START:

<S> →	<class_dec> <S> | ε
First S =	{public, private, abstract, class, ε }
Follow S =	{ $ }
Selection =	{ public, private, abstract, class, $ }

Expression:

<OE> →	<AE> <OE’>
First<OE> =	{this., ID , num, deci , alpha , str, Boolean , null, ++ --, !, [}
follow<OE>=	{], : , , ,(}
selection =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [}

<OE’> →	OR <AE> <OE’>
<OE’> →	E
First<OE’>=	{OR, ε }
follow<OE’>=	{], : , , , (}
selection =	{ OR,], : , , , (}

<AE> →	<RE> <AE’>
<AE’> →	AND <RE><AE’>
<AE’> →	E
First<AE’>=	{AND , ε }
follow<AE’>=	{ OR,], : , , ,(}
selection =	{ AND, OR,], : , , ,(}

<RE> →	<E> <RE’>
First<RE>=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , [}
follow<RE>=	{ AND , OR,], : , , ,(}
selection =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , [}

<RE’> →	ROP <E> <RE’>
<RE’> →	ε
First<RE’>=	{ ROP, ε }
follow<RE’>=	{ AND , OR,], : , , ,(}
selection =	{ ROP, AND , OR,], : , , ,(}

<E> →	<T> <E'>
First<E>=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , [}
follow<E>=	{ ROP , AND , OR,], : , , ,(}
selection=	{this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , [}

<E'> →	PM <T> <E'>
<E'> →	ε
First<E'>=	{PM, ε }
follow<E'>=	{ROP , AND , OR,], : , , ,(}
selection =	{ PM, ROP , AND , OR,], : , , ,(}

<T> →	<F><T'>
First<T>=	{this., ID , num, deci , alpha , str, Boolean , null, ++ --, !, [}
follow<T>=	{PM, ROP , AND , OR,], : , , ,(}
selection=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [}

<T’> →	MDM <F> <T’>
<T’> →	ε
First<T’>=	{MDM, ε }
follow<T’>=	{ PM, ROP , AND , OR,], : , , ,(}
selection=	{ MDM, PM, ROP , AND , OR,], : , , ,(}

<F> →	<Th> ID <F’>
<F> →	<const>
<F> →	[<OE>]
<F> →	<inc_dec> ID
<F> →	!<F>
First<F> =	{this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , [}
follow<F>=	{MDM , PM, ROP , AND , OR,], : , , ,(}
selection =	{this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , [}

<F’> →	[<PL>]
<F’> →	< inc_dec>
<F’> →	ε
First<F’> =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, C }
follow<F’>=	{ MDM , PM, ROP , AND ,], : , , ,(}
selection =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !,

<Th> →	this.
<Th> →	ε
First<Th>=	{this., ε}
follow<Th>=	{ID}
Selection=	{this. , ID}

Single Statement(SST) & Multi Statement(MST):

<SST> →	*<dec>*
<SST> →	*<switch_case>*
<SST> →	*<array>*
<SST> →	*<2d-array>*
<SST> →	*<func_dec>*
<SST> →	*<for_st>*
<SST> →	*<if_else>*
<SST> →	*<Array-L>*
<SST> →	*<While_st>*
<SST> →	*ID <COM>*
<SST> →	*<inc_dec_st>*
<SST> →	*end:*
<SST> →	*jump:*
<SST> →	*ID <X>:*
<SST> →	*this.ID:*
First<SST>=	*{switch, DT, public, private, void, if, ArrayList, for, while, ID,++,--, end: jump:, this.ID:}*
follow<SST>=	*{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }*
selection =	*{ switch, DT, public, private, void, if,*

	ArrayList, for, while, ID,++,--, end: jump:, this.ID:}

<COM> →	<obj_dec_list>
<COM>→	<X> <D>
First<com> =	{.ID,ID,=,+=,-=,/=,*=,%=,++,--,]}
follow<com>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{.ID,ID,=,+=,-=,/=,*=,%=,++,--,]}

<D> →	<assign_opr> <OE>:| <inc_dec> :
First<D> =	{ =,+=,-=,*=,/=,%=,++,--}

Follow<D> =	*{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, c end:, jump:, this.ID: ,) }*
Selection=	*{ =,+=,-=,*=,/=,%=,++,--}*

<body> →	*:*
<body> →	*(<MST>)*
First<body> =	*{ :, (}*
follow<body>=	*{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) ,] }*
selection =	*{ :, (}*

<MST>→	*<SST><MST>*
<MST>→	*ε*

First<MST>=	{ switch, DT, public, private, void, if, ArrayList, for,while,ID,++,--, end: jump:, this.ID: , ε }
follow<MST>=	{) }
selection =	{ switch, DT, public, private, void, if, ArrayList, for,while,ID,++,--, end: jump:, this.ID:,) }

Declaration:

