

# MUA 说明文档

---

姓名：秦昇 学号：3120000060 课程：程序设计方法学

## 系统开发说明：

---

1. 基于 JAVA 语言，以 MadeUpProgrammingLanguage.md 文档内容为语法规则编写的 MUA 语言解释器。
  2. 支持文档中所提的所有函数，以及自定义函数操作等。
- 

## 具体实现步骤

---

### 1. 类的定义

定义一个主类，在 main 函数中实现对程序的读入，当输入“exit”的时候退出，否则一直等待用户输入。

定义 Value 类来存储数据，这里包括 number, word, list, bool 等基本数据类型和函数名，并在 type 中加以区分。

定义一个 Line 的类来处理程序的输入，包括每读一行后过滤掉行首的空格等无关字符，以及过滤掉“//”注释之后的内容。

定义 OpHandler 类来对所有 MUA 本身的函数和用户自定义的函数进行操作。

定义 ErrorHandle 类来处理所有的错误情况，并终止程序。

### 2. 基本思想

程序设计的基本思想是，将所有用户输入的字符串视作一个 Value，其中包括了操作符和函数，每次调用 Value 类中的 getValue 函数来获取对应 Value 的值，如果是操作符，则调用对应的函数，Value 的值就是该函数的返回值。这样设计，便于实现函数的嵌套。

需要说明的是，由于 list 中可以保存代码，所以单独写了 getListValue 这样一个函数，在读到操作符的时候保存操作符。

所以整个程序可以解释为，若干语句的连接，每次执行一个语句，在执行一个语句的过程中，遇到嵌套的语句则执行嵌套的语句，但是层次仍然相同。

当执行到用户自定义的函数或者 repeat 等可以利用 stop 函数来停止的语句的时候，构造一个新的 Line 对象，来保存要执行的语句，遇到 stop 则直接跳出，可

以看出这些语句，相比于调用他们的语句来说，层次低了以及，相当于一个子程序，这样一层一层地去执行，就可以支持函数的嵌套和递归调用。

## 重点难点

### 1. 函数的实现

参考如下代码，函数的实现分两步

第一步是用实参替换所有的形参；

第二步则是将函数代码放到一个新的 Line 里面去执行。

```
public void doFunction(String op, Line line, Value returnValue){
    if(!space.containsKey(op)){
        ErrorHandle EH = new ErrorHandle();
        EH.Handle(EH.FUNCTIONNAMEERROR);
    }
    Value oparg = space.get(op);
    if(oparg.type == oparg.LIST && oparg.list.size() == 2 &&
        oparg.list.elementAt(0).type == oparg.LIST &&
        oparg.list.elementAt(1).type == oparg.LIST){
        Value argument = space.get(op).list.elementAt(0);
        Value operation = space.get(op).list.elementAt(1);
        Map<String, Value> args = new HashMap<String, Value>();

        for(int i = 0; i < argument.list.size(); i++){
            Value tmparg = new Value();
            tmparg.getValue(0, line);
            args.put(argument.list.elementAt(i).word, new Value(tmparg));
        }

        Line addline = new Line();
        addline.nowline = replaceListToLine("", args, operation);

        Value re = new Value();
        while(!addline.nowline.isEmpty()){
            if(re.getValue(0, addline)){
                returnValue.type = re.type;
                switch (re.type){
                    case 1:
                        returnValue.number = re.number;
                        break;
                    case 2:
                        returnValue.word = re.word;
                        break;
                }
            }
        }
    }
}
```

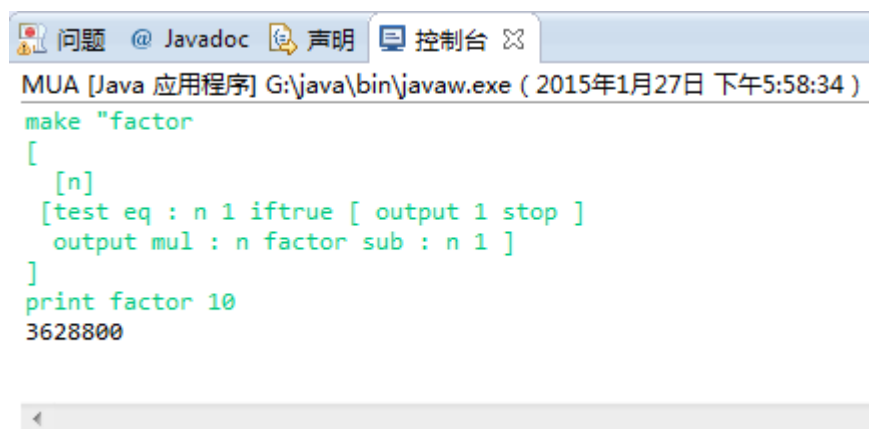
```

        case 3:
            returnValue.list = re.list;
            break;
        case 4:
            returnValue.bool = re.bool;
            break;
    }
}
lineFilter(addline);
}
}
else{
    returnValue.type = oparg.type;
    returnValue.Copy(oparg);
}
}
}

