

C++大作业——黎容熙

C++大作业——黎容熙

T1

设计内容
设计思路
程序框图
运行效果
总结分析

T2

设计内容
设计思路
程序框图
运行效果
总结分析

T3

设计内容
设计思路
程序框图
运行效果
总结分析

T5

设计内容
设计思路
程序框图
运行效果
总结分析

T1

设计内容

程序运行时，自动读入与程序同目录下，存有汇率的 `rate.csv` 文件，若目录下没有该文件，程序会报错。当需要新增、修改、删除货币时，只需要修改csv文件即可，无须更改程序。

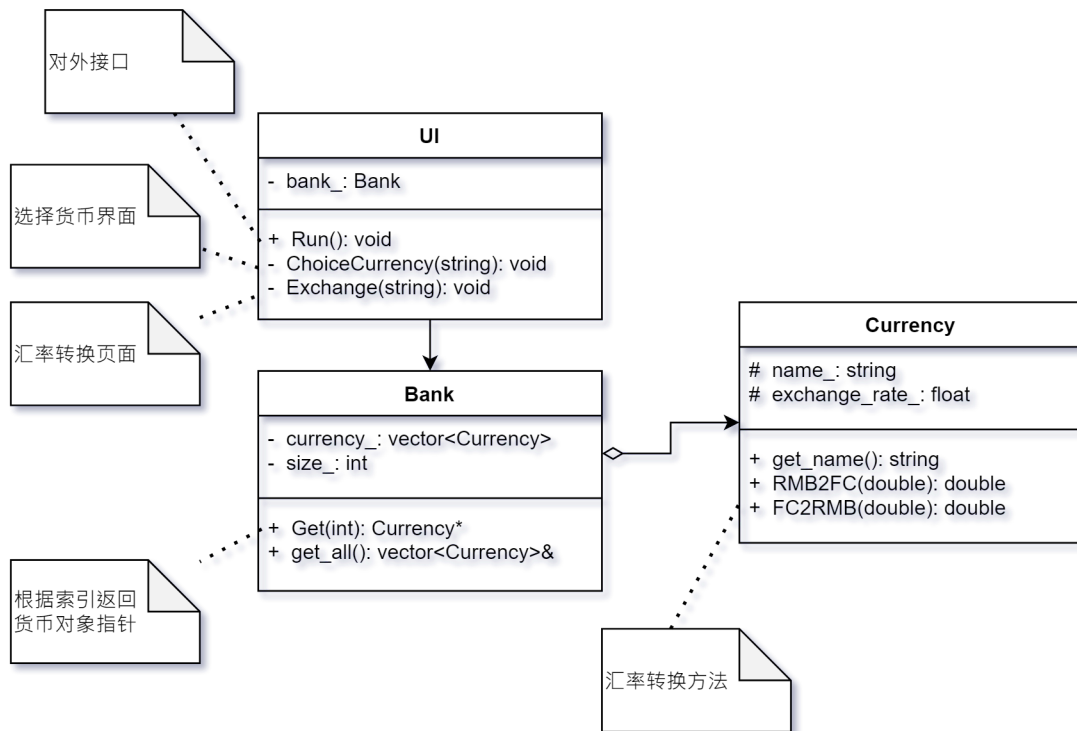
程序实现了RMB兑换外币和外币兑换RMB的功能，同时在各级菜单上方显示当前目录。

设计思路

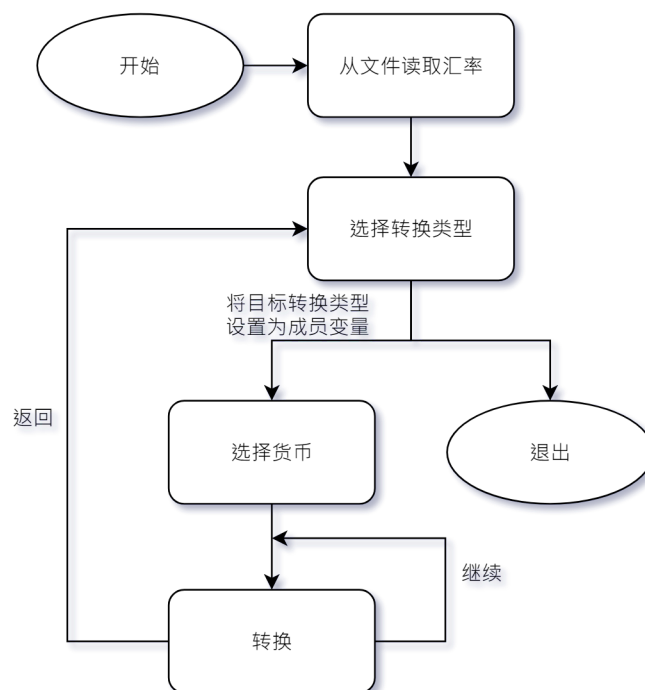
- 先实现一个货币类 `Currency`，使用初始化列表对货币名称和汇率进行初始化。这个类两个主要的成员函数是外币兑换RMB和RMB兑换外币。
- 再实现一个银行类 `Bank`，构造时读入csv档，并将文件中的所有货币分别新建一个货币类的对象，存入vector中。并且通过重载实现了根据货币名或者索引返回相应货币对象的指针。
- 最后实现一个 `UI` 类，进行与用户的交互。

