<Structure> --> static class ID <Inherit> { <S_Cst>
--> condensed class ID <Inherit> { <C_Cst>
--> <Concrete'> class ID <Inherit> { <G_Cst>
--> symbol ID { <Int_St> } <Structure>

<EOF> --> ~

<S_Cst> --> } <Structure>
--> function <PubPriv'> static <Function_Sig> <S_Cst>
--> private static <Class_Vars> <S_Cst>
--> <Public'> static <S_Cst'>

<S_Cst'> --> DT <S_CstDT>
--> ID <S_CstID>
--> dict <Multi_Arr> <S_Cst>
--> void Main <S_Main>

<S_CstDT> --> ID <Declare'> <S_Cst>
--> [] <S_CstDT'>
--> Main <S_Main>

<S_CstDT'> --> ID <G_Arr'> <S_Cst>
--> Main <S_Main>

<S_CstID> --> ID <Object'> <S_Cst>
--> [] <S_CstID'>
--> Main <S_Main>
--> () <Body_MST> <S_Cst>

<S_CstID'> --> ID <O_Arr'> <S_Cst>
--> Main <S_Main>

<S_Main> --> (<PL>) <BodyMST> <S_CstNM>

<S_CstNM> --> } <Main_Done>
--> function <PubPriv'> static <Function_Sig> <S_CstNM>
--> private static <Class_Vars> <S_CstNM>
--> <Public'> static <ConsVar> <S_CstNM>

<ConsVar> --> DT <N_Dec>
--> ID <ConsVar'>
--> dict <Multi_Arr>

<ConsVar'> --> (<AL>) <Body_MST>
--> ID <Object'>
--> [] <O_Arr'>

<C_Cst> --> } <Structure>
--> <PrivPres> <SC> <Class_Vars> <C_Cst>
--> function <AM> <Types> <C_Cst>
--> <Class_Vars> <C_Cst>
--> <Public'> <C_Cst'>

<C_Cst'> --> static <C_Cst''>
--> concrete <Class_Vars> <C_Cst>

<C_Cst''> --> DT <C_CstDT>
--> ID <C_CstID>
--> dict <Multi_Arr> <S_Cst>
--> void Main <GC_Main>

<C_CstDT> --> ID <Declare'> <C_Cst>
--> [] <C_CstDT'>

	--> Main <GC_Main>
<C_CstDT'>	--> ID <G_Arr'> <C_Cst>
	--> Main <GC_Main>
<C_CstID>	--> ID <Object'> <C_Cst>
	--> [] <C_CstID'>
	--> Main <GC_Main>
	--> (<AL>) <Body_MST> <C_Cst>
<C_CstID'>	--> ID <O_Arr'> <C_Cst>
	--> Main <GC_Main>
<GC_Main>	--> (<AL>) <BodyMST> <GC_Nst>
<GC_Nst>	--> } <Main_Done>
	--> function <AM> <Types> <GC_Nst>
	--> <PresPriv> <SC> <Class_Vars> <GC_Nst>
	--> <Public'> <GC_ConsVar> <GC_Nst>
<GC_ConsVar>	--> concrete <Class_Vars>
	--> <Static'> <ConsVar>
<G_Cst>	--> } <Structure>
	--> <PrivPres> <SC> <Class_Vars> <G_Cst>
	--> function <AM> <Types> <G_Cst>
	--> <Public'> <G_Cst'>
<G_Cst'>	--> concrete <Class_Vars> <G_Cst>
	--> <Static'> <G_Cst''>
<G_Cst''>	--> DT <G_StDT>
	--> ID <G_StID>
	--> dict <Multi_Arr> <S_Cst>
	--> void Main <GC_Main>
<G_CstDT>	--> ID <Declare'> <G_Cst>
	--> [] <G_CstDT'>
	--> Main <GC_Main>
<G_CstDT'>	--> ID <G_Arr'> <G_Cst>
	--> Main <GC_Main>
<G_CstID>	--> ID <Object'> <G_Cst>
	--> [] <G_CstID'>
	--> Main <GC_Main>
	--> (<AL>) <Body_MST> <G_Cst>
<G_CstID'>	--> ID <O_Arr'> <G_Cst>
	--> Main <GC_Main>
<Main_Done>	--> $
	--> static class ID <Inherit> { <S_CstNM>
	--> <ConcCond'> class ID <Inherit> { <GC_Nst>
	--> symbol ID { <Int_St> } <Main_Done>
<AssignOp>	--> =
	--> CompAssign
<Ref>	--> . ID <Ref>
	--> [<Exp>] <Ref>
	--> (<PL>) . ID <Ref>
	--> $

