

Home

What is Janino?

Janino is a super-small, super-fast Java™ compiler. Not only can it [compile a set of source files to a set of class files](#) like the JAVAC tool, but also can it compile a Java™ [expression](#), [block](#), [class body](#) or [source file in memory](#), load the bytecode and execute it directly in the same JVM. Janino is not intended to be a development tool, but an embedded compiler for run-time compilation purposes, e.g. expression evaluators or "server pages" engines like JSP.

JANINO is integrated with [Apache Commons JCI \("Java Compiler Interface"\)](#) and [JBoss Rules / Drools](#).

JANINO can also be used for [static code analysis](#) or [code manipulation](#).

JANINO can be configured to use the [javax.tools.JavaCompiler](#) API (available since JDK 1.6), which removes the Java 5-related [limitations](#).

Properties

The major design goal was to keep the compiler small and simple, while providing an (almost) JAVAC 1.4 compatible compiler. I don't like the idea of carrying around huge libraries for simple applications.

The following elements of the Java programming language are implemented:

- package declaration, import declaration
- class declaration, interface declaration
- Inheritance (extends and implements)
- Static member type declaration
- Inner classes (member classes, local classes, anonymous classes)
- Class initializer, Instance initializer
- Field declaration, Method declaration
- Local variable declaration
- Class variable initializer, Instance variable initializer
- Block statement ({...})
- if ... else statement
- for statement
- while statement
- do ... while statement
- try ... catch ... finally statement
- throw statement
- return statement
- break statement
- continue statement
- switch statement
- synchronized statement
- All primitive types (boolean, char, byte, short, int, long, float, double)
- Assignment operator =
- Assignment operators +=, -=, *=, /=, &=, |=, ^=, %=, <<=, >>=, >>>=
- Conditional operators ?...:, &&, ||
- Boolean logical operators &, ^, |
- Integer bitwise operators &, ^, |

- Numeric operators `*`, `/`, `%`, `+`, `-`, `<<`, `>>`, `>>>`
- String concatenation operator `+`
- Operators `++` and `--`
- Type comparison operator `instanceof`
- Unary operators `+`, `-`, `~`, `!`
- Parenthesized expression
- Field access (like `System.out`)
- Superclass member access (`super.meth()`; `super.field = x;`)

Properties (cont'd.)

- `this` (reference to current instance)
- Alternate constructor invocation (like `this(a, b, c)`)
- Superclass constructor invocation (like `super(a, b, c)`)
- Method invocation (like `System.out.println("Hello")`)
- Class instance creation (like `new Foo()`)
- Primitive array creation (like `new int[10][5][1]`)
- Class or interface array creation (like `new Foo[10][5][1]`)
- Array access (like `args[0]`)
- Local variable access
- Integer, floating-point, boolean, character, string literal
- `null` literal
- Unary numeric conversion, binary numeric conversion, widening numeric conversion, narrowing numeric conversion
- Widening reference conversion, narrowing reference conversion
- Cast
- Assignment conversion
- String conversion (for string concatenation)
- Constant expression
- Block scope, method scope, class scope, global scope
- `throws` clause
- Array initializer (like `String[] a = { "x", "y", "z" }`)
- Primitive class literals, e.g. `"int.class"`
- Non-primitive class literals, e.g. `"String.class"`
- References between uncompiled compilation units
- Line number tables a la `"-g:lines"`
- Source file information a la `"-g:source"`
- Handling of `@deprecated` doc comment tag
- Accessibility checking (PUBLIC, PROTECTED, PRIVATE)
- Local variable information information for debugging (i.e. `"-g:vars"`)
- Checking of "definite assignment" (JLS2 16)
- Methods that compile to more than 32 KB
- J2SE 5.0: Static imports (single and on-demand; fields, types and methods)
- J2SE 5.0: Autoboxing and unboxing
- J2SE 5.0: `StringBuilder` class used (if available) for string concatenation
- J2SE 5.0: Covariant return types

Limitations

The following elements of the Java programming language are **not** implemented:

- `assert` (a rarely-used JDK 1.4 language feature)
- J2SE 5.0: Parametrized types (generics)

- J2SE 5.0: Enhanced FOR loop
- J2SE 5.0: Typesafe enums
- J2SE 5.0: Variable arguments
- J2SE 5.0: Annotations