|  |  |  |  |
| --- | --- | --- | --- |
| <S> --> <final> <struct> { <CB> | First<S> = First<final> = { final, E } U First <Struct> = { class , final } | First<S> = { abstract } = {abstract , final , class } | Follow<S> = { $ } |
| <S> --> abstract <struct> { <ACB> | First<S> = { abstract } |  |  |
|  |  |  |  |
| <S’> --> <S> | First<S’> = first<S> = {abstract , final , class } | First<S’> = { abstract ,final , class, E } | Follow<S’> = follow <d4> U follow <d5> U follow <F> U follow <dd4> U follow <dd5> U follow <CB> U follow <ACB> U follow <ACBB> U follow <FACB> = { $ ,} } |
| <S’> --> E | First<S’> = first<S> = { E } |  |  |
| <CB> --> } <S’> | First<CB> = { } } | First<CB> = { public, private, final, static, void, ID, DT , } } | Follow<CB> = follow<S> = { $ } |
| <CB> --> <CBB> | First<CB> = First<CBB> = { public, private, final, static, void, ID, DT } |  |  |
| <CBB> --> public <d1> | First<CBB> = { public } | First<CBB> = { public, private, final, static, void, ID, DT } | Follow<CBB> = follow<CB> U follow<CB’> = { $ ,} } |
| <CBB> --> private <d6> | First<CBB> = { private } |  |  |
| <CBB> --> final<d3> | First<CBB> = { final } |  |  |
| <CBB> --> static<F> | First<CBB> = { static } |  |  |
| <CBB> --> <F> | First<CBB> = { void, ID, DT } |  |  |
| <d1> --> final <d3> | First<d1> = { final } | First<d1> = { final, static, void, ID, DT } | Follow<d1> = follow<CBB> = { $ ,} } |
| <d1> --> <d2> | First<d1> = { static } |  |  |
| <d1> --> <F> | First<d1> = first<F> = { void, ID, DT } |  |  |
| <d2> --> static <d4> | First<d2> = { static } |  |  |
| <d4> --> void<d5> | First<d4> = { void } | First<d4> = { void, ID, DT } | Follow<d4> = { $ ,} } |
| <d4> --> DT<f2><CB> | First<d4> = { DT } |  |  |
| <d4> --> ID<f1><CB> | First<d4> = { ID } |  |  |
| <d5> --> <fn\_d> <CB> | First<d5> = { func } | First<d5> = { func, main } | Follow<d5> = { $ ,} } |
| <d5> --> <Main><d’> | First<d5> = { main } |  |  |
| <CB’> --> <CBB> | First<CB’> = first<CBB> = { public, private, final, static, void, ID, DT } | First<CBB> = { public, private, final, static, void, ID, DT , } } | Follow<CB’> = } , abstract ,final , class , $ |
| <CB’> --> } | First<CB’> = { } } |  |  |
|  |  |  |  |
| <d6> --> final<d3> | First<d6> = { final } | First<d6> = { final, static, void, ID, DT } | Follow<d6> = follow<CBB> = { $ ,} } |
| <d6> --> <d3> | First<d6> = First<d3> = { static, void, ID, DT } |  |  |
| <d3> --> static <F> | First<d3> = { static } | First<d3> = { static, void, ID, DT } | Follow<d3> = follow<d6> = { $ ,} } |
| <d3> --> <F> | First<d3> = first<F> = { void, ID, DT } |  |  |
| <F> --> void <fn\_d> <CB> | First<F> = { void } | First<F> = { void, ID, DT } | Follow<F> = follow <d3> U follow <d1> U follow <CBB> = { $ ,} } |
| <F> -- ID <f1> <CB> | First<F> = { ID } |  |  |
| <F> --> DT <f2> <CB> | First<F> = { DT } |  |  |
|  |  |  |  |
| <f1> - -> ID <dec> | First<f1>={ ID } | First<f1>={ ID , ( , [ , func } |  |
| <f1> - -> <b> <fn\_dec> | First<f1>={ [ , func } |  |  |
| <f1> - -> <constructor> | First<f1>={ ( } |  |  |
| <f2>- -> ID <f2> /<b> <fn\_dec> | First<f2>={ ID , [ , func } |  |  |
| <d’> -> <ddd’> / } <def> | First<ddd’> = public, private, final, static, void, ID, DT , } |  | Follow<d’>= Follow<d5> = { $ ,} } |
