|  |  |  |
| --- | --- | --- |
| **CFG** | **First Set** | **Follow Set** |
| *<S0> 🡪 <S>* | **First(S0) = { while, for, return, if, break, continue, static, DT, func, ID, final, abstract, class, }** | **Follow (S0) = {$}** |
| *<S> 🡪<SST> <S> | <fdouble> <S2> |* | First (S) = first (SST) U first(fdouble)-{} U first(S2) | {}  = {while, return, if, break, continue, ID} U{ final, class, abstract, static, DT, func}  **First(S) ={while, return, if, break, continue, Id, static, DT, func, abstract, final, class,  }** | **Follow(S) = follow(S0) = {$}** |
| ***<S2> 🡪 abstract<anew> | <class\_def><S>|<fdec> <S> |<func\_st><S>| Static <Snew>*** | **First(S2) = {abstract, class, DT, func, Static}** |  |
| ***<Snew> 🡪 <func\_st> <S> | <fdec> <S>***  <fdec> 🡪 DT ID <declaration>  <declaration> 🡪 <Dec> | <array\_dec>| <dictionary> | **First(Snew) = {func, DT}** |  |
| ***<anew> 🡪 <class\_def> <S> | Static <func\_st> <S> | <func\_st><S>*** | **First(anew) = {class, Static, func}** |  |
| **VARIABLE DECLARATION:** |  |  |
| <Dec> 🡪 <init> <lst> | **first(Dec) = {=, ; , ‘,’}** |  |
| <lst> 🡪 ;  <lst> 🡪 , ID <init><lst> | First(lst) = {;}  First(lst) = {,}  **First(lst) = {; , ,}** |  |
| <init> 🡪   <init> 🡪 = <initE> | First(init) = {}  First(init) = {=}  **First (init) = {, =}** | **Follow(init) = {; , ’,’}** |
| <initE> 🡪 ID<new>|<TS> ID <LHP> <TDASH> <EDASH>| <const> <TDASH> <EDASH> | !<F> <TDASH> <EDASH> | (<E> ) <TDASH> <EDASH> | **First(initE) = {self, super, const, ( , ! , ID}** |  |
| <new> 🡪 <init> | <LHP><TDASH><EDASH> | **First(new) ={ =, [ ,{, ( , . , }** | **Follow(new) = {; , ,}** |
| **WHILE LOOP**: |  |  |
| <while\_st> 🡪 while ( <OE> ) {<body>} | **First(while\_st) = {while}** |  |
| <brk> 🡪 break; | **First(brk) = {break}** |  |
| <cont> 🡪 continue; | **First(cont) = {continue}** |  |
| **VAR and OBJ ASSIGNMENT STATEMENT:** |  |  |
| <asgn\_st> 🡪 <asgn\_op> <asg> | First(asgn\_st) = first(asgn\_op)  **First (asgn\_st) = {= , CompAsgnOP }** |  |
| <asg> 🡪<E>; | <object\_call> | **First(asg) = {self, super, const, ( , ! , ID, obj}** |  |
| <asgn\_op> 🡪 =  <asgn\_op> 🡪 CompAsgnOp | First(asgn\_op) = {=}  First(asgn\_op) = {CompAsgnOP}  **First (asgn\_op) = {= , CompAsgnOP }** |  |
| **IF ELSE CONDITION**: |  |  |
| <if\_else\_st> 🡪 if ( <cond>) {<body>} <op\_else> | **First (if\_else\_st) = {if}** |  |
| <op\_else> 🡪   <op\_else> 🡪 else{ <body>}  <op\_else>🡪 else <if\_else\_st> | First(op\_else) = {}  First (op\_else) = {else}  **First(op\_else) = {  , else, if}** | **Follow(op\_else) = follow (if\_else\_st) = follow (SST) = first(S) U follow(body) = {**  **while, return, if, break, continue, ID, } }**  **follow(body) = { } }** |
| <array> 🡪 [<E>]<array1>  <array1> 🡪 [<E>]|  | **First(array) = {[}**  **First (array1) = {[, }** | **Follow(array1) = {)}** |
| **EXPRESSION GRAMMAR**: |  |  |
| <OE> 🡪 <AE> <OEDASH> | First (OE) = first(AE) = {self, super, const, ( , ! , ID}  **First (OE) = {self, super, const, ( , ! , ID}** | **Follow (OE) = { ‘,’ , ; , ),}, ] }** |
| <OEDASH> 🡪 || <AE> <OEDASH>  <OEDASH> 🡪 | **First (OEDASH) = {||, }** | Follow(OEDASH) = follow(OE)  **Follow (OEDASH) = {‘,’ , ; , ),}, ] }** |
