MUA语言解释器2.0

2018.12

3160103113 曾显珺

MUA语言解释器基本操作：

1.0：

* //: 注释

实现方法：读入所有字符串prase剪去“//”及后面部分

String[] word = line.split("\\s++"); //deal with "//"

**for**(i = 0; i < word.length; i ++) {

**if**(word[i].substring(0, 1) == "\\") {

length = i;

}

length = i;

}

* make <word> <value>: 将value值绑在word上

条件约束：word必须以字母开头，字面量必须以“开头

实现方法：

**public** **static** **int** make(List<String> cmd, **int** index) {

**int** i;

//check if the first argument is a word

**if**(cmd.get(index+1).charAt(0) == '\"') {

String name = cmd.get(index+1).substring(1, cmd.get(index+1).length());

//check whether there exists such varibles

**for**(Map.Entry<String, String> entry: mua.*v*.entrySet()) {

String tmpname = entry.getKey();

**if**(tmpname.equals(name)) {

String tmpvalue = cmd.get(index + 2);

mua.*v*.put(name, tmpvalue);

cmd.remove(index);

cmd.remove(index);

cmd.remove(index);

function.*checkFunc*(name);

**return** 1;

}**else** **continue**;

}

String tmpname = name;

String tmpvalue = cmd.get(index+2);

mua.*v*.put(name, tmpvalue);

//update the cmd

cmd.remove(index);

cmd.remove(index);

cmd.remove(index);

function.*checkFunc*(name);

**return** 1;

}**else** {

Parse.*clear*();

**return** 0;

}

}

* thing <word>: 返回word 绑定的值（float）
* :word : 同thing <word>

方法实现：通过绑定hashtable<string, float> v容器实现：

**public** **static** Hashtable<String, String> *v* = **new** Hashtable(); //mua.java

**public** **static** **int** thing(List<String> cmd, **int** index) {

**int** i,j,k;

String name = cmd.get(index+1);

name = name.substring(1, name.length());

**for**(Map.Entry<String, String> entry: mua.*v*.entrySet()) {

//DataStructure tmp = (DataStructure) mua.v.get(i);

String tmpname = entry.getKey();

**if**(tmpname.equals(name)) {

String tmpvalue = entry.getValue();

cmd.set(index, tmpvalue);

cmd.remove(index+1);

**return** 1;//success

}

}

Parse.*clear*();

**return** 0;//fail

}

* erase <word>: 在hashtable中清除word所绑定的值 //在execute.java中

**public** **static** **int** erase(List<String> cmd, **int** index) {

**int** i;

String name = cmd.get(index+1).substring(1, cmd.get(index+1).length());

**for**(Map.Entry<String, String> entry: mua.*v*.entrySet()) {

String tmpname = entry.getKey();

**if**(tmpname.equals(name)) {

mua.*v*.remove(tmpname);

cmd.remove(index);

cmd.remove(index);

**return** 1;//success

}

}

cmd.remove(index);

cmd.remove(index);

**return** 0;//failed

}

* isname <word>: 返回Boolean，判断word是否为一个名字

**public** **static** **int** isname(List<String> cmd, **int** index) {

**int** i;

String name = cmd.get(index+1).substring(1, cmd.get(index+1).length());

**for**(Map.Entry<String, String> entry: mua.*v*.entrySet()) {

String tmpname = entry.getKey();

**if**(tmpname.equals(name)) {

cmd.set(index, "true");

cmd.remove(index + 1);

**return** 1;//

}

}

cmd.set(index, "false");

cmd.remove(index + 1);

**return** 1;//

}

* print <value>: 输出value
* read: 返回读取的数字或字

**public** **static** **int** read(List<String> cmd, **int** index) {

String tmp;

Scanner input=**new** Scanner(System.***in***);

tmp = input.next();

cmd.remove(index);

cmd.add(index, tmp);

**return** 1;

}

* repeat <number> <list>`：运行list中的代码number次

读取次数后加倍命令行数

**public** **static** **int** repeat(List<String> cmd, **int** index) {

String repeat = cmd.get(index + 2);

String[] word = repeat.split("\\s++");

Integer count = Integer.*valueOf*(cmd.get(index + 1));

List<String> repeatCmd = **new** ArrayList<String>();

**for**(**int** i = 1; i < word.length-1; i ++) {

repeatCmd.add(word[i]);

}

cmd.remove(index);

**for**(**int** i = 0; i < count; i ++) {

**for**(**int** j = 0; j < repeatCmd.size(); j ++) {

cmd.add(index+j+i\*repeatCmd.size(), repeatCmd.get(j));

}

}

**return** 1;

}

* readlist 返回一个从标准输入读取的一行，构成一个表，行中每个以空格分隔的部分是list的一个元

**public** **static** **int** readlist(List<String> cmd, **int** index) {

List<String> tmpcmd= **new** ArrayList<String>();//store the cmd

**for**(**int** i = 0; i < cmd.size(); i ++) {

String tmp = cmd.get(i);

tmpcmd.add(tmp);