| <ddd’> --> public <d6’> | First<ddd’> = { public } | First<dd d’> = { public, private, final, static, void, ID, DT } | Follow<ddd’> = }  $,abstract ,final ,class, |
| <ddd’> --> private <d6’> | First< ddd’> = { private } |  |  |
| <ddd’> --> final<d3’> | First< ddd’> = { final } |  |  |
| <ddd’> --> static<F’> | First< ddd’> = { static } |  |  |
| <ddd’> --> <F’> | First< ddd’> = First<F’> = { void, ID, DT } |  |  |
|  |  |  |  |
| <d6’> --> final<d3’> | First<d6’> = { final } | First<d6’> = { final, static, void, ID, DT } | follow<d6’> = } |
| <d6’> --> <d3’> | First<d6’> = First<d3’> = { static, void, ID, DT } |  |  |
| <d3’> --> static <F’> | First<d3’> = { static } | First<d3’> = { static, void, ID, DT } | Follow<d3’> = follow <d’> U follow<d6’> = } |
| <d3’> --> <F’> | First<d3’> = First<F’> = { void, ID, DT } |  |  |
| <F’> --> void <fn\_d> <d’> | First<F’> = { void } | First<F’> = { void, ID, DT } | Follow<F’> = follow <d3’> U follow <d’> = } |
| <F’> --> ID <f1> <d’> | First<F’> = { ID } |  |  |
| <F’> --> DT <f2> <d’> | First<F’> = { DT } |  |  |
|  |  |  |  |
| <ACB> --> } <S’> | First<ACB> = { } } | First<ACB> = { public, private, final, static, void, ID, DT , } , abstract } | Follow<ACB> = follow<S> = { $ } |
| <ACB> --> <ACBB> | First<ACB> = First<ACBB> = { public, private, final, static, void, ID, DT, abstract } |  |  |
| <ACBB> --> public <dd1> | FirstA<ACBB> = { public } | First<ACBB> = { public, private, final, static, void, ID, DT , abstract} | Follow<ACBB> = follow<ACB> U follow<ACB’> = { $ ,} } |
| <ACBB> --> private <dd6> | FirstA<ACBB> = { private } |  |  |
| <ACBB> --> final<dd3> | First<ACBB> = { final } |  |  |
| <ACBB> --> static <FACB> | First<ACBB> = { static } |  |  |
| <ACBB> --> <FACB> | First<ACBB> = First<FACB> = { void, ID, DT } |  |  |
| <ACBB> --><abs\_func> <ACB> | First<ACBB> = { abstract } |  |  |
| <dd1> --> final <dd3> | First<dd1> = { final } | First<dd1> = { final, static, void, ID, DT } | Follow<dd1> = follow<ACBB> = { $ ,} } |
| <dd1> --> <dd2> | First<dd1> = { static } |  |  |
| <dd1> --> <FACB> | First<dd1> = first<F> = { void, ID, DT } |  |  |
| <dd2> --> static <dd4> | First<dd2> = { static } |  |  |
| <dd4> --> void<dd5> | First<dd4> = { void } | First<dd4> = { void, ID, DT } | Follow<dd4> = { $ ,} } |
| <dd4> --> DT <f2> <ACB> | First<dd4> = { DT } |  |  |
| <dd4> --> ID <f1> <ACB> | First<dd4> = { ID } |  |  |
| <dd5> --> <fn\_d> <ACB> | First<dd5> = { func } | First<dd5> = { func, main } | Follow<dd5> = { $ ,} } |
| <dd5> --> <Main> <dd’> | First<d5> = { main } |  |  |
| <ACB’> --> <ACBB> | First<ACB’> = first<ACBB> = { public, private, final, static, void, ID, DT , abstract} | First<ACB’> = { public, private, final, static, void, ID, DT , } , abstract } | Follow<ACB’> = } , $ , abstract , final , class |
| <ACB’> --> } | First<ACB’> = { } } |  |
|  |  |  |  |
| <dd6> --> final<dd3> | First<dd6> = { final } | First<dd6> = { final, static, void, ID, DT } | Follow<dd6> = follow<ACBB> = { $ ,} } |
| <dd6> --> <dd3> | First<dd6> = First<dd3> = { static, void, ID, DT } |  |  |
| <dd3> --> static <FACB> | First<dd3> = { static } | First<dd3> = { static, void, ID, DT } | Follow<dd3> = follow<dd6> U follow<ACBB> = { $ ,} } |
| <dd3> --> <FACB> | First<dd3> = first <FACB> = { void, ID, DT } |  |  |