程序框图

外币兑换系统UML图

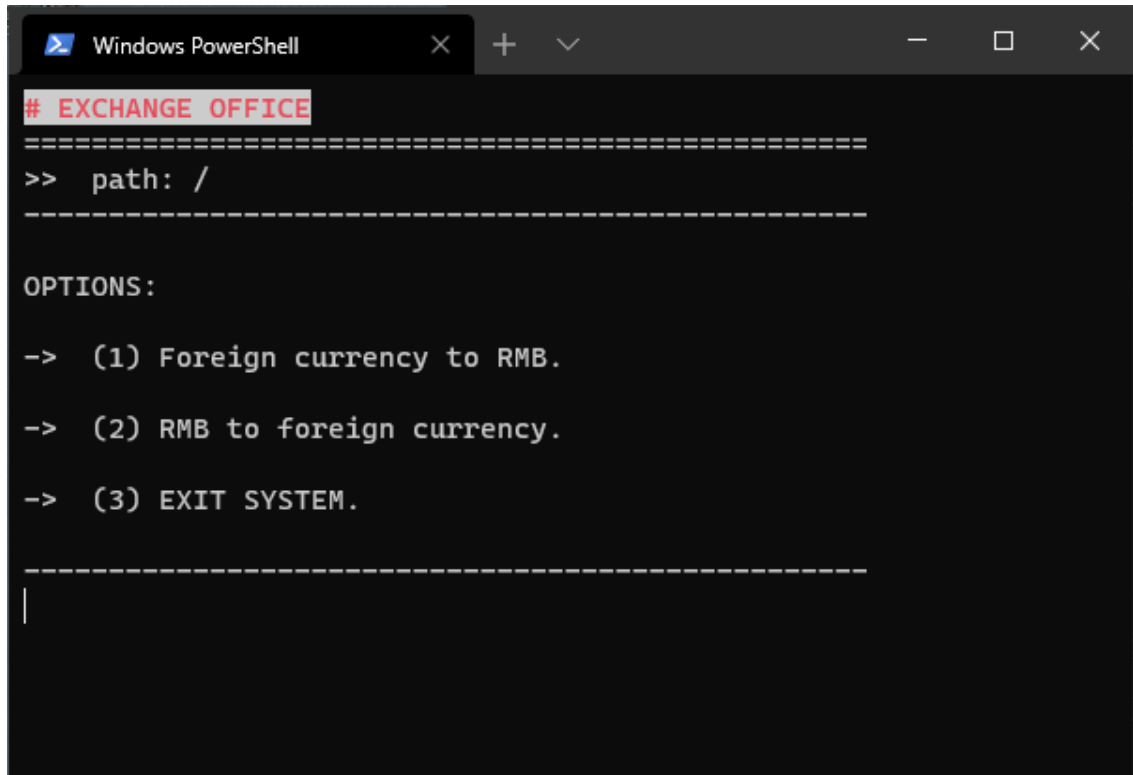


外币兑换系统流程图



运行效果

- 主菜单



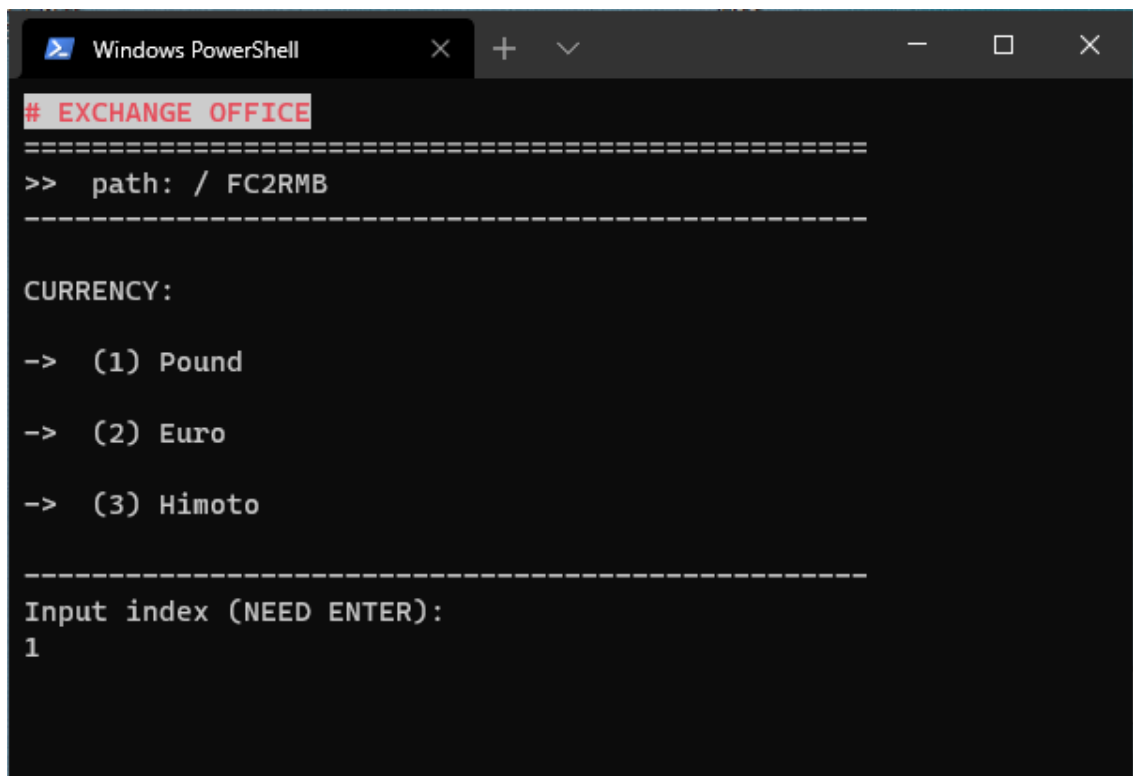
```
Windows PowerShell
# EXCHANGE OFFICE
=====
>> path: /
=====

OPTIONS:

-> (1) Foreign currency to RMB.
-> (2) RMB to foreign currency.
-> (3) EXIT SYSTEM.

=====
|
```

- 选择币种



```
Windows PowerShell
# EXCHANGE OFFICE
=====
>> path: / FC2RMB
=====

CURRENCY:

-> (1) Pound
-> (2) Euro
-> (3) Himoto

=====
Input index (NEED ENTER):
1
```

- 币种配置文件

	A	B	C	D
1	currency name	exchange rate		
2	Pound	869.9		
3	Euro	782.78		
4	Himoto	6.3352		
5				

- 兑换结果

```

# EXCHANGE OFFICE
=====
>> path: / FC2RMB / Pound
=====

EXCHANGE:

      10      Pound  ==>      86.99      RMB

=====

-> (1) Continue.
-> (2) Return.

=====
|

```

总结分析

- 一开始我使用了子类来实现不同货币的不同汇率和名字，但是鉴于没有用到不同的方法，所以并没有太多意义，同时增删改货币类型的时候还需要对代码进行更改和重新编译，故最后并没有使用子类。
- 同时将 `Bank`、类和 `Currency` 类设计成聚合关系，也更加符合逻辑，对货币增删改不需要更改代码。
- 同时单独设计 `UI` 类进行交互，也降低了代码的耦合。

