

VoiceJava 定义

1. 定义package:

- 语法: `define package [_]+ [dot [_]+]*`
- 示例: `define package hello dot world, define package hello dot world dot star`
- 注意: 目前还不支持组合词情况, 即 `hello world` 还不能拼接为 `helloWorld`。

2. 导入module:

- 语法: `import static? _ [dot [_|star]]*`
- 示例: `import lyun, import cn dot edu dot lyun, import cn dot edu dot lyun dot star`

3. 定义interface:

- 语法: `define interface _`
- 示例: `define interface hello`
- 注意: 同 1.

4. 定义class:

- 语法: `define (Annotation | public | protected | private | abstract | static | final | strictfp) class _ [extends _]? [implements _]?`
- 示例: `define public class HelloWorld, define public class HelloWorld extends Greeting implements Bonjour`

5. 定义构造函数

- 语法: `define constructor`
- 备注: 还未实现。

6. 定义方法method:

- 语法: `define (Annotation | public | protected | private | abstract | static | final | synchronized | native | strictfp) function _ [throws Exception]?`
- 示例: `define public function sayHello, define public function sayHello throws Exception`

7. 定义箭头函数:

- 语法: `define arrow function`
- 示例: 备注: 还未实现。

8. 属性/变量定义:

- 语法: `define (Annotation | public | protected | private | static | final | transient | volatile) (_ list | _ [dot _]? [with _+]?) variable _`

- 示例: `define private int variable count, define int list variable list, define Pair with Integer String`

9. 定义类型:

- 语法: `type (_ list | _ [dot _]? [with _+]?) [extends _]?`
- 示例: `type int, type void`

10. 定义参数:

- 语法: `type (_ list | _ [dot _]? [with _+]?) variable _`
- 示例: `type int variable count, type NodeList with Statement variable nodelist`

11. 定义for循环:

- 语法: `define [enchanced]? for`
- 备注: `enchanced`还未实现。

12. 定义while循环:

- 语法: `define [do]? while`
- 备注: `do`还未实现

13. 定义if:

- 语法: `define if`

14. 定义switch:

- 语法: `define switch`

15. 定义try-catch:

- 语法: `define try catch`
- 备注: 还未实现。

16. 定义@Override

- 语法: `define at override`
- 备注: 还未实现。

17. 定义子表达式, 即括号。

- 语法: `subexpression`

18. break

19. continue

20. 构建新实例:

`new instance _ [dot _]*`

- 示例: `new instance Puppy, new instance HashMap dot Entry`

21. 抛出异常: `throw new _`

22. 6 种赋值形式:

- `let _ [dot _]? equal call _`
 - 示例: `let count equal call compute`
- `let _ [dot _]? equal _ [call _]+`
 - 示例: `let x dot a equal a call b call c`
- `let _ [dot _]? equal _ [dot _] _`
 - 示例: `let x equal b dot c dot d`
- `let _ [dot _]? equal [variable]? _`
 - 示例: `let x equal variable y`
- `let _ [dot _]? equal (int | byte | short | long | char | float | double | boolean | String) _`
 - 示例: `let x equal int 2`
- `let _ [dot _]? equal [expression]?`
 - 示例: `let x equal, let x equal expression`

23. 6 种返回形式:

- `return call _`
- `return _ [call _]+`
 - 示例: `return a call b`
- `return _ [dot _]*`
 - 示例: `return a, return a dot b`
- `return [variable]? _`
 - 示例: `return variable y, return y`
- `return (int | byte | short | long | char | float | double | boolean | String) _`
 - 示例: `return int 2`
- `return [expression]?`
 - 示例: `return, return expression`

24. 12 种表达式

- `expression? call _`
 - 示例: `expression call a`
- `expression? _ [call _]+`
 - 示例: `expression a call b`
- `expression? _ [dot _]?`
 - 示例: `expression a dot b`
- `expression? [variable]? _`
 - 示例: `expression variable y`
- `expression? (int | byte | short | long | char | float | double | boolean | String) _`
 - 示例: `expression int 2`
- `expression? _ plus plus`
- `expression? _ minus minus`

- expression? plus plus _
- expression? minus minus _
- expression? expression (op | compare) expression
- expression? _ (op | compare) expression
 - 示例: expression 3 times expression
- expression? _ (op | compare) _
 - 示例: expression 3 plus 4
- expression? variable _ index _
 - 示例: expression variable ns index i, 数组索引
- 备注:
 - op ::= plus | minus | times | divide | mod
 - compare ::= less than | less equal | greater than | greater equal | double equal | and | double and

25. 常用指令:

- move next
- jump out
- jump before _
 - 示例: jump before hello
- jump after _
 - 示例: jump after hello
- jump to line [_]? [start | end]?
 - 示例: jump to line 100 start, jump to line 100 end
- up [_ lines]?
 - 示例: up 5 lines
- down [_ lines]?
- left
- right
- select line
 - 备注: 选中当前行
- select body
 - 备注: 选中当前的区域
- select _
 - 备注: 选中名字
- select function [_]?
 - 示例: select function sayHello
 - 备注: 选中函数
- replace _ to _
- delete
- undo:撤销