PROGRAM STRUCTURE

<PS> → <import_st> <classes> <main_class>

<import_st> → import <qualified_name> <import_tail>

<import_tail> → ; | . *

<qualified_name> → ID <qualified_name_tail>

<qualified_name_tail> → ε | . ID <qualified_name_tail>

<classname> → ID

<main class> → <main method>

<classes> → <class><classes>| ε

<class> → <class header><Inheritance><class body>

<class header> → <Modifiers>Class ID

<Inheritance> → extends ID| ε

<class body> → { | <Attributes> <class body>|< constructors > <class body>|< Methods > <class body>| ε

<Attributes> → <Modifiers>DT ID <Exp>

<Modifiers'> → <Access_Modifier> | static | final | abstract

<Access_Modifier> → public | private | protected

<constructor> → <constructor header><Method Body>

<constructor header>→ <Modifiers> ID(<Parameters>)

<Methods> → <Method><Methods>|ε

<Method> → <Method header><Method body>

<Method header> → <Modifiers>DT ID(<Parameter>)

<Method body> → {<M.S.T>}

<Parameters> → <Parameter><Parameter'>|ε

<Parameter'> → ε|<Parameters>

<Parameter> → DT ID

<MST> → <SST><MST>| ε

<SST> → <Exp>;|<TS >;|<ReturnSt>;|< assign_st t>;|<Dec>;|<if St>|<while St>|<for St>;|ObjCall

<Unary Opr> → inc dec |NOT

<Binary Opr> → PM |MDM|Comparison|Logical

<assign_st> → ID <Assign Opr><Exp>

<Assign Opr> → =|+=|-+|*=|/=|%=

<Method Call> → ID(<Args>)

<constructor call> → new ID (<Args>)

<Args> → <Exp><Args'> |ε
<Args'> → ε|,<Args>

<TS> → <This or Super or ID><Args>

<This or Super or ID> → this|super|ID

<Return St> → return <Exp>|return this.

<main method> → <m.m header> {<m.m body>}

<m.m body> → <MST>

< m.m header> → public static void main (Strings args[])

<object decl> → <obj header> ;

<obj header> → Type ID = <new expr>

<new expr> → new Type (<arg list opt>)

<arg list opt> → <arg list> | ε

<arg list> → Expr <arg list tail>

<arg list tail> → , <arg list> | ε

Expr → ID <expr tail> | < Const> | <new expr>

<expr tail> → (<arg list opt>) | ε

<Type> → ID

<Const> → int_const | string_const | boolean_const

<object call> → <primary expr> <access chain>

<primary expr> → ID | this | super | <new expr> | <method call>

<access chain> → <access> <access chain> | ε

<access> → . ID <access tail>

<access tail> → (<arg list opt>) | ε

<Exp> → <OE>

<OE> → <AE> <OE'>

<OE'> → OR <AE> <OE'> | ε

<AE> → <RE2> <AE'>

<AE'> → AND <RE2> <AE'> | ε

<RE2> → <RE1> <RE2'>

<RE2'> → RO2 <RE1> <RE2'> | ε

<RE1> → <E> <RE1'>

<RE1'> → RO1 <E> <RE1'> | ε

<E> → <T> <E'>

<E'> → PM <T> <E'> | ε

<T> → <F> <T'>

<T'> → MDM <F> <T'> | ε

<F> → <primary>

| - <F>

| NOT <F>

| (<OE>)

<primary> → ID

| const

| <method call>

| <constructor call>

| <assign st>

<try**>** → try { <MST> } <catch_list>
<catch_list**>** → catch (ID) { <MST> } <catch_list_tail>
<catch_list_tail**>** → catch (ID) { <MST> } <catch_list_tail> | ε
<throw**>** → throw <throw_options>:
<throw_options**>** → ID | Const | new ID (<param_list>)

<While St**>** → while (<cond>)<loop_body>
<cond**>** → <Const_or_ID> | <Const_or_ID> <ROP> <Const_or_ID> | <exp>

<ROP> → RO1 | RO2
<loop_body> → ; | <SST> | {<MST>}
<for_loop> → for (<F1><F2>;<F3>) <loop_body>
<F1> → <dt_dec> | <assign_st> | ;
<F2> → <cond> | ε
<F3> → <inc_dec> | <assign_st> | null

<if> → if (<cond>) <loop_body> <else>
<else> → else <loop_body> | null

<array_dec> → <arr_type> ID [] = { <arr_const_or_id> };

<arr_type> → DT | ID
<arr_const_or_id> → ε | <Const_or_ID> | ID , | Const ,

<dt_dec> → <var_init> <var_init_tail> ;
<var_init> → = <Const_or_ID> | ε
<var_init_tail> → , ID <var_init> <var_init_tail> | ε