| Abstract Syntax | Code generation |
|---|---|
| <Program> ::= IDENTIFIER <Block> | Create a class file for the program. The name is given by the identifier. **The visit method should return an array of bytes containing the class.** |
| <Block> ::= <DecOrCommand> * | |
| <Declaration> ::= <Type> IDENTIFIER | Add a static field with name given by the indicator and appropriate type to the classfile |
| | |
| <SimpleType> ::= int \| boolean \| string (use enum in Kind class) | These types are represented by java int, boolean, and java.lang.String respectively. |
| <CompoundType> ::=<SimpleType> <Type> | Not required in project 5. |
| <AssignExprCommand> ::= <LValue><Expression> | Generate code to evaluate the expression and store the results in the variable indicated by the LValue |
| <AssignPairListCommand> ::= <LValue><PairList> | Not required in project 5 |
| <PrintCommand> ::= <Expression> | Generate code to invoke System.out.print with the value of the Expression as a parameter. Note that this method is overloaded—you must invoke the correct one. |
| <PrintlnCommand> ::= <Expression> | Generate code to invoke System.out.println with the value of the Expression as a parameter. Note that this method is overloaded. You must invoke the correct one. |
| <DoCommand> ::= <Expression> <Block> | Generate as if for a Java while loop. |
| <DoEachCommand> ::= <LValue> $IDENTIFIER_0$ $IDENTIFIER_1$ <Block> | Not required in project 5 |
| <IfCommand> ::= <Expression> <Block> | Generate code as if for a Java if statement |
| <IfElseCommand> ::= <Expression><Block> <Block> | Generate code as if for a Java if-else statement |
| <SimpleLValue> ::= IDENTIFIER | When this appears on the lhs of an assignment, use as target of the assignment |
| <ExprLValue> ::= IDENTIFIER <Expression> | Not required in project 5 |
| <Pair> ::= <Expression> <Expression> | Not required in project 5 |
| <PairList> ::= <Pair> * | Not required in project 5 |

| | |
|---|---|
| <BinaryOpExpression>::= <Expression$_0$> op <Expression$_1$> (use enum in Kind class for Op) | Generate code to leave the value of the expression on top of the stack. See the table below for more details. |
| <LValueExpression> := <LValue> | Generate code to leave the value of the expression on top of the stack |
| <IntegerLiteralExpression> ::= INTEGER_LITERAL | Generate code to leave the value of the literal on top of the stack. |
| <BooleanLiteralExpression> ::= BOOLEAN_LITERAL | Generate code to leave the value of the literal on top of the stack. |
| <StringLiteralExpression> ::= STRING_LITERAL | Generate code to leave the value of the literal on top of the stack. |
| <Expression> | Generally, for all expressions, generate code to leave the value of the expression on top of the stack. |
| <UnaryOpExpression> ::= op <Expression> (use enum in Kind class for Op) | if op is -, negate the value on top of the stack, if op is !, logically negate the value on top of the stack. |

## BinaryOpExpressions

| operator\type of arguments | int | boolean | string | map |
|---|---|---|---|---|
| + | int addition | not defined | string concatenation | not required for project 5 |
| - | int subtraction | not defined | not defined | not required for project 5 |
| * | int multiplication | not defined | not defined | not required for project 5 |
| / | int division | not defined | not defined | not defined |
| & | not defined | conditional and | not defined | not defined |
| \| | not defined | conditional or | not defined | not defined |
| == | the two ints have the same value | the two booleans have the same value | the two strings contain the same characters | not required for project 5 |
| != | the two ints have different values | the two booleans have different values | the two strings have different values | not required for project 5 |
| < | as expected | as expected (false < true) | Not defined | not required for project 5 |
| > | as expected | as expected | Not defined | not required for project 5 |
| <= | as expected | as expected | Prefix or equals relation (invoke the startsWith method of the String class) | not required for project 5 |
| >= | as expected | as expected | Not defined | not required for |

If only one of the arguments is a String, the other one is coerced into a string, then String concatenation is performed.

The startsWith and  the various valueOf methods in the java.lang.String class may be useful.