Code Generation for assignment 5

AST Node Type	AST rule	Constraints and Actions
Prog	Program ::= IDENT Dec* Stmt*	This has already been (mostly) implemented. Set up main method Visit children Finalize main method Return bytecode
Dec	Dec ::= Type IDENT	This has already been implemented. Create a static or local variable (your choice) for this variable. If it is an image, instantiate a PLPImage object.
AlternativeStmt	Stmt ::= Expr $(\text{Stmt*})_{\text{ifStmtList}}$ $(\text{Stmt*})_{\text{elseStmtList}}$	*Assignment 6*
AssignExprStmt	Stmt ::= IDENT Expr	Visit the expression to generate code to leave its value on top of the stack. Store it in the variable on the left hand side.
AssignPixelStmt	Stmt ::= IDENT Pixel	*Assignment 6*
FileAssignStmt	Stmt ::= IDENT FileName	This has already been implemented for you Acquire a BufferedImage indicated by the filename or URL using the PLPImage.loadImage method and store it into the PLPImage object
ScreenLocationAssignmentStmt	Stmt ::= IDENT $\text{Expr}_{\text{xScreenExpr}}$ $\text{Expr}_{\text{yScreenExpr}}$	Visit the expressions. Store the values in the appropriate fields of the image indicated by the IDENT. Invoke the image's updateFrame method.
SetVisibleAssignmentStmt	Stmt ::= IDENT Expr	This has already been implemented for you Visit the expression. Store the value in the isVisible field of the image indicated by the IDENT. Invoke the image's updateFrame method.
ShapeAssignment Stmt	Stmt ::= IDENT $\text{Expr}_{\text{width}}$ $\text{Expr}_{\text{height}}$	Visit the expressions. Store the values in the width and height fields of the image indicated by the IDENT. Invoke the image's updateImageSize method. Invoke the image's updateFrame method.
SinglePixelAssignmentStatement	Stmt ::= IDENT $\text{Expr}_{\text{xExpr}}$ $\text{Expr}_{\text{yExpr}}$ Pixel	Visit the expressions to generate code to evaluate expressions indicating a location in the image indicated by the IDENT. Visit the Pixel to generate code to pack it into an integer. Update the pixel in the image and invoke the image's updateFrame method.

SingleSampleAssignmentStmt	Stmt ::= IDENT $\text{Expr}_{\text{xExpr}}$ $\text{Expr}_{\text{yExpr}}$ COLOR $\text{Expr}_{\text{rhsExpr}}$	Visit the expressions to generate code to leave their values on top of the stack. Use the PLPImage.setSample method to update the given sample of the indicated image.
IterationStatement	Stmt ::= Expr Stmt*	*Assignment 6*
PauseStatement	Stmt ::= Expr	Visit the expression to generate code to leave its value on top of the stack. Invoke the PLPImage.pause method
BinaryExpr	Expr ::= Expr_{e0} Op Expr_{e1}	Visit the expressions to generate code to leave their values on top of the stack. Evaluate the binary expression and leave its value on top of the stack. Implement the following operators on ints: +,-,*,/,%, ≪, ≫. *The rest will be done in assignment 6*.
BooleanLitExpr	Expr ::= BooleanLit	This has already been implemented. Generate code to leave the value (0 or 1) of the Boolean literal on top of the stack.
ConditionalExpr	Expr ::= $\text{Expr}_{\text{condition}}$ $\text{Expr}_{\text{trueValue}}$ $\text{Expr}_{\text{falseValue}}$	*Assignment 6*
IdentExpr	Expr ::= IDENT	Generate code to load the value of the given variable on top of the stack.
ImageAttributeExpr	Expr ::= IDENT SELECTOR	Generate code to load the appropriate attribute of the image indicated by the ident on top of the stack. You may use getter/setter methods of the PLPImage class or just access the fields directly.
IntLitExpr	Expr ::= INT_LIT	Generate code to leave the int value of the literal on top of the stack.
PreDefExpr	Expr ::= CONSTANT_LIT	Generate code to load the value on top of the stack. Z is defined in ImageConstants SCREEN_SIZE is defined in PLPImage.[1] *We will deal with x and y in assignment 6.*
SampleExpr	Expr ::= IDENT $\text{Expr}_{\text{xLoc}}$ $\text{Expr}_{\text{yLoc}}$ COLOR	Visit the expressions to generate code to leave their values on top of the stack. Use the getSample method to return the value of the sample and leave it on top of the stack.
Pixel	Pixel ::= $\text{Expr}_{\text{redExpr}}$ $\text{Expr}_{\text{greenExpr}}$ $\text{Expr}_{\text{blueExpr}}$	Visit the expressions and invoke the Pixel.makePixel method to pack it into an int.

[1] It would have made more sense to make this a pair instead of a single value. It gets the size, in pixels of the smallest of the width and height of the available screen (not including task bars, etc.)