**Attributed: 0.4**

**Rexton Keys:**

<IF\_Statement>🡪 agar

<IF\_ELSE>🡪 agar Nahi To

<ELSE>🡪warna

<WHILE\_Statement>🡪 Jab tak

<FOR\_Loop>🡪 Bar Bar

**General Rules:**

T <CONST> 🡪 INT\_CONSTT=int| FLOAT\_CONSTT=float | STRING\_CONSTT=string | CHAR\_CONSTT=char | BOOL\_CONSTT=bool

TM<Static>🡪 Static TM=Token.CP

AM<Access\_Modifier> 🡪access\_modifier AM=Token.VP | Null AM=”Private”

RT<Return\_Type> 🡪 void RT=”VOID” | DTRT=Token.VP

<M\_ST>🡪<S\_ST><M\_ST> | Null

<Body>🡪 ; | <S\_ST> | {<M\_ST>}

AL<List\_Param>PL 🡪 DTT=Token.VP **[PL+=T]** IDN=Token.VP AL<List\_Param1>PL | Null **[AL=PL]**

AL<List\_Param1>PL🡪 , DTT=Token.VP **[PL+= “,”+T]** IDN=Token.VP AL<List\_Param1>PL | Null **[AL=PL]**

//for object array declaration

//AL<Param> PL 🡪 T<Exp>**[PL+=T]** AL<Param1> PL | Null AL=PL

AL <Param1> PL 🡪 , T<Exp> **[PL+= “,”+T]** AL<Param1> PL | Null AL=PL

RT<Method\_Call\_1>N🡪 ( **[PL=””]** AL<Param>PL) **[RT = lookup(N,AL)**

**If(RT == Null) “Undeclared Method”]**

**Single Statement:**

RT<S\_ST>🡪<Jab\_Tak> | DTT=Token.VP <S\_St\_DT>T | <Bar\_Bar> | <agar\_warna> | RT<Return> | inc\_dec IDN=Token.VP **[lookup(N,S) == false “Undeclared Variable”]**<inc\_dec\_list>;|ID N=Token.VP , T=Token.CP <S\_St\_ID>N,T | <break> | <continue> |<this>

<S\_St\_ID>N,T🡪**[if(lookup(N,S) == false) “Undeclared Variable”]** inc\_dec; |**[if(lookup(N,S) == false) “Undeclared Variable”]** <Assign\_Op>T|**[if(Clookup(N) == false**) **“Undeclared Class”]** <Object\_link>N | **[if(lookup(N,S) == false) “Undeclared Object”]** <Object\_Call>; | RT<Method\_Call\_1>N ; |**[if(lookup(N,S) == false) “Undeclared Array”]** [T<Exp>] <Assign\_Op>T

<S\_St\_DT>T🡪IDN=Token.VP **[dlookup(N,S) == false “Redeclaraion Error”**

**Else INSERT(N,T,S)]** <S\_St\_DT2>T | <Array\_DEC> T

<S\_St\_DT2> T 🡪<Variable\_Link2> T

**Variable Declaration:**

<DEC>🡪DTT=Token.VP <Variable\_Link>T

<Variable\_Link>T🡪 IDN=Token.VP  **[if(dlookup(N,S) == true ]** <Varaiable\_Link2> T

**“Redeclaration Error”**

**Else**

**INSERT(N,T,S)**

<Variable\_Link2>T🡪AOPOP=Token.VP <Variable\_Value>T,OP| <LIST>T

<Variable\_Value> T,OP 🡪 ET<Exp>**[if(Comp(T,ET,OP) == false ]** <LIST> T

**“Type Mismatch”**

<LIST> T 🡪 , ID N=Token.VP  **[if(dlookup(N,S) == true ]** <Varaiable\_Link2> T  | ;

**“Redeclaration Error”**

**Else**

**INSERT(N,T,S)**

**Assignment of Variable:**

<Assign\_Op>T🡪 AOPOP <Assign\_Op2> T,OP

<Assign\_Op2> T,OP 🡪 ET<Exp>**[if(Comp(T,ET,OP) == false ]** ;

**“Type Mismatch”**

**If Else Condition:**

<agar\_warna>🡪 agar (ET<Exp>**[if(ET != BOOL “Error”]**) {**CS()**<M\_ST>}**DS()** <O\_Else>

<O\_Else>🡪warna {**CS()**<M\_ST>}**DS()** | Null

**While Statement:**

<Jab\_tak>🡪jabtak (ET<Exp>**[if(ET != BOOL “Error”]**) <Body>

**Return:**

RT<Return> 🡪 return RT<Return2>

RT<Return2>🡪 ; **[RT=”NULL”]**| T<Exp>;

**Break:**

<Break>🡪break ;

**Continue:**

<Continue>🡪continue ;

**This:**

<this>🡪 this.IDN=Token.VP,T=Token.CP **[if(lookup(N,S) == NULL) “Undeclared”]** <LIST1>T

