Simple Jay

--Manual--

Usage

Once you have Simple Jay you're able to run its command line, which can compile .sj files into .java files. Our command line takes the following options:

-h, --help	Show usage help message and exit.
-c, --compile	Compile a .sj script into a .java file to the same name and directory location.
-k, --chunk	Pipe in Simple Jay to STDIN and get back Java over STDOUT.
-m, --map	Write out the compiled java file into the specified path; use in conjunction with –compile.

Examples:

- **Usage of help command:**

  simple -h

  simple --help

- **Compile a .sj file into a .java file:**

  simple -c myfile.sj

  simple -c myfile.simple

  simple --compile myfile.sj

  simple --compile ../../myfile.sj

- **Pipe in SimpleJay to STDIN and get back Java over STDOUT:**

  simple -k "a = 9.099"

  simple --chunk "a = 12"

- **Write out your compiled java file into the specified path:**

  simple -c test.simple -m c:/

Language syntax

Our following examples are shown first in Simple Jay syntax on the left side and then the result in Java on the right side.

*Notice the examples have the correct indentation, but a .sj file doesn't require to be indented. In the compilation process the code get its proper indentation.

- **Comments:**

// This is a comment	// This is a comment

- **Variable declaration:**

  Variables names follow the same rule as in Java.

a = 9	int a = 9;
v_float = 2.0	float v_float = 2.0;
v_long = 1011231319813139999999	long v_long = 1011231319813139999999;
isTrue = true	boolean isTrue = true;

- **Conditions:**
  - **If conditions**

    if conditions are declared first by an expression enclosed or not in parenthesis followed by 'if:', a new line, statements and 'end'

a>0 if:
  a--
end
	if (a > 0){
	  a--;
	}

factor = 0.8
(a >= 100) if:
  a = a*factor
end
	float factor = 0.8
	if (a> 100) {
	  a = a*factor;
	}

  - **Or conditions**

    or conditions acts like else if and else conditions depending if it's preceded by a condition or not. If the 'or' is preceded by a condition it will act like an else if. On the other hand, if it's not preceded by a condition it'll be processed as an else.

factor = 0.8
(a >= 100) if:
  a = a*factor
end
or:
  a = 0
end
	float factor = 0.8;
	if (a> 100) {
	  a = a*factor;
	
	}
	else {
	  a = 0;
	
	}

burgers = 3
(burgers > 3) if:
  burgers--
end
(burgers == 0) or:
  burgers++
end
	int burgers = 3;
	burgers++;
	if(burgers>3) {
	  burgers = burgers + 1;
	
	}
	else if(burgers==0) {
	  burgers++;
	
	}

apples = 10
(apples > 10) if:
apples = apples – 1
end
or:
apples = 50
end
	int apples = 10;
	if(apples>10) {
	  apples = apples - 1;
	
	  }
	else {
	  apples = 50;
	
	}

- **Loops:**
  - **For loops**

    These loops are declared by typing ‘loop’ followed by a <<variable name>>, ‘to’, <<variable name>> or int number, a new line with statements and a new line with ‘end’. Also there are other optional specifications like ‘jump’ or ‘start’:

    ‘jump’ specifies the increment of the variable used.

    ‘start’ specifies where to start the used variable.

goToEat = 100
  loop pizza to goToEat:
end
	int goToEat = 100;
	  for(int burgers=0; burgers<goToEat;
	  ++burgers) {
	
	}

loop a to 20 jump 2:
end
	for(int a=0; a<20; a+=2) {
	
	}

loop a to 20 jump 2 start 5:
end
	for(int a=5; a<20; a+=2) {
	
	}

loop a to 20 start 5:
end
	for(int a=5; a<20; ++a) {
	
	}

- **While loops**

  To declare a while loop the user must type an expression enclosed by parenthesis or not, followed by ‘loop:’ a new line, statements and in a line ‘end’.

a = 10
(a > 0)loop:
end
	int a = 10;
	while(a>0) {
	
	}

- **Functions:**

  To declare functions just type a valid function name followed by its parameters (must specify its types) enclosed by parenthesis, ‘:’, new line, statements and ‘end’.

  Simple Jay auto assigns the function type based on what the function returns.

fibo (int nums):
x = 1
y = 1
(nums == 0) if:
-> 0
end
(nums > 2) or:
c = 0
loop a to nums start 2:
c = x+y
x = y
y = c
end
-> c
end
or:
-> 1
end
end
	int fibo (int nums){
	int x = 1;
	int y = 1;
	if(nums==0) {
	return 0;
	
	}
	else if(nums>2) {
	int c = 0;
	for(int a=2; a<nums; ++a) {
	c = x+y;
	x = y;
	y = c;
	
	}
	return c;
	
	}
	else {
	return 1;
	
	}
	
	}

myFunction():
a = 1000.00
end
	void myFunction (){
	float a = 1000.00;
	
	}

myFunction(float a, float b): -> a+b	float myFunction (float a, float b){ return a+b;

end	}

myFunction(int a):
(a > 0) if:
-> true
end
or:
-> false
end
end
	boolean myFunction (int a){
	if(a>0) {
	return true;
	}
	else {
	return false;
	}
	}