <SST> -> <return\_st>  <SST> -> <assign\_st>  <SST> -> <decl>  <SST> -> <try\_catch\_st>  <SST> -> Tassign\_st> | While , break , for ,  If ,, continue ,  Return , ID , DT , try | while , break , for ,if , continue , return ,  ID , DT ,try , } |
| <while\_st> -> while (<exp>) { <MST> } | While | while , break , for ,if , continue , return , ID , DT ,try , } |

|  |  |  |  |
| --- | --- | --- | --- |
| <for\_st> -> for (<P1> <P2> ; <P3>) { <MST> } | For | while , break , for ,if , continue , return , ID , DT ,try , } | |
| <P1> -> decl  <P1> -> assign\_st  <P1> -> ;  <P1> -> <Tassign\_st> | DT , ID , ; | - | ID , ( , ! , const ,++ ,  - , ) |
| <P2> -> <exp>  <P2> -> E | ID , const , ( , ! | ; |  |
| <P3> -> <for\_assign>  <P3> -> <inc\_dec>  <P3> -> E | ID , ++ , - - , E | ) |  |
| <for\_assign> -> <TSD> ID <VAS> | ID | ) |  |
| <VAS> -> = <exp>  <VAS> -> . ID <for\_B>  <VAS> -> inc\_dec | = , . , + + , - - | ) |  |
| <for\_B> -> (<argS>) <FNB>  <for\_B> -> [<exp>] <D2DA> <Ref2> <ETEXP>  <for\_B> -> . ID <for\_B>  <for\_B> -> = <exp>  <for\_B> -> inc\_dec | ( , [ , . , = , ++ , - - | ) |  |
| <FNB> -> . ID <for\_B> | . | ) |  |
| <Ref2> -> . ID <for\_B>  <Ref2> -> E | . , E | = |  |
| <ETEXP> -> = <exp> | = | , , ) | |
| <continue-st> -> continue ; | Continue | while , break , for ,if , continue , return , ID , DT ,try , } | |
| <break\_st> -> break ; | Break | while , break , for ,if ,, continue , return , ID , DT ,try , } | |
| <return\_st> -> return <expN> ; | return | while , break , for ,if , continue , return , ID , DT ,try , } | |
| <expN> -> <exp>  <expN> -> E | ( , ! , ID , const , E | ; | |
| <if\_else\_elif> -> if (<exp>) { <MST> } <elif\_st>  <else\_st> | If | while , break , for ,if , continue , return , ID , DT ,try , } | |
| <elif\_st> -> elif (<exp>) { <MST> } <elif\_st>  <elif\_st> -> E | elif | else , } | |
| <else\_st> -> else : { <MST> }  <else\_st> -> E | else | } | |
| <try\_catch\_st> -> try { <MST> } <OTC> | try | } , while , break , for ,if , continue , return , ID , DT ,try | |
| <OTC> -> catch (exception ID) { <MST> } <OTFD>  <OTC> -> <OTF> | catch | } , while , break , for ,if , continue , return , ID , DT ,try | |
| <OTF> -> finally { <MST> } | finally | } , while , break , for ,if , continue , return , ID , DT ,try | |
| <OTFD> -> finally { <MST> }  <OTFD> -> E | finally , E | } , while , break , for ,if , continue , return , ID , DT ,try | |

|  |  |  |
| --- | --- | --- |
| <Tassign> -> <TS> <ADT> | This , super | while , break , for ,if , continue , return , ID , DT ,try , final , static , basics ,abstract ,public ,  def , E , private |
| <TS> -> this  <TS> -> super | This , super | . |
| <ADT> -> . ID <R>  <ADT> -> ( <args> ) ; | . , ( | while , break , for ,if , continue , return , ID , DT ,try , final , static , basics ,abstract ,public ,  def , E , private |
| <TRAS> -> <TSD> ID <MAS>  <TRAS> -> ID <MAS> | This , super , ID | ROP ,&& ,ll , ; , ) |
| <TSD> -> <TS> | This , super | ID |

|  |  |  |
| --- | --- | --- |
| <inc\_dec> -> indDec ID | + + , - - | ) , ROP , && ,ll , ; |
| <int\_var> -> public static final  <int\_var> -> DT ID = <exp> ; | public | DT , abstract , E |
| <MVA> -> <func\_def>  <MVA> -> <decl>  <MVA> -> <abs\_func\_def>  <MVA> -> <constract\_def> | Basic , DT , abstract , def | final , static , basics ,abstract ,public ,  def , E , private |
| <MVC> -> <func\_def>  <MVC> -> <decl>  <MVC> -> <construct\_def> | Basic , DT , def | final , static , basic , DT , ID ,E , this , super , public , private , def |
| <AM> -> public  <AM> -> private  <AM> -> E | Public , private , E | Static ,final ,Basic , DT , abstract , def |
| <NAMS> -> static  <NAMS> -> E | Static , E | Basic , DT , abstract , def , final |
| <inc\_decT> -> incDec ID ; | ++ , - - | While , break , for ,if , continue ,  return , ID , DT ,try |
| <D2DB> -> [ <expN> ]  <D2DB> -> E | [ , E | = , ; , , |
| <BAS> -> ( <args> ) <Ref1>  <BAS> -> [ <exp> ] <D2DA>  <BAS> -> . ID <BAS>  <BAS> -> E  <BAS> -> inc\_dec | ( , [ , . ++ , - - , E | ROP , && , ll , ; , ) |
| < if\_elif\_else> -> if ( <exp> ) { <MST> } <elif\_st>  <else\_st> | If | while , break , for ,if , continue ,  return , ID , DT ,try |
| <elif\_st> -> elif ( <exp> ) { <MST> } <elif\_st>  <elif\_st> -> E | Elif , E | while , break , for ,if , continue ,  return , ID , DT ,try |
| <else\_st> -> else : { <MST> }  <else\_st> -> E | Else , E | while , break , for ,if ,, continue , return , ID , DT ,try |
|  |  |  |