CFGS		
<structure> -> <defs>		
<defs> -> <class_def> <defs> <defs> -> <interface_def> <defs> <defs> -> <abs_class_def> <defs> <defs> -> <enum_def> <defs> <defs> -> E		
<class_def> -> <NAM> class ID <INH> { <CB> }		
<interface_def> -> interface ID <int_INH>{ <int_B> }		
<abs_class_def> -> abstract class ID <INH> { <abs_B> }		
<enum_def> -> enum ID { <enum_body> }		
<NAM> -> final <NAM> -> E		
<INH> -> extends ID <imp> <INH> -> E <INH> -> <imp>		
<CB> -> <func_def> <CB> <CB> -> <construct_def> <CB> <CB> -> <decl> <CB> <CB> -> <assign_st> <CB> <CB> -> E		
<int_INH> -> extends ID <id Dash>		
<int_B> -> <int_var> <int_B> <int_B> -> <abs_func_def> <int_B>		
<abs_B> -> <func_def> <abs_B> <abs_B> -> <decl> <abs_B> <abs_B> -> <abs_func_def> <abs_B> <abs_B> -> <assign_def> <abs_B> <abs_B> -> <construct_def> <abs_B> <abs_B> -> E		
<enum_body> -> stringConst <m stringConst> <enum_body> -> E		
<imp> -> implements ID <id Dash> <imp> -> E		
<func_def> -> <FNAM> <RAD> def ID (<Param>) { <MST> }		
<construct_def> -> def ID (<Param>) { <MST> }		
<Decl> -> DT <RAS> <AR> < int1> <list1>		
<assign_st> -> ID <LAS>		
<B> -> (<args>) <FNA> <B> -> [<exp>] <D2DA> <ref> <EEXP> <B> -> . ID <B> <B> -> = <exp>;		

<FNA> -> ; <FNA> -> . ID <B>		
<D2DA> -> [<exp>] <D2DA> -> E		
<Ref> -> . ID <B> <Ref> -> E		
<EEXP> -> = <exp> ;		
<LAS> -> = <exp> ; <LAS> -> . ID <B> <LAS> -> ID <obj R>		
<obj R> -> = new ID (<args>) ; <obj R> -> ;		
<id Dash> -> , ID <id Dash> <id Dash> -> E		
<int_var> -> DT ID = <exp> ;		
<abs_func_def> -> abstract <RTD> def ID (<Param>) ;		
<m StringConst> -> , StringConst <m StringConst> <m StringConst> -> E		
<FNAM> -> final <FNAM> -> static <FNAM> -> basic		
<RTD> -> void <RTD> -> DT <AN> <RTD> -> ID		
<AN> -> [ ] <AN> -> E		
<Param> -> <dec> <Param1> <Param> -> <obj> <Param1> <Param> -> E		
<Param1> -> , <dec_obj> <Param1> <Param1> -> E		
<dec> -> DT ID <AN>		

<obj> -> ID ID		
<dec_obj> -> <dec> <dec_obj> -> <obj>		
<MST> -> <SST> <MST> <MST> -> E		
<RAS> -> ID <MAS>		
<MAS> -> (<arg>) <Ref1> <MAS> -> [<exp>] <D2DA> <Ref1> <MAS> -> . ID <MAS> <MAS> -> E		
<Ref1> -> . ID <MAS> <Ref1> -> E		
<AR> -> [ ] <D2DB> <AR> -> E		
<init1> -> = <init1> <init1> -> E		
<list1> -> ; <list1> -> , <RAS> <AR> <init1> <list1>		
<init D> -> <RAS> <init1> <init D> -> <const>		
<args> -> <exp> <argsD> <args> -> E		
<argsD> -> , <exp> <argsD> <argsD> -> E		
<exp> -> <AND OP> <expD>		
<expD> -> ll <AND OP> <expD> -> E		
<AND OP> -> <ROPOP> <ANDOPD>		
<ANDOPD> -> && <ROPOP> <ANDOPD> -> E		

<ROPOP> -> <E> <ROPOPD> <ROPOP> -> E		
<E> -> <T> <ED>		
<ED> -> PM <T> <ED> <ED> -> E		
<T> -> <F> <TD>		
<TD> -> MDM <F> <TD> <TD> -> E		
<F> -> (<exp>) <F> -> <const> <F> -> ! <F> <F> -> <RAS>		
<SST> -> <white_st> <SST> -> <break_st> <SST> -> <for_st> <SST> -> <if_elif_else> <SST> -> <switch_st> <SST> -> <continue_st> <SST> -> <return_st> <SST> -> <assign_st> <SST> -> <decl> <SST> -> <try_catch_st>		
<while_st> -> while (<exp>) { <MST> }		
<for_st> -> for (<P1> <P2> ; <P3>) { <MST> }		
<P1> -> decl <P1> -> assign_st <P1> -> ;		
<P2> -> <exp> <P2> -> E		
<P3> -> <for_assign> <P3> -> <inc_dec> <P3> -> E		
<for_assign> -> ID <VAS>		
<VAS> -> = <exp>		

<VAS> -> . ID <for_B>		
<for_B> -> (<argS>) <FNB> <for_B> -> [<exp>] <D2DA> <Ref2> <ETEXP> <for_B> -> . ID <for_B> <for_b> -> = <exp>		
<FNB> -> . ID <for_B>		
<Ref2> -> . ID <for_B> <Ref2> -> E		
<ETEXP> -> = <exp>		
<continue-st> -> continue ;		
<break_st> -> break ;		
<return_st> -> return <expN> ;		
<expN> -> <exp> <expN> -> E		
<if_else_elif> -> if (<exp>) { <MST> } <elif_st> <else_st>		
<elif_st> -> elif (<exp>) { <MST> } <elif_st> <elif_st> -> E		
<else_st> -> else : { <MST> } <else_st> -> E		
<switch_st> -> switch (<switch_in>) { <switch_B> }		
<switch_in> -> ID <switch_in> -> <const>		
<switch_B> -> <condition> { <MST> } <switch_B> <switch_B> -> E		
<condition> -> case <const> : <condition> -> default :		
<try_catch_st> -> try { <MST> } <OTC>		
<OTC> -> catch (exception ID) { <MST> } <OTFD> <OTC> -> <OTF>		

<OTF> -> finally { <MST> }		
<OTFD> -> finally { <MST> } <OTFD> -> E		