## Declaration:

<DEC> : DT ID <init> <list>

<INIT> : = <init2> | e

<INIT2> : <const> | ID <init>

<const> : int-const |str-const | bool-const | char-const | double-const

<list> : , ID <init> <list> | ;

## Loops:

**For Loop:**

<For\_st> : for : <c1> <c2> ; <c3> : <body>

<c1> : <dec> | <asgn\_st> | ;

<c2> : <OE> | e

<c3> : ID <X> <c4> | inc\_dec ID <X> | e

<c4> : <asgn\_op> <E> | inc\_dec

**<Body>** : <SST> | ( <MST> ) | ;

<SST> : <For\_st> | <while\_st> |<foreach\_st> |<if\_else> | <return\_st>| <ArrList\_st> | DT<SST03> |inc\_dec ID <X> ; |ID <SST01>|e

<SST01> : ID = new ID :<Arg\_st>: ; | <Y> :<Argu\_list>: ;

|<X> <SST02>

<SST02> : <asgn\_op> <exp>; | inc\_dec;

<SST03> : ID <init> <list> | [] ID <list3> ; |

<MST> : <SST><MST> | e

<obj\_dec> **:** ID ID = new ID :<Arg\_list>: ;

**<Arg\_list>** : <OE> <Arg\_list2> | e

<Arg\_list2> : ,<OE><Arg\_list2> | e

**<funct\_call>** : ID <Y> : <Argu\_list> : ;

<Y> : .ID <Y> | e

**<Asgn\_st>** : ID <X> <asgn\_op> <E>;

<Asgn\_op> : = | <CoP>

<CoP> : RO | AO

<X> : <Handle\_Array> <Handle\_Dot>

<X> : e

<Handle\_Array> : [int-constant] | e

<Handle\_Dot> : .ID<X2> | e

<X2> : <X> | :<Argu\_st>: <Handle\_Dot>

<Inc\_Dec\_st> : inc\_dec ID <X> ; | ID <X> inc\_dec;

<OE> : <AE><OE2>

<OE2> : || <AE> <OE2> | e

<AE> : <RE> <AE2>

<AE2> : && <RE> <AE2> | e

<RE> : <E> <RE2>

<RE2> : RO <E> <RE2> | e

<E> : <T> <E2>

<E2> : PM <T> <E2> | e

<T> : <F> <T2>

<T2> : MDM <F> <T2> | e

<F> : <Th> ID <F1> | <Const> | ( <OE> ) | INC\_DEC ID | ! <F>

<F1> : Ɛ | :<Argu\_St>: | INC\_DEC | [ <OE> ]

<Th> : this. | Ɛ

<ID\_const> : ID | <const>

<print\_st> : print(<ID\_const>);

**Foreach Loop:**

<foreach\_st> : foreach : DT ID in <DS> : <body>

<DS> : <Arr\_St> | <ArrList\_St>

**While Loop:**

<While\_st> : while : <OE> : <body>

## Conditions:

**IF\_Else:**

<if\_else> : if : <OE> : <body> <elif\_st>

<elif\_st> : elif : <OE> : <body> | e | <else>

<else> : else <body> | e

## Function:

<funct\_st> : <static\_st> <DT\_Void> ID : <param> : <body>

<static\_st> : static | e

<DT\_Void> : void | DT

<param> : DT ID <multiParam> | e

<MultiParam> : , DT ID <MultiParam> | e

<return> : ret <const>; | e

**Abstract Function:**

<abst\_funct> : <AccessModifier2> abstract <DT\_void>

ID : <param> : ;

**Data Structure:**

**<Arr\_St>** : DT[] ID <list3> ;

<list3> : = <list2> | e

<list2> : new DT [int] | { <const> <multi\_value> }

<Multi\_Value> : , <const> <multi\_value> | e

// <const> of same DT

**<ArrList\_St>** : ArrayList ID <list4> ;

<list4> : = new ArrayList() <list5> | e

<list5> : { <const> <multi\_value> } | e

### OOP:

**Class:**

<class\_st> : <AccessModifier> <static\_abstract> class ID <inherit\_inter> <body2>

<AccessModifier> : AccessModifier | e

<static\_abstract> : static | abstract | e

<Body2> : { <MST2> }

<MST2> : <SST2><MST2> |e

<SST2> : <body> | <funct\_st> | <class\_st> | e

<inherit\_inter> : <inherit\_st> <interface\_imple> | <interface\_imple>

**Inheritance:**

<inherit\_st> : (:) ID | e

**Polymorphism:**

<Poly\_st> : <AccessModifier2> VO <funct\_st2>

<AccessModifier2> : public | protected

<funct\_st2> : <DT\_Void> ID : <param> : <body>

**Interface:**

<interface\_imple> : <col\_comma> ID <interface\_imple> | e

<col\_comma> : : | ,

<Interface\_st> : interface ID <Body3>

<Body3> : { <MST3> }

<MST3> : <SST3><MST3> | e

<SST3> : <funct\_St3> | e

<funct\_st3> : <DT\_void> ID : <params> : ;

### 