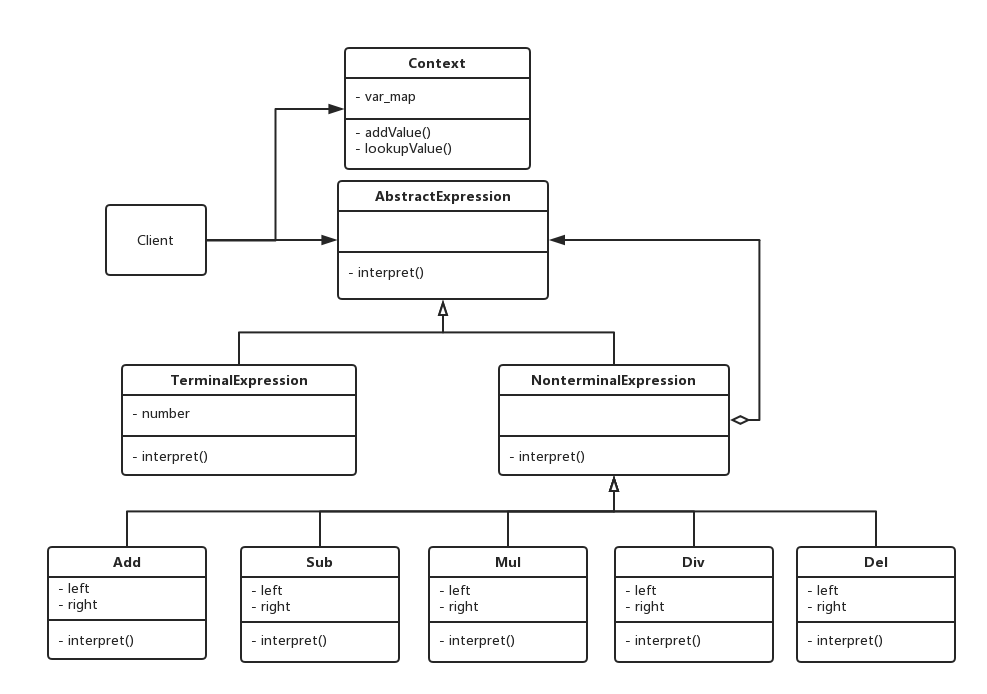
系统要求

实现一个针对整数的加减乘除以及取模的解释器

UML类图



类图使用PrcessOn（https://www.processon.com）在线生成工具完成。

程序采用解释器模式进行设计。

解释器程序主要包含四个类，Context、AbstractExpression、TerminalExpression、NonterminalExpression。各类之间关系见上图，上图中包含相应类的接口及其依赖关系。

程序实验过程中对上述设计中的AbstractExpression、TerminalExpression、NonterminalExpression进行了简化，使用Expression基类以及其六个子类，包括Variable、Add、Sub、Mul、Div、Del，由于后五个类极其相似，下面只介绍Add类。

Context类用于对输入表达式字符进行处理功能的抽象，包含两个方法，一个属性：

一个属性为：

1. var\_map：用于建立变量与相应数值之间的映射；

Expression类用于对基础表达式进行抽象，包含一个方法：

一个方法为：

1. interpret(): 主要用于子类相关功能的重写；

Variable类用于对终止表达式的抽象，包含一个方法，一个属性：

一个属性为：

1. var：记录变量名；

一个方法为：

1. interpret(): 用于依据变量得到相应数值；

Add类用于对非终止表达式，表达式相加功能的抽象，包含一个方法，两个属性：

两个属性为：

1. left：记录加号左边表达式；
2. right: 记录加号右边表达式；

一个方法为：

（1）interpret(): 将左右两边表达式的结果相加；类似计算功能在该函数重写上实现左右表达式相关的运算操作；

系统实现

该版本的系统使用C++实现。工程包五个文件（main.cpp context.cpp context.h expression.cpp expression.h）。

其中：

1. main.cpp: 包含简单用户引导功能实现，用户依据提示输入各个变量的实际数值，最终得到提示运算表达式的结果；
2. context.cpp & context.h: 包含环境角色类定义的实现，主要实现建立变量与相应数值的映射；
3. expression.cpp & expression.h：包含表达式类定义的实现，功能包括对于终结表达式，根据变量名得到相应的变量数值，对于非终结表达式，计算左右表达式之间相应的运算结果；

整数运算解释器系统实现过程：在用户进入交互界面后，通过输入各个变量的相应数值，最终得到提示表达式的运算结果。

由于时间有限，需要注意的实现细节如下：

1. 功能更加强大的运算解释器应支持各种运算表达式的计算并提供对输入表达式的分析，考虑到本次实验的目的在于设计模式的了解而不在于输入语言的处理，因此本版本只提供固定运算表达式的计算；
2. 由于本版本采用C++实现，对于类的重写以及类之间的递归调用可能不如使用Java实现便捷，对于解释器中表达式的递归调用，本版本中使用了显式调用，这种实现方式在更换表达式时灵活性会收到约束，期待能够在未来的版本中加以改进。

系统运行结果