| <FACB> --> void <fn\_d> <ACB> | First<FACB> = { void } | First<FACB> = { void, ID, DT } | Follow<FACB> = follow <dd3> U follow <dd1> U follow <ACBB> = { $ ,} } |
| <FACB> -- ID <f1> <ACB> | First<FACB> = { ID } |  |  |
| <FACB> --> DT <f2> <ACB> | First<FACB> = { DT } |  |  |
|  |  |  |  |
| <dd’> -> <dddd’> / }<def> | First< dddd’> = public, private, final, static, void, ID, DT , abstract ,} |  |  |
| <dddd’> --> public <dd6’> | First<dddd’> = { public } | First< dddd’> = { public, private, final, static, void, ID, DT , abstract } | Follow<dddd’> = }  final , abstract , class ,$ |
| <dddd’> --> private <dd6’> | First< dddd’> = { private } |  |  |
| <dddd’> --> final<dd3’> | First< dddd’> = { final } |  |  |
| <dddd’> --> static <FACB’> | First< dddd’> = { static } |  |  |
| <dddd’> --> <FACB’> | First< dddd’> = First<F’> = { void, ID, DT } |  |  |
| <dddd’> --> <abs\_func> <dd’> | First< dddd’> = { abstract } |  |  |
|  |  |  |  |
| <dd6’> --> final<dd3’> | First<dd6’> = { final } | First<dd6’> = { final, static, void, ID, DT } |  |
| <dd6’> --> <dd3’> | First<dd6’> = First<dd3’> = { static, void, ID, DT } |  |  |
| <dd3’> --> static <FACB’> | First<dd3’> = { static } | First<dd3’> = { static, void, ID, DT } | Follow<dd3’> = follow <dd’> = } |
| <dd3’> --> <FACB’> | First<dd3’> = First<FACB’> = { void, ID, DT } |  |  |
| <FACB’> --> void <fn\_d> <dd’> | First< FACB’> = { void } | First< FACB’> = { void, ID, DT } | Follow<FACB’> = follow <dd3’> U follow <dd’> = } |
| <FACB’> --> ID <f1> <dd’> | First< FACB’> = { ID } |  |  |
| <FACB’> --> DT <f2> <dd’> | First< FACB’> = { DT } |  |  |
|  |  |  |  |
| <Main> --> main () { <MST> } | First< Main > = { main } |  |  |
|  |  |  |  |
| <def> --> <class\_def> <def> | First< def >= First< class\_def> = { final , abstract , class } | First< def >= First< class\_def> = { final , abstract , class , E } | Follow<def> = Follow<FACB’> U Follow<dd’> U Follow<F’> = follow<d5> = follow<dd5> = $ , } |
| <def> --> E | First< def > = { E } |  |  |
| <class\_def> --> <final> <Struct> { <d’> | First< class\_def > = { final ,E } U First <Struct> = { final , class } | First< class\_def > = { final , abstract , class } |  |
| <class\_def> --> abstract <Struct> { <dd’> | First< class\_def > = { abstract } |  |  |
|  |  |  |  |
| <final> --> final | First< final > = { final } | First< final > = { final , E } | Follow< final > = First <Struct> = { class } |
| <final> --> E | First< final > = { E } |  |  |
| <Struct>--> class ID <inherit> | First< Struct > = { class } |  |  |
| <inherit> --> inherit ID <lst> / E | First < inherit > = { inherit } |  | Follow<inherit> = { |
| <lst> --> , ID <lst> / E | First < lst > = { , , E} |  | Follow< lst > = { |
|  |  |  |  |
|  |  |  |  |
| <fn\_dec> --> func ID (<ar\_dec>) {<MST>} | First< fn\_dec > = { func } |  |  |
| <b> --> [ ] <b1> | First<b> = { [ } | First<b> = { [ ,E} | Follow<b> = First < fn\_dec > = { func } |
| <b> --> E | First<b> = { E } |  |  |
| <b1> --> [ ] <b2> | First<b1> = { [ } | First<b1> = { [ ,E} | Follow<b1> = Follow < b > ={ func} |
| <b1> --> E | First<b1> = { E } |  |  |
| <b2> --> [ ] | First<b2> = { [ } | First<b2> = { [ ,E} | Follow<b2> = Follow <b1> = { func } |
| <b2> --> E | First<b2> = { E } |  |  |
|  |  |  |  |
|  |  |  |  |
| < ar-dec > --> DT ID <b><decc> | first <ar\_dec> = {DT} | first <ar\_dec> = {DT, ID, E} | Follow<arr\_dec> = ) |
