jminusminus

Class JCastOp

- java.lang.Object
  - jminusminus.JAST
    - jminusminus.JStatement
      - jminusminus.JExpression
        - jminusminus.JCastOp

.


```
class JCastOp
extends JExpression
```

The AST for an cast expression, which has both a cast (a type) and the expression to be cast.

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

Constructors

Constructor and Description

`JCastOp(int line, Type cast, JExpression expr)`
Construct an AST node for a cast operation from its line number, cast, and expression.

- **Method Summary**

Methods

Modifier and Type	Method and Description
JExpression	**analyze**(Context context) Analyzing a cast expression means, resolving the type (to which we are casting), checking the legality of the cast, and computing a (possibly null) conversion for use in code generation.
void	**codegen**(CLEmitter output) Generating code for a cast expression involves generating code for the original expr, and then for any necessary conversion.
void	**writeToStdOut**(PrettyPrinter p) Write the information pertaining to this AST to STDOUT.

- **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**
  - **JCastOp**

```
publicJCastOp(intline,
       Typecast,
       JExpressionexpr)
```

Construct an AST node for a cast operation from its line number, cast, and expression.

Parameters:

`line` - the line in which the operation occurs in the source file.
`cast` - the type we're casting our expression as.
`expr` - the expression we're casting.

- **Method Detail**
  - **analyze**

```
publicJExpressionanalyze(Contextcontext)
```

Analyzing a cast expression means, resolving the type (to which we are casting), checking the legality of the cast, and computing a (possibly null) conversion for use in code generation.

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)
```

Generating code for a cast expression involves generating code for the original expr, and then for any necessary conversion.

Specified by:
`codegen` in class `JAST`

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

- **writeToStdOut**

```
publicvoidwriteToStdOut(PrettyPrinterp)
```

Description copied from class: JAST

Write the information pertaining to this AST to STDOUT.

Specified by:
`writeToStdOut` in class `JAST`

Parameters:
p - for pretty printing with indentation.