<dec> →	DT ID <init> <list>
First<dec>=	{ DT }
Follow<dec>=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [, : , DT, switch , public ,private, void, if, ArrayList, for, while, end:, jump:, this.ID: ,) }
selection =	{ DT }

<init> →	<OE>
<init> →	ε
First<init>=	{=, ε}

follow<init> =	{:, , }
selection =	{=, :, ,}

<list> →	:
<list> →	, ID <init> <list>
First<list>=	{ : , ,}
follow<list>=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [, : , DT, switch , public ,private, void, if, ArrayList, for, while, end:, jump:, this.ID: ,) }
selection=	{ : , ,}

Parameter List:

<PL> →	<OE> <PL2>
First<PL>=	{ this. , ID , num, deci , alpha , str, Boolean ,

	null, ++ --, !, [}
follow<PL>=	*{] }*
selection=	*{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [}*

<PL2>→	*, <OE><PL2>*
<PL2>→	*ε*
First<PL2>=	*{ , ,ε}*
follow<PL2>=	*{] }*
selection =	*{, ,] }*

<PL3> →	*DT ID <f-list>*
First<PL3>=	*{ DT }*
follow<PL3> =	*{] }*
selection =	*{ DT }*

<f-list> →	ε | , DT ID <f-list>
First<f-list>=	{ , , ε }
follow<f-list>=	{] }
selection =	{, ,] }

IF-Else:

<if_else> →	if <OE> <body> <o_else>
First<if_else>=	{if}
follow<if_else>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{ if }

<o_else> →	else <body>
<o_else> →	ε
First<o_else>=	{else, ε}
Follow<o_else>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
Selection =	{else, DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }

Object Oriented Concept:

<obj_dec_list>→	*ID<list1>*
<obj_dec_list>→	*<list2> :*
First<obj_dec_list>=	*{ID, =}*
follow<obj_dec_list>=	*{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }*
selection =	*{ID, =}*

< list1> →	*:*
<list1> → =	*<list2> :*
<list1> →	*, ID <list1>*
First<list1>=	*{: , =, ,}*
follow<list1>=	*{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }*
selection =	*{: , , , =}*

<list2> →	new ID [<PL>]
<list2> →	ID
First<list2>=	{new , ID}
follow<list2>=	{ : }
selection =	{ new , ID }

Class Declaration:

<Class_dec> →	<AM> <Abs> class ID <extends> [<class_body>]
First<class_dec>=	{public,private,abstract,class}
follow<class_dec>=	{ $,public, private, abstract, class}
selection =	{public,private,abstract,class}

<Abs> →	abstract
<Abs> →	ε
First<Abs>=	{abstract , ε}
follow<Abs>=	{class}
selection =	{abstract , class}

<AM> →	public
<AM> →	private
<AM> →	ε
First<AM>=	{ public , private ,ε}
follow<AM>=	{abstract, class, void, DT, ID}
selection =	{ public , private , abstract, class, void, DT, ID}

< extends >→	inherit ID
< extends >→	ε
First<extends>=	{ inherit , ε}
Follow <extends> =	{ [}
Selection =	{inherit , [}

<class_body> →	<main-method> | <body>
First<class_body>=	{ main , : , (}
Follow<class_body>=	{] }
Selection =	{ main , : , (}

<main-method> →	main method [ ] <body>
first<main-method> =	{main}
follow<main-method>=	{] }
selection =	{main}

Function Declaration:

<func_dec> →	<AM> <return_type> ID [<PL3>] (<func_body>)
First<func_dec>=	{public,private,void,DT,ID}
follow<func_dec>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection set=	{public,private,void,DT,ID}

<return_type> →	Void
<return_type> →	DT
<return_type> →	ε
First<return_type>=	{ void , DT , ε}
follow<return_type>=	{ID}
selection =	{void , DT, ID}

<func_body> →	<MST> <ret_line>
<func_body> →	:
First<func_body>=	{ switch, DT, public, private, void, if, ArrayList, for,while,ID,++,--, end: jump:, this.ID:, : , ε }
follow<func_body>=	{) }
selection =	{ switch, DT, public, private, void, if, ArrayList, for,while,ID,++,--, end: jump:, this.ID:, : ,) }

<ret_line> →	return <ret>:
First<ret_line>=	{return}
follow<ret_line>=	{) }
selection =	{ return }

<ret> →	<OE>
<ret> →	E
First<ret>=	{this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , ε }
follow<ret>=	{ : }
selection =	{ First<ret>={this. , ID , num, deci , alpha , str, Boolean , null, ++ --, ! , ε }

X:

<X> →	.ID<X> | [<index>] <X3> | [<PL>]<X2>
<X> →	ε
First<X>=	{ .ID, [, ε}
Follow<X> =	{ ++, --, : , =, += ,-= , *= , /=, %=,] }
Selection =	{.ID, [++, --, : , =, += ,-= , *= , /=, %=,] }