| < ar-dec > --> ID ID <b><decc> | first <ar\_dec> = {ID} |  |  |
| < ar-dec > --> E | first <ar\_dec> = {E} |  |  |
| <decc> --> , <q> | first <decc> = { , } | first <decc> = { , , E } | Follow <decc> = Follow <arr\_dec> = ) |
| <decc> --> E | first <decc> = { E } |  |  |
| <q> --> DT ID <bb><decc> | first <q> = { DT } | first <q> = { DT , ID } | Follow <q> = follow <decc> = ) |
| <q> --> ID ID <bb><decc> | first <q> = { ID } |  |  |
|  |  |  |  |
|  |  |  |  |
| <bb> --> [ ] <bb1> | First<bb> = { [ } | First<bb> = { [ ,E} | Follow<bb> = First < decc > = { , , ) } |
| <bb> --> E | First<bb> = { E } |  |  |
| <bb1> --> [ ] <bb2> | First<bb1> = { [ } | First<bb1> = { [ ,E} | Follow<bb1> = Follow < bb >= { , , ) } |
| <bb1> --> E | First<bb1> = { E } |  |  |
| <bb2> --> [ ] | First<bb2> = { [ } | First<bb2> = { [ ,E} | Follow<bb2> = Follow <bb1> = { , , ) } |
| <bb2> --> E | First<bb2> = { E } |  |  |
|  |  |  |  |
|  |  |  |  |
| <abs-func> --> abstract <A> <Ret-type> func ID (<ar-dec>) ; | First< abs\_func > = { abstract } |  |  |
| <A> --> public / E | First< A > = { public ,E} |  | Follow<A> = First < Ret\_type > = { void , DT , ID } |
| <Ret\_type> --> void / DT <b> / ID <b> | First< Ret\_type > = { void , DT , ID } |  |  |
|  |  |  |  |
| <As\_fn> --> (<exp\_d>) <h> | First< As\_fn > = { ( } | First<As\_fn> =  { ( , [ , . ,= } | Follow <As\_fn> = { ; } |
| <As\_fn> --> <ass> | First< As\_fn > = { = } |  |  |
| <As\_fn> --> [<exp>] <more> <t> | First< As\_fn > = { [ } |  |  |
| <As\_fn> --> <new> | First< As\_fn > = { . } |  |  |
|  |  |  |  |
| <t>-> -> <new> | First<t> = { . } | First<t> = {= , .} | Follow <t> = { ; } |
| <t>-> -> <assign> | First<t> = { = } |  |  |
|  |  |  |  |
|  |  |  |  |
| <more> --> [<exp>] <more2> | First< more > = { [ } | First< more > = { [ ,E } | Follow<more> = first <t> = { = , . } |
| <more> --> E | First< more > = { E } |  |  |
| <more2>- ->[<exp>]/E |  | First< more2 > = { [ ,E } | Follow<more2> = follow<more> = { = , .} |

|  |  |  |  |
| --- | --- | --- | --- |
| <h> --> <new> | First< h > = { . } | First< h > = { . , E } | Follow<h> =  Follow <As\_fn> = { ; } |
| <h> --> E | First< h > = { E } |  |  |
| <new> --> . ID <As\_fn> | First< new > = { . } | First< new > = { . } | ; , = |
| <assign> --> = <exp> | First< assign > = { = } |  | Follow(as\_fn> = { ; } U follow <t,as\_fn,Rhp> =  ; |
| <RHP> -->  (<exp\_d>) <h’> /  <RHP’> /  <assign> / [<exp>] <more> <h’> | First<RHP> = { ( , = , . , [ } |  | Follow<RHP> U follow <o> = MDM, PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <h’> - -> <RHP’>/E | First<h’> = { . , E} |  | Follow<h’>=follow<RHP>=  MDM, PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <RHP’> - -> .ID<RR> | First<RHP’> = { . } |  |  |
| <RR> - -> <RHP>/E | First<RR>=  First<RHP> = { ( , = , . , [ } |  | Follow<RR> = follow <RHP’> = follow <RHP> =  MDM, PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <exp> --> <AE><exp’> | First< exp > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } |  | follow (assign)=  ; , = , ) , ] , , |
| <exp’> --> E / LOL <AE> <exp’> | First< exp’ > = { E , LOL } |  | Follow<exp’> = follow<exp> =  ; , = , ) , ] , , |
| <AE> --> <RE> <AE’> | First< AE > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } |  | Follow<AE> = first<exp’> =  { E , LOL} U follow<exp’> =  ; , = , ) , ] , , |