<SST> --> <For_St>
--> <If_St>
--> <While_St>
--> <DoWhile_St>
--> <Return_St>
--> <Continue>
--> <Break>
--> <Try_St>
--> inc_dec <SP'> ID <Ref> ;
--> throw <Throw'>
--> <SP> ID <Ref> <AssignOP> <Exp> ;
--> DT <N_Dec>
--> dict <Multi_Arr>
--> ID <SST_ID>

<SST_ID> --> ID <Object'>
--> <SST_ID'>

<SST_ID'> --> . ID <SST_ID'>
--> [<Exp>] <SST_1>
--> (<PL>) <SST_2>
--> inc_dec ;
--> <AssignOp> <Exp> ;

<SST_1> --> . ID <SST_ID'>
--> inc_dec ;
--> <AssignOp> <Exp> ;
--> ;

<SST_2> --> ;
--> . ID <SST_ID'>

<Break> --> Break ;

<Continue> --> Continue ;

<While_St> --> While (<Exp>) <Body>

<Body> --> ;
--> <BodyMST>

<DoWhile_St> --> do <BodyMST> while (<Exp>)

<Try_St> --> try <BodyMST> <CatchFinally>

<CatchFinally> --> <Finally>
--> <Catch> <Finally'>

<Finally> --> finally <BodyMST>

<Finally'> --> finally <BodyMST>
--> $

<Catch> --> catch (<Exception> ID) <BodyMST> <Catch'>

<Catch'> --> <Catch>
--> $

<Swicth_St> --> swicth (ID) { <Swicth_Body> }

<Swicth_Body> --> <Case> <Default>

<Case> --> case (<Const>) : <MST> <Case>
--> $

<Default> --> default : <MST> <Case>
--> $

<If_St> --> if (<Exp>) <BodyMST> <OElse>

<OElse> --> else <OElse'>

--> $

<OElse'> --> if (<Exp>) <BodyMST> <OElse>

--> <BodyMST>

<For_St> --> For (<St1> <St2> ; <St3>) <Body>

<St1> --> <Dec>

--> <AssignSt>

--> ;

<Dec> --> DT ID <Declare'>

<AssignSt> --> <SP'> ID <Ref> <AssignOP> <Exp> ;

<St2> --> $

--> <Cond>

<St3> --> inc_dec <Var>

--> ID <For_Opt>

--> $

<Cond> --> ID <Cond'>

--> <Const> <Cond'>

<Cond'> --> $

--> ROR <Exp>

<Return> --> return <Ret'> ;

<Ret'> --> $

--> <Exp>

<For_Opt> --> . ID <For_Opt>

--> [<Exp>] <For_Opt'>

--> (<PL>) . ID <For_Opt>

--> inc_dec <For_Opt''>

--> <AssignOp> <Exp> <For_Opt''>

<For_Opt'> --> . ID <For_Opt>

--> inc_dec <For_Opt''>

--> <AssignOp> <Exp> <For_Opt''>

<For_Opt''> --> $

--> , ID <For_Opt>

<Inherit> --> <Expands> <Applies>

<Expands> --> Expands ID

--> $

<Applies> --> Applies ID <Applies'>

--> $

<Applies'> --> , ID <Applies'>

--> $

<Concrete'> --> Concrete

--> $

<Int_St> --> <Public'> <RT> ID (<AL>) { } ;

--> $

<Public'> --> Public

--> $

<RT'> --> [ ]

--> $

<RT> --> DT <RT'>
--> ID <RT'>
--> Void
--> dict [ ]

<AL> --> $
--> <AL'>

<AL'> --> ID <AL'''> ID <AL''>
--> DT <AL'''> ID <AL''>
--> dict [ ] <AL''>

<Al''> --> $
--> , <AL'>

<AL'''> --> $
--> [ ]