| <AE> 🡪 <RE> <AEDASH> | First (AE) = first(RE) = {self, super, const, ( , ! , ID}  **First (AE) = {self, super, const, ( , ! , ID}** | **Follow (AE) = {||, ‘,’ , ; , ), },] }** |
| <AEDASH> 🡪 && <RE> <AEDASH>  <AEDASH> 🡪 | **First (AEDASH) = {&&, }** | **Follow (AEDASH) = {||, ‘,’ , ; , ),}, ] }** |
| <RE> 🡪 <E> <REDASH> | First (RE) = first(E) = {self, super, const, ( , ! , ID}  **First (RE) = {self, super, const, ( , ! , ID}** |  |
| <REDASH> 🡪ROP <E> <REDASH>  <REDASH> 🡪 | **First (REDASH) = {ROP, }** | **Follow (REDASH) ={&&, ||, ‘,’ , ; , ), },] }** |
| <E> 🡪 <T> <EDASH> | First (E) = first(T) = {self, super, const, ( , ! , ID}  **First (E) = {self, super, const, ( , ! , ID}** |  |
| <EDASH> 🡪 PM <T> <EDASH>  <EDASH> 🡪 | **First (EDASH) = {PM, }** | **Follow(EDASH) ={ROP, &&, ||, ‘,’ , ; , ),}, ] }** |
| <T> 🡪 <F> <TDASH> | First (T) = first(F) = {self, super, const, ( , ! , ID}  **First (T) =** **{self, super, const, ( , ! , ID}** |  |
| <TDASH> 🡪 MDM <F> <TDASH>  <TDASH> 🡪 | **First (TDASH) = {MDM, }** | **Follow (TDASH) = {PM, ROP, &&, ||, ‘,’ , ; , ),}, ]}** |
| <F> 🡪 <TS> ID <LHP>  <F> 🡪 <const>  <F> 🡪 ( <E>)  <F> 🡪 !F  <F> 🡪 ID<LHP> | First (F) = first(TS) = {self, super}  First (F) = {const}  First (F) = {(}  First (F) = {!}  First (F) = {ID}  **First(F) = {self, super, const, ( , ! , ID}** |  |
| <TS> 🡪 self. | super. | **First(TS) = {self, super}** |  |
| **<LHP> 🡪 <array>** *//array*  **<LHP> 🡪 (<argument>) <LHP3>**  *//func call*  **<LHP>🡪<dictionary>** *//dictionary*  **<LHP> 🡪 <LHP1>**  **<LHP> 🡪 ** | First(LHP) = first(I1) = {[}  First (LHP) = {(}  First (LHP) = { { }  First(LHP) = first(LHP1) = {.}  **First (LHP) = { { , [ , ( , . , }** | **Follow(LHP) = {= , CompAsgnOP, ++, --, MDM, PM, ROP, &&, ||, ‘,’ , ; , ),}, ]}** |
| **<LHP3> 🡪 <LHP1> | **  **<LHP1> 🡪 .ID <LHP>** | **First (LHP3) = {. , }**  **First (LHP1) = {.}** | **Follow(LHP3) = {= , CompAsgnOP, ++, --, MDM, PM, ROP, &&, ||, ‘,’ , ; , ),}, ]}** |
| **FUNCTION DECLARATION**: |  |  |
| <func\_st> 🡪 func ID ( <parameters>) { <MST>} | **First (func\_st) = {func}** |  |
| <parameters> 🡪 DT ID <p0> <p1>  <parameters> 🡪 | **First (parameters) = {DT, }** | **Follow (parameters) = { ) }** |
| <p0> 🡪 = <E><p1>  <p0> 🡪 <array\_dec>  <p0> 🡪  | **First(p0) = {= ,, [ }** | **Follow (p0) = { , , ) }** |
| <p1> 🡪,<parameters>  <p1> 🡪  | **First (p1) = {, , }** | **Follow (p1) = { ) }** |
| <return\_st> 🡪 return <OE>; | **First (return\_st) = {return}** |  |
| <argument> 🡪 <OE><A1> | | First (argument) = first(OE) = {self, super, const, ( , ! , ID}  First (argument) = {}  **First (argument) = {self, super, const, ( , ! , ID, }** | **Follow (argument) = { ) }** |
| <A1> 🡪 , <argument>  <A1> 🡪  | **First (A1) = {, , }** | **Follow (A1) = { ) }** |
| **CLASS DECLARATION**: |  |  |
| <class\_def> 🡪class ID <inherits> {<class\_body>} | **First (class\_def) = {class}** |  |
| <inherits> 🡪 : ID  <inherits>🡪 implements  <inherits> 🡪 | **First (inherits) = {:, implements, }** | **Follow (inherits) = { { }** |
| <class\_body> 🡪 <SST2> <class\_body>| **** | **First(class\_body) = {static, final, abstract, func, DT,  }** | **Follow(class\_body) = { } }** |
| *<SF> 🡪 sdouble|fdouble|******* | *first(SF) = {static, final,  }* |  |