<LIST1> T 🡪 ; | AOPOP=Token.VP <LIST2> T,OP

//<LIST2> T,OP 🡪 <Exp>ET=Token.CP **[Comp(T,ET,OP) == false) “Type Mismatch”]** ;

**Class Declaration:**

<Class\_Dec>🡪AM<Access\_Modifier><Class\_Link>AM

<Class\_Link>AM🡪 class IDN=Token.VP PN<Class\_Base> **[if(Clookup(N) == NULL) INSERT(N,AM,PN)**

**Else “Class Redeclaration” ]** {**CS()**<Class\_Body>}**DS()** <Second\_Class>

<Second\_Class> 🡪 <Class\_Dec> | Null

N<Class\_Base>🡪 Null | : IDN=Token.VP **[if(Clookup(N) == NULL) “Class Undeclared”]**

<Class\_Body>🡪<Class\_Member><Class\_Body> | Null

<Class\_Member>🡪AM<Access\_Modifier> <Member\_Link>AM

<Member\_Link>AM🡪 TM<Static><SS\_A>AM,TM| void **[RT=”Void”]** IDN=Token.VP  **[TM=””]**<Method\_Link 3> AM,TM,RT,N | DTT=Token.CP <DT\_2>AM,T|IDN=Token.VP  **[if(Clookup(N) == false) “Undeclared Class”]** <Object\_Constructor\_DEC> AM,N

<Object\_Constructor\_DEC> AM,N 🡪 <object\_link> AM, N | <Constructor\_DEC> AM,N

<DT\_2> T🡪IDN=Token.VP <ID\_1> AM,T,N | <Array\_DEC> AM,””,T

<ID\_1> AM,RT,N 🡪<Varaiable\_Link2>T | **[TM=””]** <Method\_Link 3> AM,TM,RT,N

<SS\_A>AM,TM🡪 DTRT=Token.CP <DT\_1>AM,TM,RT |void **[RT=”Void”]** IDN=Token.VP<Method\_Link 3> AM,TM,RT,N

<DT\_1> AM,TM,RT 🡪IDN=Token.VP<ID\_2> AM,TM,RT,N | <Array\_DEC> AM,TM,RT

<ID\_2> AM,TM,RT,N 🡪<Method\_Link 3> AM,TM,RT,N | <Variable\_Link2>T

<DT\_A>🡪<Variable\_Link2> T |<Method\_Link 3>

<Id\_A>🡪<Method\_Link 3> | <Object\_Creation\_Exp>

**Constructor Declaration:**

<Constructor\_DEC>AM,N🡪 (**[PL=””]** AL<List\_Param>PL) **[TM=””]** **[RT = Clookup(N,AL) ]** {**CS()**<M-St>}**DS()**

**if(RT == Null) “Redeclaration Constructor”**

**else INSERT(N,AL,AM,TM)**

**Array Declaration:**

<Array\_DEC> AM,TM,RT 🡪 []**[T+=”[]”]** IDN=Token.VP **[if(dlookup(N) == false) “” ]** <INIT\_Array>T

**“Redeclaration Error”**

**Else**

**INSERT(N,RT,AM,TM)**

<INIT\_Array>T 🡪 ; | = new DTT2=Token.VP [ET<Exp>] **[T2+=”[]”] [if(T2!=T) “Array Type Mismatch”]** <Array\_const>

<Array\_const>🡪<Array\_C>| ;

<Array\_C>🡪{ <Exp><Array\_C2>

<Array\_C2>🡪 , <Exp> <Array\_C2>| } ;

**Method Declaration:**

<Method\_Link 3> AM,TM,RT,N 🡪 (**[PL=””]** AL<List\_Param>PL) **[PL+=”🡪”+RT]** **[RT2 = lookup(N,AL)**

**if(RT2 == Null) “Undeclared Method”**

**else**

**INSERT(N,PL,AM,TM)** **]**

{**CS()** <M-St> }**DS()**

**Object Declaration:**

<Object\_Link>AM,N🡪 IDN1=Token.VP **[if(Lookup(N1) == false) “Redeclaration Error”]** <Object\_Creation\_Exp>N,N1,AM| [] IDN1=Token.VP <object\_array\_dec>N,N1,AM

<object\_array\_dec> N,N1,AM 🡪 = new IDN2=Token.VP **[IF(Clookup(N2) == false “Class Unspecified”]** [T<Exp>] **[if(Lookup(N1) == false) “Redeclaration Error” else TM= ”” INSERT(N1,N,AM,TM)]** <obj\_arr\_dec1>

<obj\_arr\_dec1> 🡪 ;| {<obj\_arr\_dec2>

//<obj\_arr\_dec2> 🡪 new IDN=Token.VP **[IF(Clookup(N) == false “Class Unspecified”]** (AL<Param>PL) <obj\_arr\_dec3>

<obj\_arr\_dec3> 🡪 , <obj\_arr\_dec2>|};