<<PubPriv'> --> Public
--> Private
--> $

<Function_Sig> --> <RT> ID (<AL>) <Body_MST>

<Class_Vars> --> DT <N_Dec>
--> ID <O_Dec>
--> dict <Multi_Arr>

<Multi_Arr> --> [ ] ID <<Multi_Arr'>

<Multi_Arr'> --> ;
--> , ID <Multi_Arr'>
--> = <Init_Multidim> <Multi_Arr'>

<Init_Multidim>--> new dict [<Init_Multidim'>]
--> ID

<Init_Multidim'>--> <Exp>]
-->] { <Val_Multidim> }

<Val_Multidim>--> ID <Val_Multidim'>
--> $

<Val_Multidim'>--> $
--> . ID

<N_Dec> --> ID <Declare'>
--> [ ] ID <G_Arr'>

<O_Dec> --> ID <Object'>
--> [ ] ID <O_Arr'>

<Declare'> --> ;
--> , ID <Declare'>
--> = <Init_List>

<G_Arr'> --> ;
--> , ID <G_Arr'>
--> = <Init_GArr> <G_Arr'>

<Init_GArr> --> ID
--> new DT [<Init_GArr'>

<Init_GArr'> --> <Exp>]
-->] { <Val_GArr> }

<Val_GArr> --> <Const> <Val_GArr'>
--> $

<Val_GArr'>	--> $
	--> , <Const> <Val_GArr'>
<Init_List'>	--> ;
	--> , ID <Declare>
<Init_List>	--> <SP'> ID <List1>
	--> <Const> <List2>
	--> (<Exp>) <List2>
	--> ! <F> <Init_List'>
<List1>	--> = <Init_List>
	--> . ID <List1>
	--> [<Exp>] <List3>
	--> (<PL>) <List2>
	--> inc_dec <List2>
	--> <List2>
<List2>	--> <T'> <E'> <C'> <B'> <A> <Init_List'>
<List3>	--> = <Init_List>
	--> . ID <List1>
	--> inc_dec <List2>
	--> <List2>
<Exp>	--> <B> <A>
<A>	--> || <B> <A>
	--> $
<B>	--> <C> <B'>
<B'>	--> && <C> <B'>
	--> $
<C>	--> <E> <C'>
<C'>	--> ROR <E> <C'>
	--> $
<E>	--> <T> <E'>
<E'>	--> PM <T> <E'>
	--> $
<T>	--> <F> <T'>
<T'>	--> MDM <F> <T'>
	--> $
<F>	--> (<Exp>)
	--> <Const>
	--> ! <F>
	--> <SP'> ID <OptF>
<SP>	--> Self .
	--> Parent .
<SP'>	--> Self .
	--> Parent .
	--> $
<SC>	--> Static
	--> Concrete
	--> $
<OptF>	--> . ID <OptF>

	--> [<Exp>] <OptF>
	--> (<PL>) <OptF2>
	--> inc_dec
	--> $
<OptF1>	--> inc_dec
	--> . ID <OptF>
<OptF2>	--> . ID <OptF>
	--> $
<BodyMST>	--> { <MST> }
<MST>	--> <SST> <MST>
	--> $
<Object'>	--> ;
	--> , ID <Object'>
	--> = <Init_Obj> <Object'>
<Init_Obj>	--> new ID (<PL>)
	--> ID
<O_Arr'>	--> ;
	--> , ID <O_Arr'>
	--> = <Init_OArr> <O_Arr'>
<Init_OArr'>	--> ID
	--> new ID [<Init_OArr'>
<Init_OArr'>	-->] { <Val_OArr> }
	--> <Exp>]
<Val_OArr>	--> new ID <PL>) <Val_OArr'>
	--> $
<Val_OArr'>	--> , new ID (<PL>) <Val_OArr'>
	--> $
<PL>	--> $
	--> <Exp> <PL'>
<PL'>	--> , <Exp> <PL'>
	--> $
<Function_Sig>	--> <RT> ID (<AL>) <BodyMST>
<AM>	--> Public
	--> Private
	--> Preserved
	--> $
<Types>	--> Condensed <RT> ID (<AL>) ;
	--> <SC> <Function_Sig>
<PresPriv>	--> Preserved
	--> Private
<ConcCond'>	--> Concrete
	--> Condensed
	--> $
<Const>	--> Int_Const
	--> Float_Const
	--> String_Const
	--> Char_Const
	--> Bool_Const