**CFGS for syntax**

**<S>** <defs> public class Main<INH>{ <CLASS\_BODY> public static fun main():void {<MST>} <CLASS\_BODY> } <defs>

**<INH**> extends ID| Eps

**<defs>** <CLASS\_DEF><defs>|Eps

**<CLASS\_BODY>** <o\_static> <AM> <DT> <CLASS\_BODY’> |<AM> <CLASS\_BODY’’> |<OBJ><CLASS\_BODY>|<CLASS\_BODY2>|Eps

**<CLASS\_BODY’>** <M>;<CLASS\_BODY>|(<ARGS>){<MST>}<CLASS\_BODY>

**<CLASS\_BODY’’**> DT ID<CLASS\_BODY’> | ID <CLASS\_BODY’’’>

**<CLASS\_BODY’’’>** ID<CLASS\_BODY’>|(<ARGS>){<CONSTRUCTOR\_CALLING><MST>}

**<CLASS\_DEF>** <AM> class ID<INH>{<CLASS\_BODY>}

**<MST>** <SST><MST>|Eps

<**SST>** ID <SST’>| DT<SST”>| <FOR\_LOOP>|<Until-st> |<IF\_ST>| <INC\_DEC>|this.<this1> | super.<super1>

**<OBJ’>** ID = new <DT> (<ARGS2>)

**<ARRAY\_DEC’>** [<E>]<opt> ID<opt2>

**<SST’>** (<ARGS2>)<opt3>;| ID<SST7> ;|[<SST9>; |<P””>=<Q>;|=<SST12>;|<INC\_DEC>;| this.<this’>|super.<super’>

**<SST”>** [<SST11>;| ID <SST10>;| this.<this’>|super.<super’>

**x<SST3>** = new <DT>| <P””>=<Q>

**X<SST4>** <OBJ’>| []<OBJ’>

**X<SST5**> ]<SST4>|<E>]<opt> ID <opt2>

**X<SST6>** ID<DT’> ID<M>| DT<DT’> ID<M>

**<SST7>** = <SST12>|<M’>

**<SST8>** [] ID <M>|ID<M>

**<SST9>** <E>] <opt> <opt3>|]<SST8>

**<SST10>** =<SST12>|<M’>| <P””>=<Q> (<ARGS2>)

**<SST11>** ]<SST13>| <E>]<opt> <opt3>

**<SST12>** new <DT> (<ARGS2>)|<OE><M’>

**<SST13>** [ ] ID <SST12>|ID<SST7>

**x<SST14>** =<SST12>|<M’>

**<opt3>** ID<opt2>|=<Q>

**<SST>** <DEC>|<FUNC\_CALL>|<OBJ>>|<ARRAY\_DEC>|<FOR\_LOOP> |<until\_st>|<IF\_ST>|<ASSIGN\_ST>|<INC>|<DECR>

**<OBJ>** <DT> ID = new <DT> (<ARGS2>)

<**CONSTRUCTOR>** <AM> ID(<ARGS>){<CONSTRUCTOR\_CALLING><MST>}

<**CONSTRUCTOR\_CALLING>** super(<ARGS2>);|this(<ARGS2>);|Eps

**<ASSIGN\_ST>** <P>=<Q>

**<P>** ID<P””>

**<P””>** <P’>| (<ARGS2>);<P’> |[<E>]<opt><P’>|,<Q>| (<ARGS2>)=<P’>| (<ARGS2>),<Q> | (<ARGS2>).<P>

**<Q>** <P>|<OE>

**<P’>** .<P>|Eps

**<DEC>** <O\_Static><AM> <DT> ID<M>;

**<M>** =<OE><M’>|,ID<M>

**<M’>** ,ID<M>|Eps

**<O\_Static>** static|Eps

**<IF\_ST>** if(<Cond>){<MST>} <O\_ELIF><O\_ELSE>

**<O\_ELIF>** elif<IF\_ST>|Eps

**<O\_ELSE>** else{<MST>}

**<switch\_st**> when(<E>) {<MCS>}

**<MCS>** {<switch\_label>} {<SST>}

**<switch\_label>** int\_const -> | else->

**<struct**> struct ID {<DEC>}

**<do\_until>** do{ <MST> } until (<COND>) ;

**<until>** until(<COND>){<MST>}

**<COND**> <OE>|true|false

**<List\_1>** <DT> ID <List\_2> | Eps

**<List\_2>** , <DT> ID <List\_2>

**<FUNC\_CALL**> ID (<ARGS2>);

**<ARGS2>** Eps|<Q><NEXT2>

**<NEXT2>** ,<OE><NEXT2>|Eps|,<OE>;|<MST>

**<FUNC\_DEF>** fun ID (<Argument>):<DT>{<MST>}

**<ARGS>** Eps|<DT><X1><NEXT>

**<NEXT>** ,<DT><X1><NEXT>|Eps

<**X1>** ID<Y1>

**<Y1>** Eps|[<OE>]<opt>

**<ARRAY\_DEC>** <DT2>[<E>]<opt> ID<opt2>;

**<opt>** [<OE>]|Eps

**<opt2>** ,ID<opt2>|Eps|=new <DT2>(<ARGS2>)

**<For\_st**> for ( ID in int\_const ..int\_const )<body>

**<OE>** <AE><OE’>

**<OE’>** or <AE><OE’>|Eps

**<AE>** <RE><AE’>

**<AE’>** and <RE><AE’>|Eps

**<RE>** <E><RE’>

**<RE’>** ROP <E> <RE’>|Eps

**<E>** <T><X>

**<X>** PM <T> <X>|Eps

**<T>** <F> <T’>

**<T’>** MDM <F> <T’>|Eps

**<F>** ID<F’>|<CONST><X>|<INC\_DEC> ID <X> |(<OE>)|

not <F>| this.<this’>| super.<super’>

**<this’>** ID<this\_super’’>

**<this\_super’’**> <X>|(<ARGS2>);<X>

**<super’>** ID <this\_super’’>

**<F’>** <X><F’’>|(<ARGS2>)<X>|[<OE>]<F’’’>|.<P””>=<Q>;

**<F’’>** <INC\_DEC>|eps

**<F’’’>** <x>|[<OE>]<X>

**<DT>** DT<DT’>|ID<DT’>

**<DT’**> [ ]|[ ][ ]|eps

**<DT2>** DT|ID

**<AM>** static|public|private