<Object\_Creation\_Exp>N,N1,AM 🡪 = new IDN2=Token.VP **[IF(Clookup(N2) == false “Class Unspecified”]** (AL<Param>PL) **[if(Lookup(N1) == false) “Redeclaration Error” else TM= ”” INSERT(N1,N,AM,TM)]**  <Object\_List>N,AM

//<Object\_List>N,AM 🡪 , IDN1=Token.VP **[if(Lookup(N1) == false) “Redeclaration Error”]** <Object\_Creation\_Exp>N,N1,AM | ;

**Object Calling:**

<Object\_Call>🡪 . T<Exp> |[ T<Exp>]. T<Exp>

**FOR Loop:**

<Bar\_Bar>🡪barbar(<F1>; <F2>; <F3>) <Body>

<F1>🡪<DEC> |IDN=Token.VP , T=Token.CP **[lookup(N,S) == false “Undeclared Variable”]** <Assign\_Op>T | Null

<F2>🡪 T<Exp> **[if(T!=BOOL) “Error”]** <X> | Null

<X>🡪 , T<Exp>**[if(T!=BOOL) “Error”]** <X> | Null

<F3>🡪inc\_dec IDN=Token.VP **[lookup(N,S) == false “Undeclared Variable”]** | IDN=Token.VP ,T=Token.CP **[lookup(N,S) == false “Undeclared Variable”]** <F4> T | Null

<F4>T🡪inc\_dec | AOPOP=Token.VP <Exp>ET **[Comp(T,ET,OP) == false “Type Mismatch”]**

**Expression:**

T<Exp> 🡪 T<OR\_Exp>

T2<OR\_Exp>🡪 T<AND\_Exp> T2<OR\_Exp2>T

T3<OR\_Exp2>T🡪 ||OP=Token.VP RT<AND\_Exp> **[T2=Comp(T,RT,OP)** T3<OR\_Exp2>T2 | Null **[T3=T2]**

**If(T2==Nulll)**

**“Type Mismatch” ]**

T2<AND\_Exp>🡪 T<ROP> T2<AND\_Exp2>T

T3<AND\_Exp2>T🡪 &&OP=Token.VP RT<ROP> **[T2=Comp(T,RT,OP)** T3<AND\_Exp2>T2 | Null **[T3=T2]**

**If(T2==Nulll)**

**“Type Mismatch” ]**

T2<ROP>🡪 T<E> T2<ROP2>T

T3<ROP2>T🡪 ROPOP=Token.VP RT<E> **[T2=Comp(T,RT,OP)** T3<ROP2>T2 | Null **[T3=T2]**

**If(T2==Nulll)**

**“Type Mismatch” ]**

T2<E>🡪 T<T> T2<E2>T

T3<E2 >T🡪Plus\_MinusOP=Token.VP RT<T> **[T2=Comp(T,RT,OP)** T3<E2>T2| Null **[T3=T2]**

**If(T2==Nulll)**

**“Type Mismatch” ]**

T3<T>🡪 T<F> T3<T2>T

T3<T2>T🡪 M\_D\_MOP=Token.CP RT<F> **[T2=Comp(T,RT,OP)** T3<T2>T2 | Null **[T3=T2]**

**If(T2==Nulll)**

**“Type Mismatch” ]**

RT<F>🡪 IDN=Token.VP,T=Token.CP RT<id\_op>N,T | T<Const> **[RT=T]**|!OP=Token.CP T<F>

**[T1 = compatibility(T,OP)**

**If(T1 == Null)**

**“Type Mismatch” ] [RT=T1]** |

(T<Exp>)**[RT=T]** | Inc\_Dec IDN=Token.VP , T=Token.CP RT<inc\_dec\_list>N,T

RT<inc\_dec\_list>N,T1 🡪 [T<Exp>] **[if(lookup(N) == Null “Undeclared Array” else RT=T1]** | **[if(lookup(N) == Null “Undeclared Object” else RT=T1]** .IDN1=Token.VP | [T<Exp>] **[if(lookup(N) == Null “Undeclared Array” else RT=T1 ]**  |Null **[if(lookup(N) == Null “Undeclared Variable” else RT=T1]**

RT<id\_op>N,T 🡪 Null **[RT=T]** **[if(lookup(N) == Null “Undeclared Variable”]** | RT<Method\_Call\_1>N | [ ET<Exp> ] **[if(lookup(N) == Null “Undeclared Array”]** **[RT=T]** |RT<Member\_exp>N |**[if(lookup(N) == Null “Undeclared Variable”]** Inc\_Dec **[RT=T]**

RT<Member\_exp>N 🡪 **[if(lookup(N) == Null “Undeclared Object”]** .IDN1=Token.VP,T1=Token.CP RT<Member\_exp\_2>N1,T1

RT<Member\_exp\_2>N,T 🡪 Null **[RT=T] [if(lookup(N) == Null “Undeclared Variable”]** | RT<Method\_Call\_1>N | [T<Exp>] **[RT=T][if(lookup(N) == Null “Undeclared Array”]**