| <AE’> --> E / LOH <RE> <AE’> | First< AE’ > = {E  , LOH } |  | Follow<AE’> = follow<AE> = LOL , ; , = , ) , ] , , |
| <RE> --> <E> <RE’> | First< RE > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } |  | Follow<RE>= first<AE’>= { E , LOH} U follow<AE>=  LOH , LOL , ; , = , ) , ] , , |
| <RE’> --> ROP <E> <RE’> / E | First< RE’ > = { E,ROP } |  | Follow<RE’> = follow<RE> = { LOH , LOL , ; , = , ) , ] , , |
| <E> --> <T> <E’> | First< E > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } |  | Follow<E> = first <RE’> = ROP , LOH , LOL , ; , = , ) , ] , , |
| <E’> --> PM <T> <E’> / E | First< E’ > = { E,PM } |  | Follow <E’> = follow <E> = ROP , LOH , LOL , ; , = , ) , ] , , |
| <T> --> <F> <T’> | First< T > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } |  | Follow<T> = first <E’> = {PM , E} U follow<E> =  PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <T’> --> MDM <F> <T’> / E | First< T’ > = { E,MDM } |  | Follow<T’> = follow<T> U follow<T’> = PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <F> --> <TS> ID <o> | First< TS > = { this , super , ID } | First< F > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } | Follow<f>=first<t’> = {E,MDM} U {  Follow <T> U follow <T’> = MDM, PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <F> --> <const> | int\_const, flt\_const,  str\_const,char  Const,bool\_const |  |
| <F> --> (<exp>) | ( |  |
|  |  |  |
| <TS> --> this. | This. | First< TS > = { this , super , E } | Follow <TS> = ID |
| <TS> --> super. | Super. |  |
| <TS> --> E | E |  |
| <o>--><RHP>/E | First<o>={ ( , = , . , [ , E} |  | Follow<o>=follow<f > = MDM, PM, ROP , LOH , LOL , ; , = , ) , ] , , |
| <Dec> --> [] <D> <list> | First<dec> = { [ } | First<dec> = { [ , = , ; , , } |  |
| <Dec> --> <init\_d> <list> | First<dec> = { E , = } U { ; , ,} |  |  |
|  |  |  |  |
| <init\_d> --> = <init> | First<init\_d> = { = } | First<init\_d> = { E , = } | Follow<init\_d> = first<list> = { ; , ,} |
| <init\_d> --> E | First<init\_d> = { E } |  |
| <init> --> new ID(<arr\_dec>) | First< init > = { new } | First< init > = { new , this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } | Follow<init> = Follow<init\_d> = { ; , ,} |
| <init> --> <exp> | First< init > = First < exp > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } |  |
| <list> --> ; | First< list > = { ; } | First< list > = { ; , , } |  |
| <list> --> , ID <dec> | First< list > = { , } |  |  |
|  |  |  |  |
| <D> --> <init1> | First< D > = First< init1 > = { = , E } | First< D > = { [ , = , E } | Follow <D> = first <list>  = { ; , , } |
| <D> --> [ ] <2D> | First< D > = { [ } |  |
| <2D> --> <init2> | First< 2D > = First< init2 > = { = , E } | First< 2D > = { = , [ , E } | Follow<2D> = follow<D>  = { ; , , } |
| <2D> --> [ ] <init3> | First< 2D > = { [ } |  |  |
| <init1> --> = <init1’> | First< init1 > = { = } | First< init1 > = { = ,E } | Follow<init1> = follow<D>  = { ; , , } |
| <init1> --> E | First< init1 > = { E } |  |
| <init1’> --> new DT <br-size> | First< init1‘ > = { new } | First< init1‘ > = { new , { } |  |
| <init1’> --> {<exp\_d>} | First< init1‘ > = { { } |  |
|  |  |  |  |
| <br-size> --> [<exp>] | First< br-size > = { [ } |  |  |