设计内容

程序让用户分别输入分数1、运算符、分数2，并计算结果，将结果作为第二次计算的分数1。

可通过 `clear` 指令对当前待计算公式进行清除，并重新从分数1开始输入。输入 `exit` 退出程序。

程序保存了10条历史计算记录，保存数量可以通过修改交互类中的成员变量 `MAX_HISTORY`。

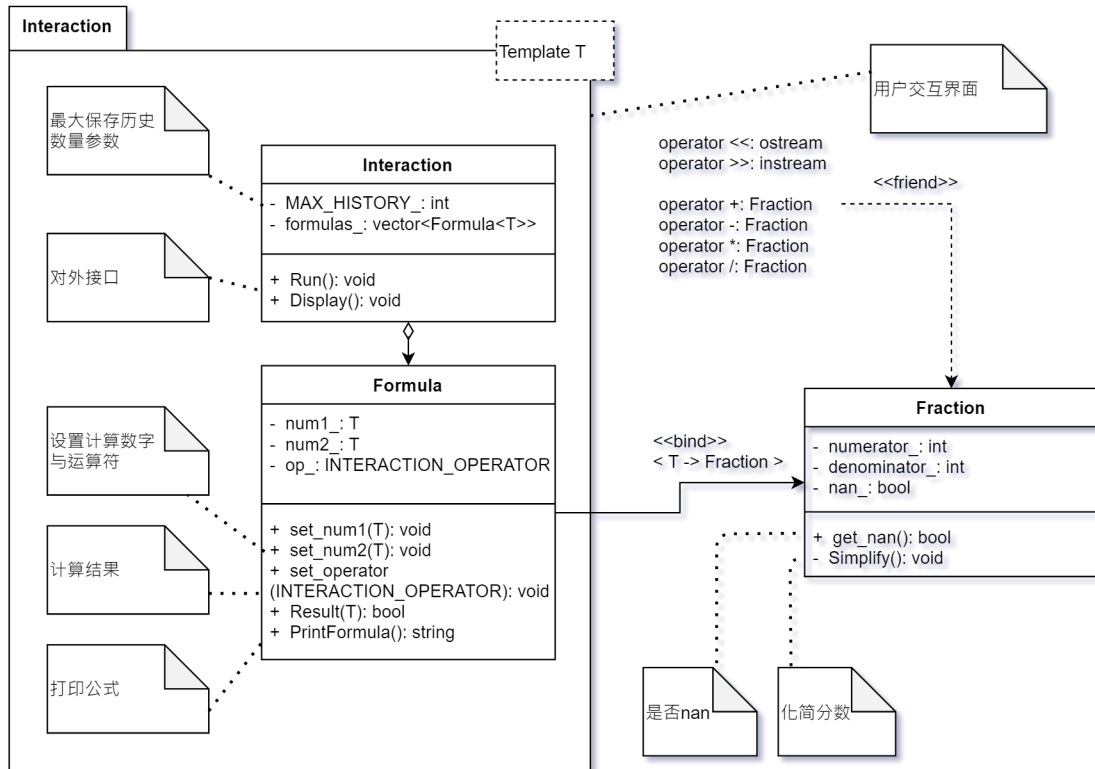
程序能自动转换字符串为分数类，可输入格式形如 $2/3$ 的分数或者形如 6 的整数。同时分数类还能表示 `nan`。

设计思路

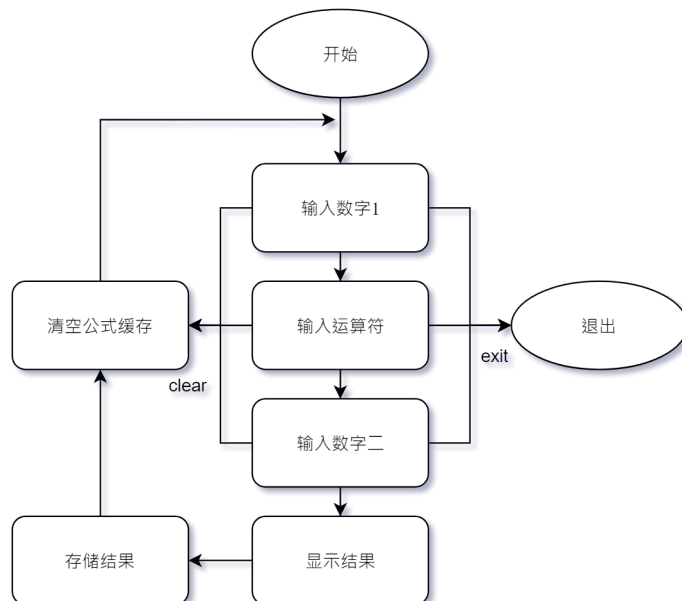
- 分数类 `Fraction` 通过重载可通过使用字符串或者输入分子和分母进行构造。同时构造的时候能够自动对分数进行化简。
- 通过递归实现计算最大公约数，并且使用成员函数 `Simplify` 实现化简功能。
- 通过友元函数重载运算符 `<<`、`>>`、`+`、`-`、`*`、`/`。
- 使用类 `Interaction` 实现人机交互。该类聚合了 `Formula` 类，这个类进行单条公式的存储，显示和计算。通过了模版类的方法，这个类可以实现对本题的分数类进行计算，也可以对下一题的大数类进行计算。使用模版提高了代码复用性，故本题用于实现交互的源代码 `Interaction.h` 和 `Interaction.cpp` 和下一题进行大数计算的源代码是一样的。