}

tmpcmd.remove(index);

cmd.clear();

Parse.*Read*();

String List;

List = "[";

**for**(**int** i = 0; i < cmd.size(); i++) {

List = List + " ";

List = List + cmd.get(i);

}

List = List + " ]";

cmd.clear();

**for**(**int** i = 0; i < tmpcmd.size(); i ++) {

cmd.add( tmpcmd.get(i));

}

cmd.add(index,List);

**return** 1;

}

* make <word> [<list1> <list2>]word为函数名，list1为参数表，ist2为操作表‘

make函数中function.*checkFunc*(name);用于检验make后的word是否为函数名字

**public** **static** **void** checkFunc(String word) {//after make operation, to check if this word is a function

String value = mua.*v*.get(word);

**if**(value.length() < 11) **return**;

**if**(value.charAt(0) != '[')**return**;

**if**(value.charAt(1) != ' ' || value.charAt(2) != '[') **return**;

List<String> list = *parseList*(value); //if is a function, we should add it to reserved word list, regard it as a operation

**if**(list.size() == 2) { //has two list, it is a list

//logger.warning("It is a function");

**int** argcount = 0;

String arglist = list.get(0);

String []arg = arglist.split("\\s++");

argcount = arg.length-2;

//we have to store the number of arg this function need

mua.*functions*.put(word, Integer.*valueOf*(argcount));

}

**return**;

}

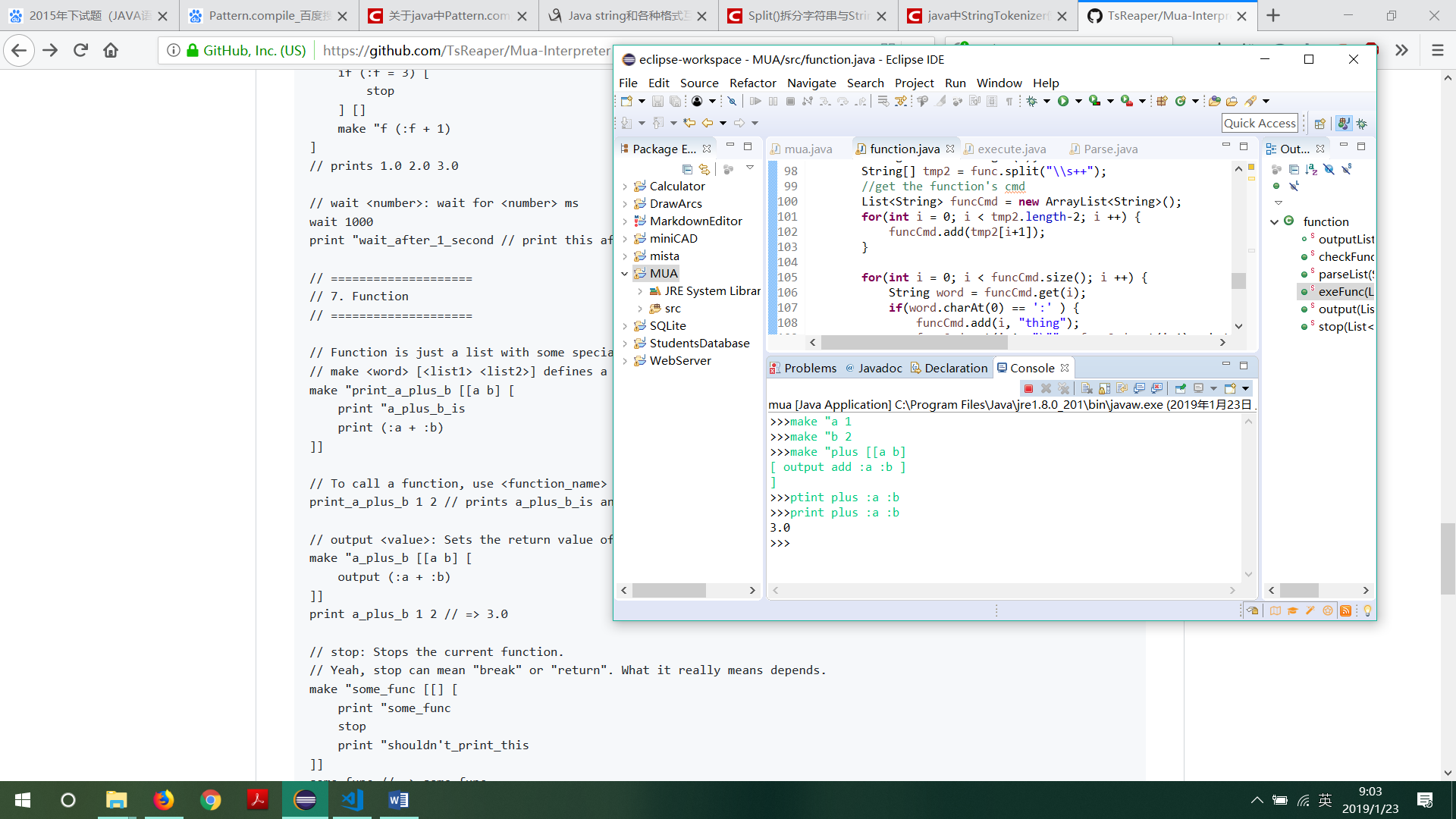
Phrase.java中Interpret函数读入参数并对应functions中的操作名