| <exp-d> --> <exp> <cont> | First< exp-d > = First< exp > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } | First< exp-d > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const , E } | Follow<exp-d> = { ) ,}} |
| <exp-d> --> E | First< exp-d > ={E} |  |
| <cont> --> , <exp> <cont> | First< cont > = { , } | First< cont > = { , , E } | Follow<cont> = follow<exp-d> = { ) , } } |
| <cont> --> E | First< cont > = { E } |  |
|  |  |  |  |
| <init2> --> = <init2’> | First< init2 > = { = } | First< init2 > = { = , E } | Follow <init2> = Follow<2D> = follow <D> = { ; , , } |
| <init2> --> E | First< init2 > = { E } |  |  |
| <init2’> --> new DT <br-size> <br-size> | First< init2‘ > = { new } | First< init2‘ > = { new , { } |  |
| <init2’> --> {<data2>} | First< init2‘ > = { { } |  |  |
| <data2> --> {<exp\_d>} <cont2> | First< data2 > = { { } | First< data2 > = { { ,E } | Follow<data2> = } |
| <data2> --> E | First< data2 > = { E } |  |  |
| <cont2> --> , {<exp\_d>} <cont2> | First< cont2 > = { , } | First< cont2 > = { , , E } | Follow<cont2> = follow<data2> = } |
| <cont2> --> E | First< cont2 > = { E } |  |  |
|  |  |  |  |
| <init3> --> = <init3’> | First< init3 > = { = } | First< init3> = { E , = } | Follow<init3> = follow<2D> = { ; , ,} |
| <init3> --> E | First< init3 > = { E } |  |  |
| <init3’> --> new DT <br-size><br-size><br-size> | First< init3‘ > = { new } | First< init3‘ > = { new , { } |  |
| <init3’> --> {<data3>} | First< init3‘ > = { { } |  |  |
| <data3> --> {<data2>} <cont3> | First< data3 > = { { } | First< data3 > = { { ,E } | Follow<data3> = } |
| <data3> --> E | First< cont3 > = { E } |  |  |
| <cont3> --> ,{<data2>} <cont3> | First< cont3 > = { , } | First< cont 3 > = { , , E } | Follow<cont3> = follow<data3> = } |
| <cont3> --> E | First< cont3 > = { E } |  |  |
| <break> --> break ; | First< break > = { break } |  | Follow<break> = follow<SST> = first <MST> = { DT, ID , this , super , break, continue, Return , try , loop , if , switch} |
| <continue> --> continue ; | First< continue > = { continue } |  | Follow <continue> = follow<SST> = first<MST> = { DT, ID , this , super , break, continue, Return , try , loop , if , switch} |
|  |  |  |  |
| <return> --> return <oo> | First< return > = { return } |  |  |
| <oo> --> <exp> ; | First< oo > = First< exp > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const } | First<o o > = { this , super , ID , ( ,  int\_const,  flt\_const,  str\_const, charConst,  bool\_const, ; } |  |
| <oo> --> ; | First< oo > = { ; } |  |  |
| <try-catch> --> try {<MST>} <catch> <finally> | First<try\_catch> = {try} |  |  |
| <catch> --> catch (ID ID) {<MST>} <catch> |  | First<catch> = {catch , E} | Follow<catch> = first<finally> U follow<sst> = finally, DT , ID , super , throw , this, break, continue, return , try , loop , if , switch, } , case , default , : , else |
| <catch> --> E |  |  |  |
| <finally> --> finally {<MST>} |  | First<finally> = {finally, E} | Follow <finally> = DT , ID , super , throw , this, break, continue, return , try , loop , if , switch, } , case , default , : , else |
| <finally> --> E |  |  |  |
|  |  |  |  |
| <loop> --> loop (<exp>) <body> | First <loop> = {loop} |  | Follow<loop> = DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch, } , case , default , : , else |