程序框图

分数计算器UML图



分数计算器流程图



运行效果

```
Windows PowerShell
# MY HOMEWORK T2
=====
->
->
->
->
->      1/2      +      1/3      =      5/6
->      5/6      -      1        =      -1/6
->      -1/6     *      36       =      -6
->      -6       /      72       =      -1/12
->      12/353   +      24       =      8484/353
->      17/2     /      5/3      =      51/10
-----
>>      51/10   ?      0        =      ?
-----
Please input the OPERATOR: (You can also input 'exit' and 'clear')
clear
```

当前待输入公式 (red arrow pointing to the prompt)

clear 清空当前待输入公式 (red arrow pointing to the 'clear' command)

总结分析

- 使用模版类进行代码的复用。

T3

设计内容

可以实现对用户给得大数进行加减乘除运算，同时输入的大数可以是整数或者是小数，也可以是负数。

可以将用户输入的字符串格式自动转换成 `LargeNum` 类的格式，输入格式可以形如

`10.`、`-0000.1000`、`-.01` 等，当格式不正确会构造出一个nan的大数对象。

通过运算符重载可以实现加减乘除和大于、小于的比较。

显示时当数字小于一会自动使用科学计数法表示。

交互界面功能同上一题。

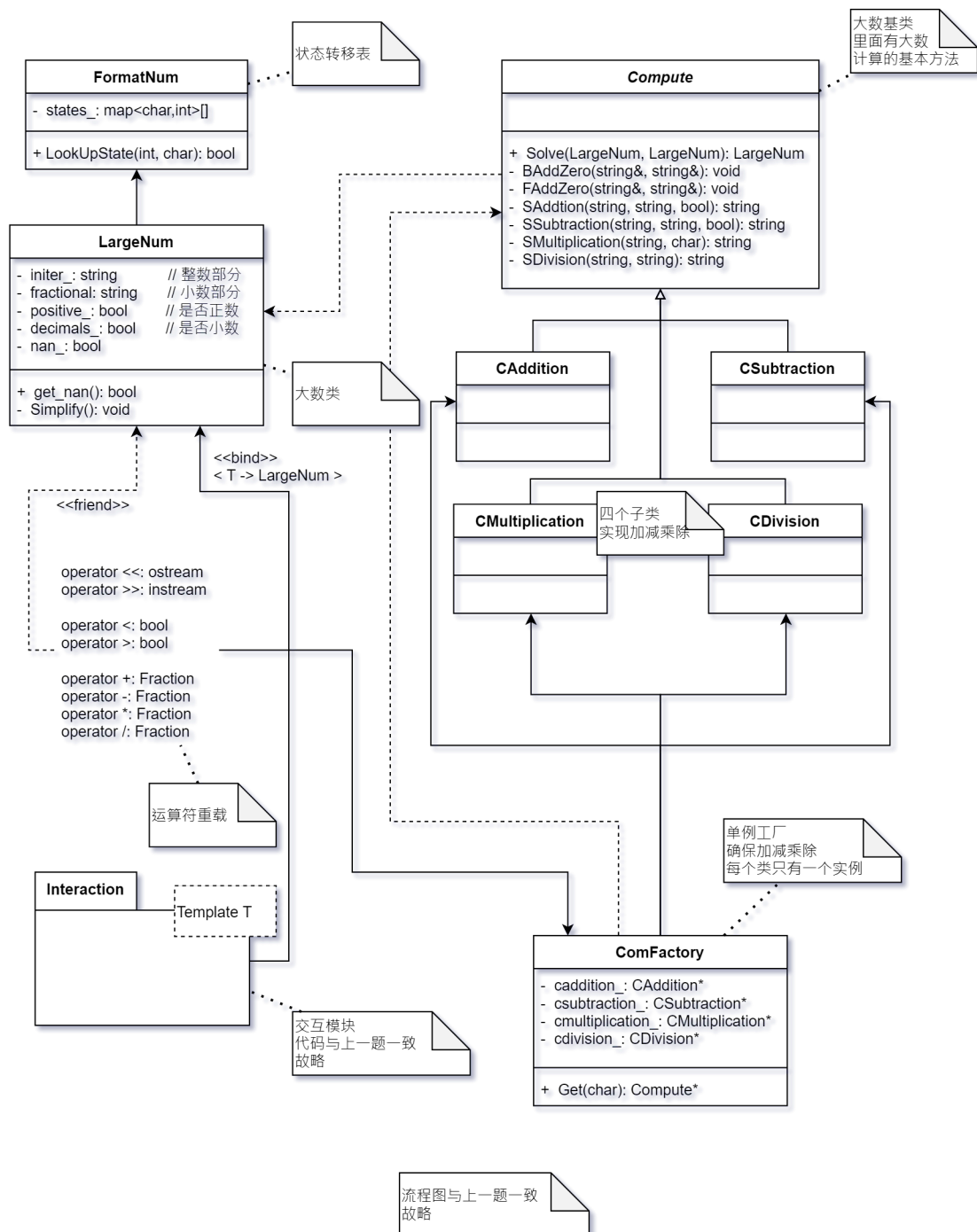
设计思路

- 由于交互界面使用模版类，所以代码与上一题一致，故不做说明了。
- 使用状态转移表对用户输入的数据进行合法判断和格式化。
- 设计大数计算虚基类 `Compute`，里面有成员函数实现功能：前补0、后补0、一位数的加减乘除。通过子类分别实现 `Solve` 方法，使用多态的特性为下一步提供便利。
- 使用单例模式，设计单例工厂 `ComFactory`，使得程序只需要分别构造出一个加减乘除类。在频繁的运算当中，不需要频繁的构造对象，节约了内存和时间。

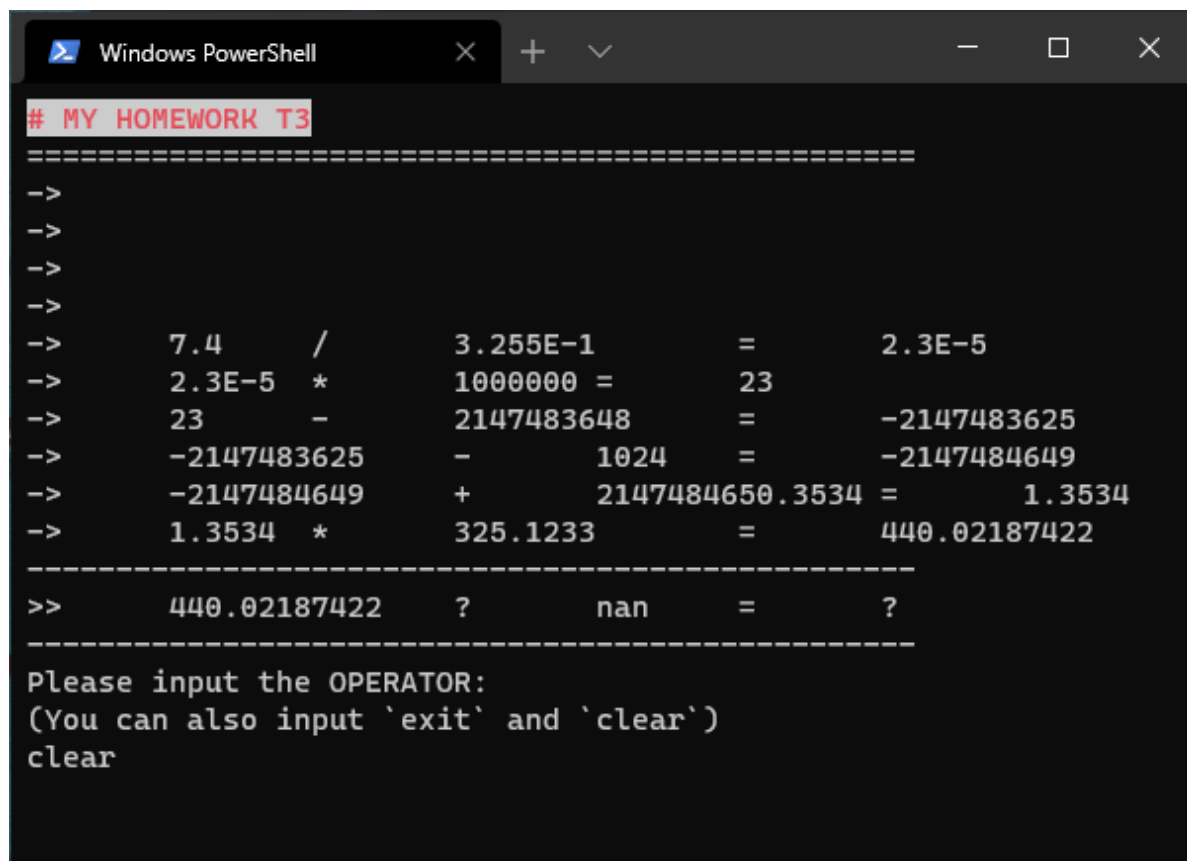
- 通过友元函数重载运算符 $<<$ 、 $>>$ 、 $+$ 、 $-$ 、 $*$ 、 $/$ 、 $<$ 、 $>$ 。
- 因为构造分数类的时候，会自动的将分数化简，所以加减只需要简单的对两个数进行通分之后操作，而乘除也只需要将分子分母进行简单的乘法即可。
- 加法主要是对数字前面补0将两个数长度对齐之后，判断两个数的正负号，并从个位开始逐位相加或者相减。乘法是用乘数逐位相乘，并相加得到结果。而除法则是用被除数与除数循环相减，直到余数为0或者达到指定精度为止。

