jminusminus

Class JPlusOp

- java.lang.Object
  - jminusminus.JAST
    - jminusminus.JStatement
      - jminusminus.JExpression
        - jminusminus.JBinaryExpression
          - jminusminus.JPlusOp


```
class JPlusOp
extends JBinaryExpression
```

The AST node for a plus (+) expression. In j--, as in Java, + is overloaded to denote addition for numbers and concatenation for Strings.

- **Field Summary**
- **Fields inherited from class jminusminus.JBinaryExpression**
  `lhs, operator, rhs`
- **Fields inherited from class jminusminus.JExpression**
  `isStatementExpression, type`
- **Fields inherited from class jminusminus.JAST**
  `compilationUnit, line`
- **Constructor Summary**

Constructors

Constructor and Description

JPlusOp(int line, JExpression lhs, JExpression rhs)
Construct an AST node for an addition expression given its line number, and the lhs and rhs operands.

- **Method Summary**

Modifier and Type	Method and Description
	Methods
JExpression	**analyze**(Context context) Analysis involves first analyzing the operands.
void	**codegen**(CLEmitter output) Any string concatenation has been rewritten as a JStringConcatenationOp (in analyze()), so code generation here involves simply generating code for loading the operands onto the stack and then generating the appropriate add instruction.

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

  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**

  - **JPlusOp**

```
publicJPlusOp(intline,
       JExpressionlhs,
       JExpressionrhs)
```

Construct an AST node for an addition expression given its line number, and the lhs and rhs operands.

Parameters:

`line` - line in which the addition expression occurs in the source file.
`lhs` - the lhs operand.
`rhs` - the rhs operand.

- **Method Detail**

  - **analyze**

```
publicJExpressionanalyze(Contextcontext)
```

Analysis involves first analyzing the operands. If this is a string concatenation, we rewrite the subtree to make that explicit (and analyze that). Otherwise we check the types of the addition operands and compute the result type.

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)
```

Any string concatenation has been rewritten as a JStringConcatenationOp (in analyze()), so code generation here involves simply generating code for loading the operands onto the stack and then generating the appropriate add instruction.
Specified by:
`codegen` in class `JAST`
Parameters:
`output` - the code emitter (basically an abstraction for producing the .class file).