| <body> --> ; |  | First <body> = ; , { , DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch, } , case , default , : , else | Follow <body> = follow<loop> U follow<if\_else> U first<else> = { else , DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch, } , case , default , : , else } |
| <body> --> <SST> |  |  |  |
| <body> --> {<MST>} |  |  |  |
|  |  |  |  |
| <if-else> --> if (<exp>) <body><else> | First<if\_else> = { if } |  | DT , ID , super ,throw , this, break, continue, return , try , loop , if , switch, } , case , default , : , else |
| <else> --> E / else <body> |  | First <else> = {E , else} | Follow <else> = first <body> = ; , { , DT , ID , super , throw, this, break, continue, return , try , loop , if , switch, } , case , default , : , else |
|  |  |  |  |
| <switch\_st> --> switch (<exp>) {<p1>} | First <switch\_st> = {switch} |  |  |
| <p1> --> <case> / <default> / E |  | First <p1> = {E , case, default} | Follow<p1> = Follow<case> U } = } |
|  |  |  |  |
| <case> --> case <const> : <LL> <p1> | First <case> = {case} |  | Follow<case> = } |
| <default> --> default : <LL> <cse> | First <default> = {default} |  |  |
| <LL> --> <MST> / {<MST>} / E |  | First <LL> = { , E , DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch | Follow<LL> = First<p1> U first <cse> = { E , case , default } U follow <case> U {case, E} U follow <default> = { case, default, : } |
| <cse> -> case <const> : <LL> <cse>/E |  | First <cse> = {case , E} | Follow<cse> = Follow <default> = } |
|  |  |  |  |
| <MST> --> E / <SST> <MST> |  | First<MST = {E , DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch } | Follow<MST> = } U follow <LL> = { } , case , default , : } |
|  |  |  |  |
| <SST> --> DT ID <Dec>/  ID<for\_fn>/<throw>/this<new>;/  super<new>;/  <break>/<continue>/<return>/  <try\_catch>/<loop>/<if\_else>/  <switch\_st> |  | First<SST> = {DT, ID , super , throw , this , break , continue , return , try , loop , if , switch} | DT , ID , super , throw , this, break, continue, return , try , loop , if , switch, } , case , default , : , else |
| <for\_fn> --> ID <Dec> | First <for\_fn> = { ID } | First <for\_fn> = { ID , instanceof , ( , [ , . ,= } |  |
| <for\_fn> --> <As\_fn> ; | First <for\_fn> First<As\_fn> = { ( , [ , . , = } |  |  |
| <for\_fn> --> <instance\_of> | First <for\_fn> = { instanceof } |  |  |
| <throw>- ->throw<k> | First<throw>={throw} |  |  |
| <k>- ->ID;/new ID (string\_const); | First<k>={ ID , new } |  |  |
| <instance\_of> --> instanceof ID; | First<instance\_of> = { instanceof } |  |  |
| <constructor> --> (<ar-dec>) {<const\_MST>} | First<constructor>={ ( } |  |  |
| <super>- ->super(<arr\_dec>); | First<super>={ super } |  |  |
| <this>- ->this(<arr\_decl>); | First<this>={ this } |  |  |
| <const\_MST> -> <super> <MST> | First< const\_MST > = { super } | First< const\_MST > = { E , DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch } | Follow< const\_MST > = } |
| <const\_MST> -> <this> <MST> | First< const\_MST > = { this } |  |  |
| <const\_MST> -> <MST> | First< const\_MST > = { E , DT , ID , super ,throw,this, break, continue, return , try , loop , if , switch } |  |  |
|  |  |  |  |
|  |  |  |  |
| <const> - -> int\_const / flt\_const / str\_const / char\_const / bool\_const | First<const> = { int\_const  flt\_const,str\_const,  char\_const,  bool\_const } |  |  |
|  |  |  |  |