程序框图

分数计算器UML图



运行效果



```
Windows PowerShell
# MY HOMEWORK T3
=====
->
->
->
->
->      7.4      /      3.255E-1      =      2.3E-5
->      2.3E-5  *      1000000 =      23
->      23      -      2147483648      =      -2147483625
->      -2147483625      -      1024      =      -2147484649
->      -2147484649      +      2147484650.3534 =      1.3534
->      1.3534  *      325.1233      =      440.02187422
-----
>>      440.02187422      ?      nan      =      ?
-----
Please input the OPERATOR:
(You can also input `exit` and `clear`)
clear
```

总结分析

- 使用单例模式，让加减乘除类为唯一实例，节省内存和时间。
- 使用多态的特性，单例工厂输入运算符号，返回不同的子类对象，但返回类型都为基类 `Compute` 的指针。
- 使用状态转移表能比较简单的实现对用户输入合法性进行判断。

T5

设计内容

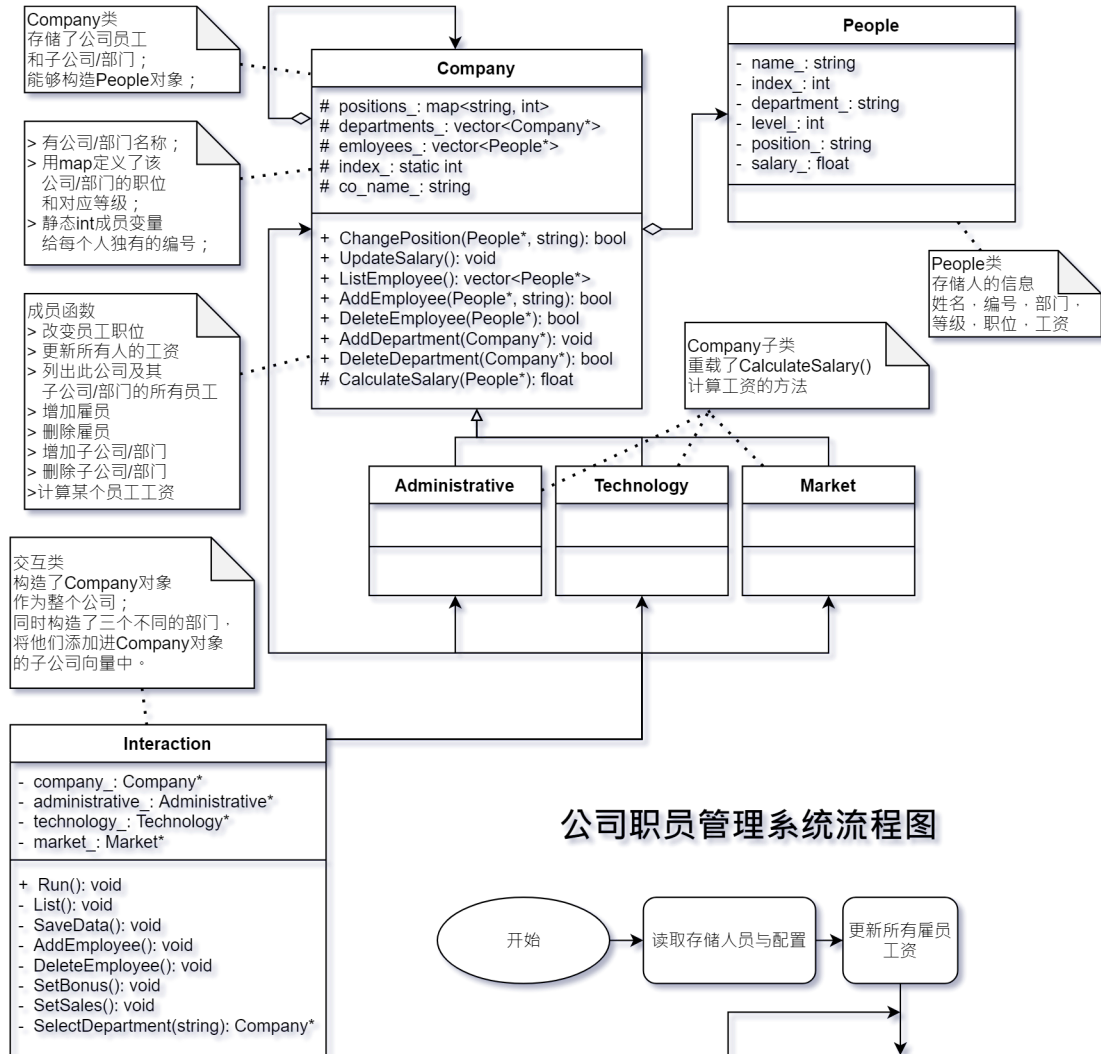
程序能够在运行的时候自动从csv档读入历史存入的员工信息，和相关配置，如：技术部门的奖金和市场部门的销售额，并根据这些配置更新所有员工的工资，实现程序初始化。同时，技术部门的奖金和市场部门的销售额也可以在程序中修改，配置将自动保存。程序可以根据工资升序列出所有员工信息。同时也可以添加员工，只需要给出姓名和部门和职位，工资等其他信息会自动计算生成。另外也可以删除员工还有保存当前所有员工数据，注意程序不会自动保存（自动保存很好实现，只是考虑到员工信息不宜轻易变动）。

