jminusminus

Class JPreIncrementOp

- java.lang.Object
  - jminusminus.JAST
    - jminusminus.JStatement
      - jminusminus.JExpression
        - jminusminus.JUnaryExpression
          - jminusminus.JPreIncrementOp

-

```
 class JPreIncrementOp
extends JUnaryExpression
```

The AST node for a ++expr expression.

- **Field Summary**

- **Fields inherited from class jminusminus.JUnaryExpression**
  `arg`
- **Fields inherited from class jminusminus.JExpression**
  `isStatementExpression, type`
- **Fields inherited from class jminusminus.JAST**
  `compilationUnit, line`

- **Constructor Summary**

Constructors

Constructor and Description
`JPreIncrementOp(int line, JExpression arg)`

Construct an AST node for a ++expr given its line number, and the operand.

- **Method Summary**

Methods

Modifier and Type	Method and Description
`JExpression`	**`analyze`**`(Context context)` Analyze the operand as a lhs (since there is a side effect), check types and determine the type of the result.
`void`	**`codegen`**`(CLEmitter output)` In generating code for a pre-increment operation, we treat simple variable (JVariable) operands specially since the JVM has an increment instruction.

- **Methods inherited from class jminusminus.JUnaryExpression**

  `writeToStdOut`

- **Methods inherited from class jminusminus.JExpression**

  `codegen, isStatementExpression, type`

- **Methods inherited from class jminusminus.JAST**

  `line, partialCodegen`

- **Methods inherited from class java.lang.Object**

  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

- **Constructor Detail**

  - **JPreIncrementOp**

```
publicJPreIncrementOp(intline,
                      JExpressionarg)
```

Construct an AST node for a ++expr given its line number, and the operand.

Parameters:
`line` - line in which the expression occurs in the source file.
`arg` - the operand.

- **Method Detail**

  - **analyze**

```
publicJExpressionanalyze(Contextcontext)
```

Analyze the operand as a lhs (since there is a side effect), check types and determine the type of the result.

Specified by:
`analyze` in class `JExpression`

Parameters:
`context` - context in which names are resolved.

Returns:

the analyzed (and possibly rewritten) AST subtree.

- **codegen**

```
publicvoidcodegen(CLEmitteroutput)
```

In generating code for a pre-increment operation, we treat simple variable (JVariable) operands specially since the JVM has an increment instruction. Otherwise, we rely on the JLhs code generation support for generating the proper code. Notice that we distinguish between expressions that are statement expressions and those that are not; we insure the proper value (after the increment) is left atop the stack in the latter case.

Specified by:

`codegen` in class `JAST`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).