| **<object\_call> 🡪 obj ID (<argument>);** | **First(object\_call) = {obj}** |  |
| <interface> 🡪interface ID {<interface\_body>}  <interface\_body>🡪<SST2> <interface\_body>| **** | **First(interface) = {interface}**  **First(interface\_body) = {static, final, abstract, func, DT,  }** | **Follow(interface\_body) = { } }** |
| **Array 2D, DEC + Initialization**: |  |  |
| <array\_dec> 🡪 [] <dec2> | **First(array\_dec) = {[}** |  |
| <dec2> 🡪 <init1><arr> | []<init2><arr> | **First(dec2)** = first(init1) U {[} => {= , } – {} U {; , ‘,’} => **{; , ‘,’, =, [}** |  |
| <arr> 🡪 ; | , ID <array\_dec> | **First (arr) = { ; , ‘,’}** |  |
| <init1> 🡪 = [<Exp>] | **** | **First(init1) = {= , }** | Follow(init1) = first(arr) = **{; , ‘,’}** |
| <Exp> 🡪 <E><Exp1> |**** | First (Exp) = first(E) U {}= **{self, super, const, ( , ! , ID,  }** | **Follow(Exp) = {]}** |
| <Exp1> 🡪 ,<E><Exp1> | **** | **First (Exp1) = {, }** | **Follow (Exp1) = {]}** |
| <init2> 🡪 = [<init2d>] | **** | **First(init2) = {=, }** | **Follow (init2) = {; , ‘,’}** |
| <init2d> 🡪 [<Exp>]<init2\_dash>| **** | **First(init2d) = {[, }** | **Follow(init2d) = {]}** |
| <init2\_dash> 🡪 ,<init2d> | **** | **First(init2\_dash) = {‘,’ , }** | **Follow(init2\_dash) = {]}** |
| <callinit2d> 🡪 ,<init2d>|  | **First(callinit2d) = {‘,’ ,}** | **Follow(callinit2d) = {]}** |
| <SST> 🡪 **<while\_st> |<return\_st> | <if\_else\_st> | <brk> | <cont> | <create>**  **ID<call>**  **<call> 🡪 <LHP><f3d>**  <create> 🡪 <TS> ID = ID; | First (SST) = {while} U {for} U {return} U {if} U {break} U {continue} U {ID}  **First(SST) = {while, for, return, if, break, continue, ID}**  **First(call) = {[ ,{, ( , ., ++, --, = , CompAsgnOP}** |  |
| <SST2> 🡪 <fdouble><sdouble> <SST3>  <SST3> 🡪<adouble> <func\_st> | <fdec> | **First(SST2) = {static, final, abstract, func, DT}**  **First (SST3) = {abstract, func, DT}** |  |
| <fdec> 🡪 DT ID <declaration>  <declaration> 🡪 <Dec> | <array\_dec>| <dictionary> | **First(fdec) = {DT}**  First (declaration) = first(dec) U first(array\_dec) = {; , ‘,’ , =} U {[}  **First (declaration) = { ; , ‘,’, =,{, [}** |  |
| **<sdouble> 🡪 static | **  **<fdouble> 🡪final |**  **<adouble> 🡪 abstract | ** | First(sdouble) = {static, }  First(fdouble) = {final, }  First (adouble) = {abstract, } | Follow(sdouble)= **{abstract, func, DT}**  Follow(fdouble) = **{static, abstract, func, DT, class}**  Follow(adouble) = **{func}** |
| <body> 🡪 ; | <SST> |<SST2> | <MST> | **First(body) = { ; , while, return, if, break, continue, ID, static, final, abstract, DT, func, }** |  |
| <MST> 🡪 <SST> <MST>|<SST2><MST> |  | **First(MST) = { while, return, if, break, continue, ID, , static, final, abstract, func, DT}** | **Follow(MST) = { } }** |
| **DICTIONARY** |  |  |
| <dictionary> 🡪{ <key\_value\_list> }  <key\_value\_list> 🡪 <key\_value\_pair>  <key\_value\_list>🡪 <key\_value\_pair> , <key\_value\_list>  <key\_value\_pair> 🡪 ID : <keyvalue>  <keyvalue> 🡪 <E> | **First(dictionary)= { { }**  **First(key\_value\_list)=first(key\_value\_pair)={ID}**  **First(keyvalue)= First(E)= {self, super, const, ( , ! , ID}** | **Follow(key\_value\_list)= { } }**  **Follow(key\_value\_pair)= Follow(keyvalue)={ }, , }** |