<X2> →	.ID<X>
First<X2> =	{.ID}
Follow<X2> =	{ ++, --, : , =, += ,-= , *= , /=, %=,] }
Selection =	{.ID}

<X3> →	.ID<X>| ε
First<X3> =	{.ID, ε }
Follow<X3> =	{++, --, : , =, += ,-= , *= , /=, %=,]}
Selection =	{.ID, ++, --, : , =, += ,-= , *= , /=, %=,]}

<index> →	int_const
First<index> =	{int_const}
Follow<index>=	{] }
Selection =	{int_const}

Assign Value:

<assign_st> →	ID <X> <assign_opr> <OE>:
First<assign_st>=	{ID}
Follow<assign_st>=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [, : }
Selection =	{ ID }

<assign_opr> →	=
<assign_opr> →	+=
<assign_opr> →	-=
<assign_opr> →	*=
<assign_opr> →	/=
<assign_opr> →	%=
<assign_opr> →	=
<assign_opr> →	+=

<assign_opr> →	-=
<assign_opr> →	*=
First<assign_opr>=	{ =,+=,-=,*=,/=,%= }
follow<assign_opr> =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [}
Selection=	{=,+=,-=,*=,/=,%= }

Increment Decrement Operator:

<inc_dec_st> →	<inc_dec> ID <X>:
<inc_dec_st> →	ID <X> <inc_dec> :
First<inc_dec_st> =	{++,--,ID}
Follow<inc_dec_st>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
Selection =	{++,--,ID}

<inc_dec > →	++
<inc_dec > →	--
First<inc_dec> =	{++,--}
follow<inc_dec> =	{: , ID ,] }
selection =	{++,--}

Constant:

<const> →	num
<const> →	deci
<const> →	alpha
<const> →	str
<const> →	Boolean
<const> →	Null
First<const> =	{num.deci,alpha,str,null,boolean}
follow<const>=	{ MDM , PM, ROP , AND , OR,], : , , ,(,)}
selection =	{num.deci,alpha,str,null,boolean}

For Loop:

<for_st> →	for [<c1><c2> :< c3>] <body>
First<for_st> =	{for}
follow<for_st>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{for}

<c1> →	<dec>
<c1> →	<assign_st>
<c1> →	:
First<c1> =	{DT,ID ,:}
follow<c1>=	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [:}
selection =	{DT,ID ,:}

<c2> →	<OE>
<c2> →	ε
First<c2> =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [, ε}
follow<c2>=	{ : }
selection =	{ this. , ID , num, deci , alpha , str, Boolean , null, ++ --, !, [, :}

<c3> →	ID <X> <c4>
<c3> →	<inc_dec> ID <X>
<c3> →	E
First<c3> =	{ID,++,--,ε}
follow<c3>=	{] }
selection =	{ID,++,--,] }

<c4> →	<assign_opr> <OE>
<c4> →	<inc_dec>
First<c4> =	{=,+=,-=,*=,/=,%=,++,--}
follow<c4>=	{] }
selection =	{=,+=,-=,*=,/=,%=,++,--}

While Loop:

<While_st> →	while <OE> <body>
First<while_st> =	{while}
follow<While_st>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{while}

Array Declaration:

1D array:

<array> →	DT ID [] <arr_list>
First<array> =	{DT}
follow<array>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{ DT }

<arr_list> →	:
<arr_list> →	= (<const> <arr2_list>)
First<arr_list> =	{:,=}
follow<arr_list>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{ :,= }

<arr2_list> →	, <const> <arr2_list>
<arr2_list> →	E
First<arr2_list> =	{, , ε }
follow<arr2_list>=	{) }
selection =	{ , ,)}

2D array:

<2d-array> →	DT ID [] [] <B1>
First<2d_array>=	{DT}
follow<2d-array>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{DT}

<B1> →	:
<B1> →	= ((<const> <B2>
First<B1> =	{ : , =}
follow<B1> =	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{ : , =}

<B2> →	, <const> <B2>
<B2> →	) <B3>
<B2> →	)):
First<B2> =	{ , ,)}

follow<B2>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{ , ,)}

<B3> →	, (<const> <B2>
First<B3> =	{,}
follow<B3>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
Selection =	{,}

Array List:

<Array-L> →	ArrayList (DT) ID = new ArrayList[]:
First<Array-L> =	{ArrayList}
follow<Array-L>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{ArrayList}

Switch Case:

<switch_case> →	switch <OE> (<case>)
First<Switch_case> =	{switch}
follow<switch_case>=	{ DT, switch , public ,private, void, if, ArrayList, for, while, ID, ++, --, end:, jump:, this.ID: ,) }
selection =	{switch}

<case> →	opt <const> : <body> end: <case> | <default_st>
First<case> =	{opt, default:}
follow<case> =	{) }
selection =	{opt, default:}

<default_st > →	default: <body>
<default_st > →	ε
First<default_st> =	{ default: , ε }
follow<default_st>=	{) }
selection =	{ default: ,) }