## 概要设计

* 预期主要实现功能：
  + 实现基本的带数字的表达式计算；
  + 可以将一个值赋给一个变量，并且实现添加变量的计算；
  + 可以查询以前的有过赋值的变量的值，并且实现在此基础上的修改或删除。
* 具体细化工作：
  + 需要实现的基础计算包含，加法("+")、减法("-")、乘法("\*")、除法("/")、乘方("^")以及带括号的计算；
  + 尝试添加变量操作，但是添加变量时通过设置等式右边为一个表达式，左边为变量名来实现；
  + 对所有的手工添加的变量，可以通过输入命令来进行查询，修改，删除操作；
  + 整个最终程序不再考虑图形化界面，主要以命令行界面为主；
  + 尽量考虑足够多的运行时候错误，保证程序最终能够正常运行。

## 详细设计

* 表达式与变量的定义
  + 字母<Letter>的定义：
    - 一位数字、大小写字母或下划线的字符
  + 实数<Real>的定义：
    - 即为程序设计语言中的实数定义
  + 变量<Var>的定义：
    - <Var> ::= <Var><Letter> | <Letter>
  + 运算量<Num>的定义：
    - <Num> ::= <Var> | <Real>
  + 单目运算符<Opr1>的定义：
    - <Opr1> ::= '()'
  + 双目运算符<Oper2>的定义：
    - <Opr2> ::= '+' | '-' | '\*' | '/' | '^'
  + 运算式<Run>的定义：
    - <Run> ::= <Num> | <Opr1><Num> | <Num><Opr2><Num>
  + 运算表达式<Exp>的定义：
    - <Exp> ::= <Run> | <Opr1><Run> | <Num><Opr2><Run> | <Run><Opr2><Run>
* 变量和表达式的输入方法
  + 借鉴Python中直接输入表达式就运行计算的优点，所有直接输入的表达式将直接运行计算；
  + 借鉴Python中直接新建变量的方法，不需要前置说明变量的类型；
  + 借鉴Pascal的赋值运算符:=，对变量的赋值语句格式写为"<Name>:=<Exp>"，通过赋值运算符来实现对变量的赋值；
  + 借鉴常用计算器中的Ans默认变量，即默认存储上一次计算的结果；
  + 如果在表达式<Exp>中出现了没有定义的变量，则报错；
  + 不允许循环定义，类似"a := a + 1 "的变量定义表达式是非法的。
* 变量的查询方法
  + 借鉴Linux的ls命令，该命令的作用是将所有的变量展示出来
  + 借鉴Linux的cat命令，该命令的作用是显示这个变量的值
  + 借鉴Linux的cp命令，该命令的作用是把一个变量的值赋给另一个变量
  + 借鉴Linux的rm命令，该命令的作用是删除一个变量
  + 借鉴Linux的mv命令，该命令的作用是修改一个变量的变量名
* 主控制台的控制命令
  + 借鉴exit命令，该命令的作用是终止程序
  + 借鉴MSDOS的cls命令，该命令的作用是清空所有的已经设置的变量

## 类与函数的设计

* 异常类
  + 包含语法异常、运行异常和数学异常
  + 语法异常，即为语法错误
  + 运行异常，诸如表达式缺少运算量，缺少后括号等异常
  + 数学异常，诸如大小越界，除以零，零的零次方等等数学错误
* 表达式类
  + 包含以下部分：运算表达式，运算中的自定义变量集合，运算主过程
  + 自定义变量集合用哈希表维护
  + 运算表达式用一串字符串维护，每当运行新的运算时，就更新该字符串，并且更新相关的数据
  + 运算主过程由两个堆栈实现：运算符栈sOpr和数值栈sNum。每进行一次运算，则sOpr弹出一个运算符，sNum弹出两个数据，将结果压入sNum中
* 命令类
  + 大体可以分成三类：针对自定义变量的命令；针对全局的命令，以及表达式命令（将变量添加与表达式计算视为一个）
  + 自定义变量命令有5种，ls, cat, cp, mv, rm
  + 全局命令有两个，exit退出和cls清零
  + 表达式命令需要判别是否为赋值命令还是计算命令

## 类的详细设计

* 表达式类
* 命令类
  + 表达式命令
    - 运算表达式
      * 语法：<Exp>
      * 返回一个布尔值
      * 成功计算返回TRUE，否则返回FALSE
      * 计算表达式并且返回一个值，如果计算失败则返回nan
      * 如果出现表达式缺少运算符，抛出异常MissingOperator
      * 如果出现计除以零错误，抛出异常MathError
    - 赋值表达式
      * 语法：<Name>:=<Exp>
      * 返回一个布尔值
      * 成功创建变量返回TRUE，否则返回FALSE
      * 如果<Name>的命名非法（首字母为数字或出现了非法字符），抛出异常UnqualifiedName
      * 如果出现了多个赋值运算符，抛出异常UnqualifiedSymbol
      * 如果出现了<Exp>的异常，则按照<Exp>的处理方式处理
    - 可能出现的错误
      * 不能识别的命令，抛出异常CommandError
      * 语法错误，抛出异常SyntaxError
  + 自定义变量的操作命令
    - ls命令
      * 语法：ls
      * 返回一个布尔值
      * 显示所有的变量（变量数大于1）
      * 如果没有变量，抛出异常EmptyVariableSet
    - cat命令
      * 语法：cat <Name>
      * 返回一个布尔值
      * 的确存在这个值，就返回TRUE，否则返回FALSE
      * 显示<Name>所对应的变量的值
      * 如果没有这个值，抛出异常NoSuchVariable
    - cp命令
      * 语法：cp <Name1> <Name2>
      * 返回一个布尔值
      * 成功复制这个值，就返回TRUE，否则返回FALSE
      * 将<Name1>的值复制到<Name2>中
      * 如果<Name1>不存在，抛出异常NoSuchVariable
      * 如果<Name2>不存在，则新建一个<Name2>，然后接着赋值
    - rm命令
      * 语法：rm <Name>
      * 返回一个布尔值
      * 成功删除这个值，就返回TRUE，否则返回FALSE
      * 删除<Name>所对应的变量
      * 如果没有这个值，抛出异常NoSuchVariable
    - mv命令
      * 语法：mv <Name1> <Name2>
      * 返回一个布尔值
      * 成功重命名这个值，就返回TRUE，否则返回FALSE
      * 将<Name1>的变量的名称修改为<Name2>
      * 如果<Name1>不存在，抛出异常NoSuchVariable
    - 可能出现的错误
      * 不能识别的命令，抛出异常CommandError
      * 语法错误，抛出异常SyntaxError
  + 全局性命令
    - exit命令
      * 语法：exit
      * 返回一个布尔值TRUE
      * 用于在主界面表示程序终止
    - cls命令
      * 语法：cls
      * 返回一个布尔值
      * 成功清空返回TRUE，失败则返回FALSE
      * 用于清空所有的自定义变量
    - 可能出现的错误
      * 不能识别的命令，抛出异常CommandError
      * 语法错误，抛出异常SyntaxError
* 异常类
  + 缺少运算符MissingOperator
  + 数学运算错误MathError
  + 命名非法UnqualifiedName
  + 多个赋值运算符UnqualifiedSymbol
  + 没有任何的自定义变量EmptyVariableSet
  + 缺少特定的自定义变量NoSuchVariable
  + 不能识别的命令CommandError