AST Node type for each rule

```
Program ::=  IDENT  {  Dec* Stmt* }         Program
Dec ::=  Type  IDENT  ;                      Dec
Type ::=  image
          | pixel
          | int
          | boolean
Stmt ::=  ;
          | AssignStmt
          | PauseStmt
          | IterationStmt
          | AlternativeStmt
AssignStmt ::= IDENT  =  Expr  ;             AssignExprStmt
          |  IDENT  =  Pixel  ;              AssignPixelStmt
          |  IDENT = STRING_LIT ;            FileAssignStmt
          | IDENT  .  pixels  [ Expr , Expr ]  =  Pixel  ;   SinglePixelAssignmentStmt
          | IDENT  .  pixels  [ Expr , Expr ] (red | green | blue ) = Expr  ;  SingleSampleAssignmentStmt
          | IDENT  . shape  = [ Expr , Expr ]  ;           ShapeAssignmentStmt
          | IDENT  . location  = [ Expr , Expr ]  ;       ScreenLocationAssignmentStmt
          | IDENT  .  visible  =  Expr ;               SetVisibleAssignmentStmt
Pixel  ::=  {{  Expr  ,  Expr  , Expr }}          Pixel
Expr  :: =  OrExpr  (  ϵ |  ?  Expr  :  Expr )    ConditionalExpr
OrExpr ::=  AndExpr  (  |  AndExpr )*  BinaryExpr, see lecture
AndExpr  ::=  EqualityExpr (  &  EqualityExpr )*  BinaryExpr, see lecture
EqualityExpr  ::=  RelExpr  (  ( == | != )  RelExpr ) *  BinaryExpr, see lecture
RelExpr ::= ShiftExpr  (  ( < | > | ≤ | ≥)  ShiftExpr ) *  BinaryExpr, see lecture
ShiftExpr ::= AddExpr  (  (  ≪  |  ≫  )  AddExpr )*  BinaryExpr, see lecture
AddExpr ::= MultExpr (  ( + | - )  MultExpr ) *  BinaryExpr, see lecture
MultExpr  ::=  PrimaryExpr  (  ( * | / | % )  PrimaryExpr )*  BinaryExpr, see lecture
PrimaryExpr ::= IDENT    IdentExpr
          |  INT_LIT         IntLitExpr
          |  BOOLEAN_LIT          BooleanLitExpr
          | x                  PreDefExpr
          | y                  PreDefExpr
          | Z                  PreD efExpr
          | SCREEN_SIZE       PreDefExpr
          | (  Expr  )           whatever Expr yields
          | IDENT [  Expr  ,  Expr  ]  (red | green | blue )    SampleExpr
```

```
        | IDENT  .  height        ImageAttributeExpr
        | IDENT  .  width         ImageAttributeExpr
        | IDENT  .  x_loc         ImageAttributeExpr
        | IDENT  .  y_loc         ImageAttributeExpr
PauseStmt ::= pause  Expr  ;      PauseStmt
IterationStmt ::= while  (  Expr ) { Stmt* }          IterationStmt
AlternativeStmt ::= if ( Expr ) {  Stmt *}            AlternativeStmt   with empty elseStmtList
        | if ( Expr )  { Stmt* }  else  { Stmt* }      AlternativeStmt
```