```

## 成果展示

1. 运行 factor 函数求阶乘

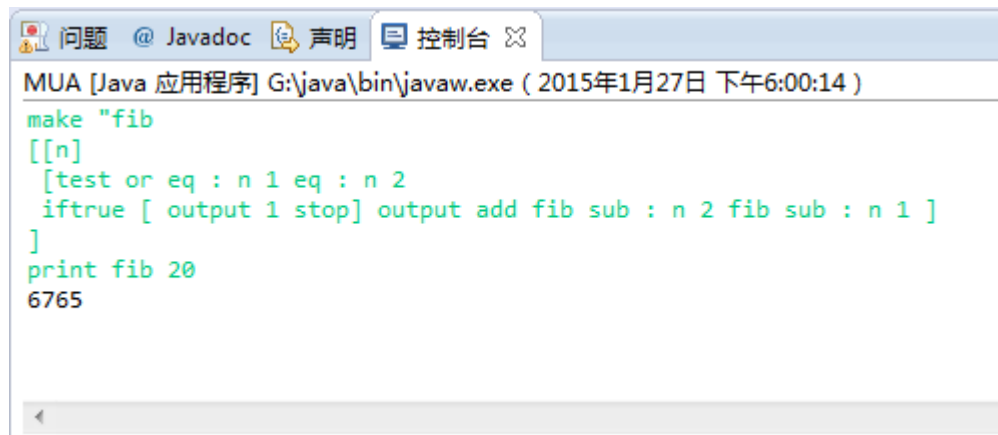


```

MUA [Java 应用程序] G:\java\bin\javaw.exe ( 2015年1月27日 下午5:58:34 )
make "factor"
[
  [n]
  [test eq : n 1 iftrue [ output 1 stop ]
    output mul : n factor sub : n 1 ]
]
print factor 10
3628800

```

2. 运行 fib 函数求斐波那契数列通项



```

MUA [Java 应用程序] G:\java\bin\javaw.exe ( 2015年1月27日 下午6:00:14 )
make "fib"
[[n]
  [test or eq : n 1 eq : n 2
    iftrue [ output 1 stop] output add fib sub : n 2 fib sub : n 1 ]
]
print fib 20
6765

```

### 3. 不退出直接运行 sort 函数

```
make "fib
[[n]
 [test or eq : n 1 eq : n 2
  iftrue [ output 1 stop] output add fib sub : n 2 fib sub : n 1 ]
]
print fib 20
6765
make "sort
[[a]
 [
  test isempty : a
  iftrue [output : a stop]
  make "m min : a
  output list join [] : m sort delete : a : m
]]
make "min
[[a]
 [
  test isempty butfirst : a
  iftrue [output first : a stop]
  make "lmin min butfirst : a
  test lt first : a : lmin
  iftrue [output first : a stop]
  output : lmin
]
]

make "delete
[[a ele]
 [
  test isempty : a
  iftrue [output : a stop]
  test eq first : a : ele
  iftrue [output butfirst : a stop]
  output list join [] first : a delete butfirst : a : ele
]]
print sort [2 12 3 8 34 9 10]
[2 3 8 9 10 12 34]
```

## 编程体会

其实整个程序设计不算难，主要是设计好框架之后一点一点地把功能、函数添加进去，但是细节的地方有很多要注意的，整体来说还是花了不少时间。感悟比较深的有以下几点

### 1. 体会函数式编程特点

其实 MUA 语言的设计思想和函数式编程基本一样，相比于 C、JAVA 等语言，函数式编程语言更加贴近于我们的思维，按照我的理解可以表示为如下形式：

确定我要干什么 -> 调用哪个函数 -> 需要什么参数 -> 得到我的结果

所以这也决定了我的程序设计思路，就是将一切语句拆分成基本语句的嵌套和连接，一层层地按照这样的思想去处理嵌入到参数里面的语句。

## 2. 在设计之前预先思考

虽然设计这个大程花了不少时间，但是庆幸的是我在预先设计的时候已经构思好了大体的处理框架和思路，让我几乎没有重写或者对程序有大的改动。而且可以节省很多代码量。

## 3. 还需改进的地方

不过这次作业我收获的更多是不足和反思，首先是对程序本身，我认为还需改进的地方是

1. 对错误的处理不够完善，虽然在一开始已经想好了专门写一个类来处理各类问题，但是却没有想好在哪些时候需要进行语法的检查等，所以到了后来变成了写完整个程序以后再回过头来往可能出错的地方去添加错误处理。
2. 处理操作符的时候不够简洁，这一点要和第一次作业 BNF for MUA 结合起来看，BNF 中把语法结构相同的函数可以归为一类，这样在我的程序中，switch 操作符的时候，所以参数类型相同的可以调用同一个函数，这样可以大大减少 OpHandler 中函数的数量和 switch 语句中 break 的数量。
3. 细节处理不够，这次大部分时间花在了改和调 bug 上，感觉写大程序很容易犯小错误，用一些样例去调试找出问题后，不断地随过程调试中基本又将自己写过的代码读了好几遍。
4. 整体感觉通过这门课学到了很多的东西，特别是把自己知道的语言的数量翻了很多倍，也第一次了解了函数式编程语言，在写论文的过程中也对编程语言未来的发展有了自己的看法。

下一步计划是在寒假完善一下 MUA 编译器，写一个 UI，扩充一下语法，让 MUA 可以完成更复杂的程序，整体就写这么多吧，考试周还是很忙的，祝翁恺大大新年快乐~!