在function.java的exeFun函数中将参数转换为值再运算

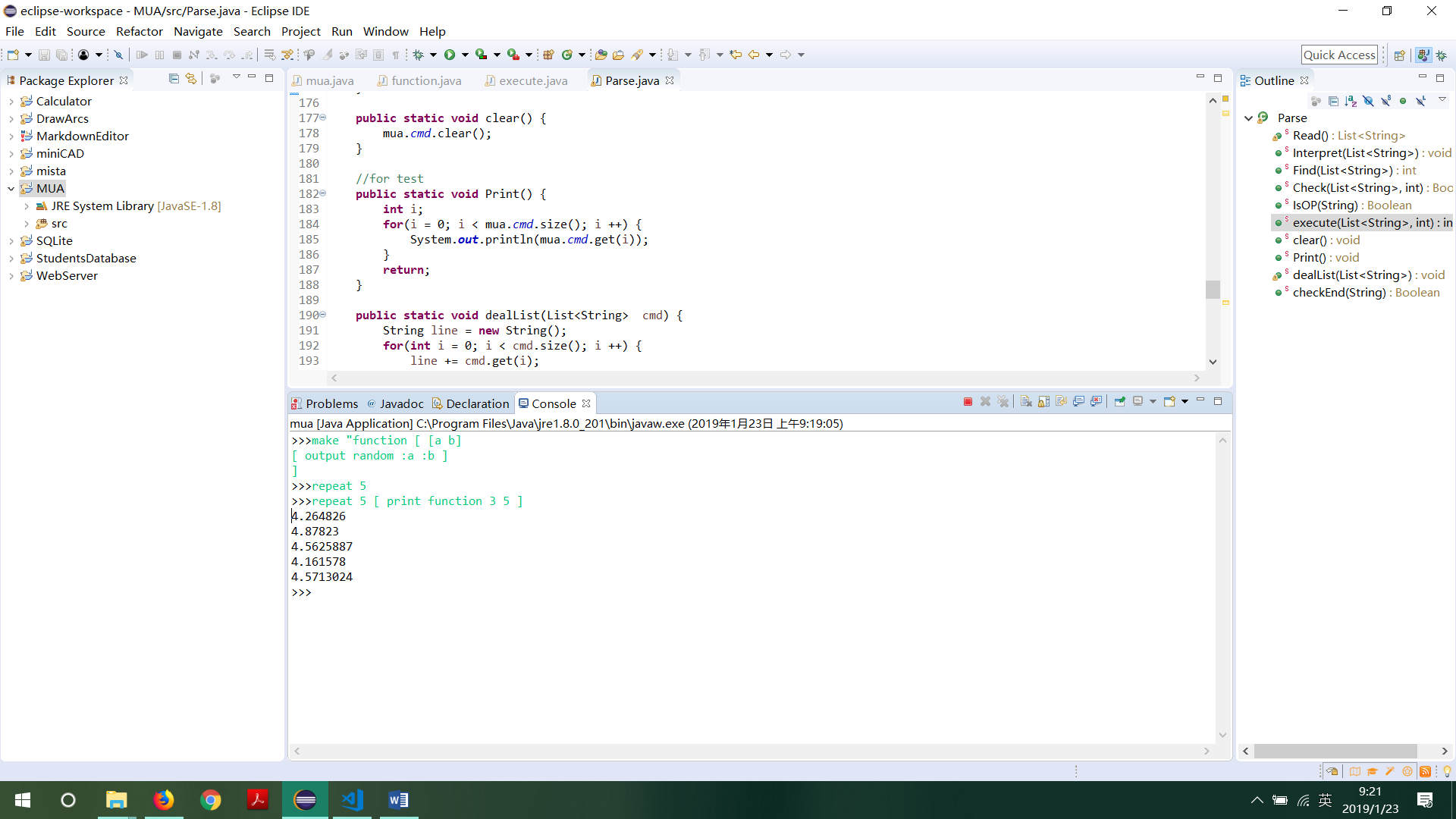
Output 语句执行function.output函数

运行结果：

简单的函数



random运算：



Sqrt运算：

