

VoiceJava 定义

0. 定义 Name:

- 语法: `[_]+`
- 示例: `hello world,`

1. 定义 package:

- 语法: `define package Name [dot Name]*`
- 示例: `define package hello world dot nice code, define package hello dot world`

2. 导入 module:

- 语法: `import static? Name [dot [Name | star]]*`
- 示例: `import longyan university, import longyan dot university, import longyan dot star`

3. 定义 interface:

- 语法: `define (Annotation | public | protected | private | abstract | static | final | strictfp)? interface Name`
- 示例: `define interface hello world, define interface hello`

4. 定义 class:

- 语法: `define (Annotation | public | protected | private | abstract | static | final | strictfp)* class Name`
- 示例: `define public class hello world, define public class hello world`
- 定义一个 class, class name 可能带参数, 可能 extends something, 可能 implements something。因此需要很多的 `move next`。
- 可以考虑实现 `move to body` 这样的语法来快速跳转。
- 完整示例:
 - 期望结果 `public class HelloWorld<T, Filter> extends Greeting implements Invoker<T>`
 - 语句: `define public class hello world, type t, type filter, move next, type greeting, move next, type invoker with t`

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 Name`
- 示例: `define public function say hello, define public function say hello`

7. 属性/变量定义:

- 语法: `define (Annotation | public | protected | private | static | final | transient | volatile)* variable Name`
- 示例: `define private variable count, define variable list`

8. 定义类型:

- 语法: `type (list of Name | Name [dot Name]? [with Name? [and Name]*]?) [extends _]?`
- 示例: `type int, type void, type Node with Integer => Node<Integer>, type Node with => Node<>`
- 注意: 若Name为question mark, 则转换为?, 比如: `type Node with question mark => Node<?>`

9. 定义for循环:

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

10. 定义while循环:

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

11. 定义if:

- 语法: `define if`

12. 定义switch:

- 语法: `define switch`

13. 定义try-catch:

- 语法: `define try`

14. 定义catch clause

- 语法: `define catch`

15. 定义@Override

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

16. 定义子表达式, 即括号。

- 语法: `subexpression`

17. break

18. continue

19. 构建新实例:

- 语法: `new instance Name [dot Name]* [with Name [and Name]*]?`
- 示例: `new instance puppy`, `new instance hash map dot entry`, `new instance puppy with => new Puppy<>()`, `new instance puppy with question mark => new Puppy<?>()`, `new instance puppy with cat and dog => new Puppy<Cat, Dog>()`

20. 定义抛出异常

- 语法: `throw Name: throw IO exception => public void sayHello() throws IOException {}`

21. 抛出异常:

- 语法: `throw new Name: throw new IO exception => throw new IOException();`

22. let赋值:

- `let Name [dot Name]? equal [expression]?`
 - 示例: `let score equal`, `let score equal expression`

23. return返回:

- `return [expression]?`

24. 12 种表达式

- `expression? Name [dot Name]* [call Name]`
- `expression? Name [dot Name]*`
- `expression? dot Name` // 仅可作用于 `Name [dot Name]+`, 暂不支持接着 `call Name`
- `expression? call Name` // 后可 `move next`, 然后继续 `call Name`
- `expression? variable Name`
- `expression? (int | byte | short | long | char | float | double | boolean) Name`
- `expression? string Name`
 - 示例: `string hello world`, 支持多个单词。
- `expression? Name plus plus`
- `expression? Name minus minus`
- `expression? plus plus Name`
- `expression? minus minus Name`
- `expression? expression (op | compare) expression`
- `expression? variable Name index Name`
 - 示例: `expression variable name list index index`, 数组索引

- `conditional expression` // 三目运算 ?:
- `cast expression` // type cast, cast expression由类型和表达式组成
- `expression? Name instance of` // Instance Expression, followed by command `type xxx`, 比如: `obj instance of InvokerCluster => obj instance of, type invoker cluster`
- `expression? lambda expression` // lambda expression
- `expression? not expression` // unary not expression
- 备注:
 - `op ::= plus | minus | times | divide | mod`
 - `compare ::= less than | less equal | greater than | greater equal | double equal | and | double and | double or`
 - 备注: `and`还不支持

25. 常用指令:

- `move next`
- `move next body`
- `move next statement`
- `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