
compilationUnit
packageDeclaration importDeclaration taskDeclaration taskDeclaration taskDeclaration taskDeclaration <EOF>
package qualifiedName ; import qualifiedName ; task data taskBody task load taskBody task handle taskBody task write taskBody
hitty sys . console { localVarDeclarations functionDeclaration } { localTaskDeclaration localVarDeclarations functionDeclaration functionDeclaration functionDeclaration functionDeclaration } { localTaskDeclaration localVarDeclarations functionDeclaration functionDeclaration functionDeclaration functionDeclaration } { localTaskDeclaration localVarDeclarations functionDeclaration functionDeclaration functionDeclaration }
localVarDeclaration add functionParameters functionBody task data ; _init_ functionParameters functionBody _input_ functionParameters functionBody _output_ functionParameters functionBody _main_ functionParameters functionBody task data ; _init_ functionParameters functionBody _input_ functionParameters functionBody _output_ functionParameters functionBody _main_ functionParameters functionBody task data ; _output_ functionParameters functionBody _init_ functionParameters functionBody _main_ functionParameters functionBody
string localVarDeclarationSingle ; (string s) blockCode () blockCode () blockCode () blockCode () blockCode () blockCode () blockCode () blockCode () blockCode () blockCode () blockCode () blockCode
array [ ] { blockStatement } { } { } { blockStatement } { blockStatement blockStatement blockStatement blockStatement } { } { blockStatement } { blockStatement } { blockStatement blockStatement } { blockStatement } { } { blockStatement blockStatement blockStatement }
statement statement statement statement statement statement statement statement statement statement statement statement statement statement
invokeStatement assignStatement localVarDeclaration localVarDeclaration localVarDeclaration whileStatement assignStatement assignStatement localVarDeclaration forStatement assignStatement localVarDeclaration localVarDeclaration forStatement
expression functionCallParameters ; assign ; string localVarDeclarationSingle ; string localVarDeclarationSingle ; int localVarDeclarationSingle ; while parExpression statement assign ; assign ; int localVarDeclarationSingle ; for (forControl) statement assign ; int localVarDeclarationSingle ; qualifiedName localVarDeclarationSingle ; for (forControl) statement
localVar (expression) localVar = expression array [ ] = varInitializer line = varInitializer i = varInitializer (expression) blockCode localVar = expression localVar = expression i = varInitializer forInit ; expression ; forUpdate blockCode localVar = expression i = varInitializer File file = varInitializer forInit ; expression ; forUpdate blockCode
array . localVar localVar this . localVar localVar expression expression expression 1 { blockStatement blockStatement blockStatement blockStatement } this . localVar localVar this . localVar localVar expression expressionList expression operator2 expression expressionList { blockStatement } this . localVar localVar expression expression expressionList expression operator2 expression expressionList { blockStatement }
add s data handle . localVar localVar "" 0 statement statement statement statement data handle . localVar data write . localVar 0 expression localVar < localVar expression statement data handle . localVar 0 localVar expression localVar < localVar expression statement
data os . localVar assignStatement invokeStatement ifStatement assignStatement data data assign i data . localVar expression operator1 assignStatement data os . localVar assign i data . localVar expression operator1 invokeStatement
openfile functionCallParameters . localVar assign ; expression functionCallParameters ; if parExpression statement assign ; localVar = expression length localVar ++ assign ; openfile functionCallParameters localVar = expression length localVar ++ expression functionCallParameters ;
(expression) readAll functionCallParameters localVar = expression localVar (expression) (expression) breakStatement localVar = expression i 0 i localVar = expression (expression) i 0 i localVar (expression)
"data" () line localVar data . localVar localVar expression operator2 expression break ; i expression operator2 expression data [localVar] localVar "data1" file . localVar localVar
array [localVar] add line expression [expression] == localVar localVar + 1 i data [localVar] . localVar writeline data [localVar]
i localVar 0 EOF i i reverse functionCallParameters i
line ()