设计思路

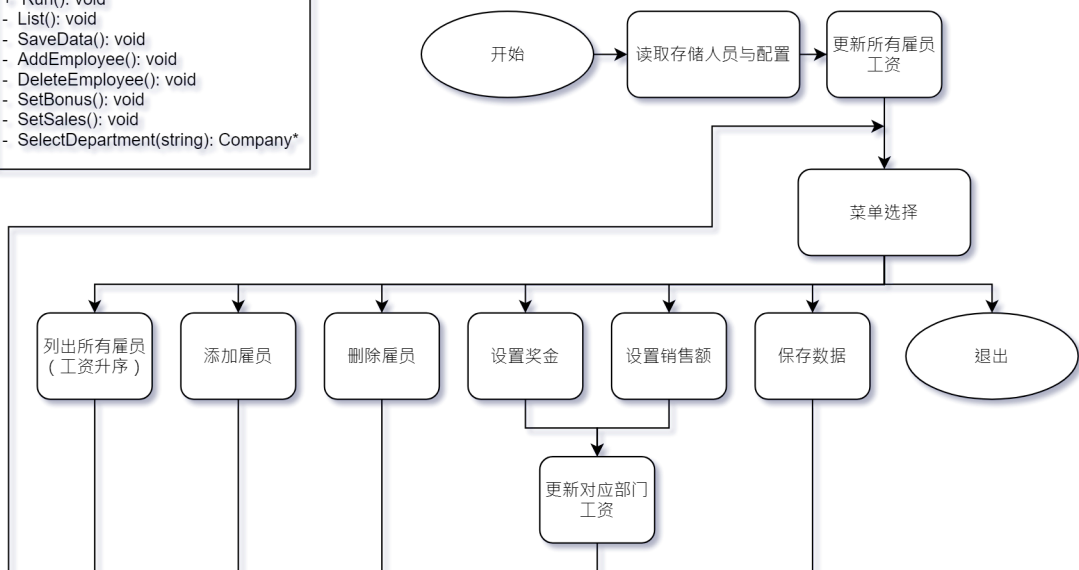
- 程序有两个主要的类，一个是 `Person`，一个是 `Company`。`Person` 类主要存储员工信息的基本的类，不依赖于 `Company` 类，避免耦合。基本信息都是私有成员，可由公有成员函数读取，但不能更改。`Company` 类作为友元函数可以修改，增加了程序的安全性。若日后程序复杂的时候，`Person` 类可以作为父类，也能提升程序扩展性。
- `Company` 类作为基类，程序可以继承并重载 `calculateSalary` 函数，实现了不同子公司/部门，拥有不同的工资计算指标（如：奖金、销售额等）。
- 同时 `Company` 与 `Person` 类为聚合关系，同时与自身也有聚合关系。我们能够清晰的表示总公司和子公司/部门之间的关系，也能反映该公司/部门下的员工，使用递归的方法能获取该公司及其子公司/部门的所有员工。我们能够轻易增删改公司组成结构。如该程序的公司结构是总公司下有管理部门、技术部门、市场部门的扁平化结构，我们也能够轻易将结构改为管理部门下管理着技术部门和市场部门，同时，获取全体员工等方法接口完全不需要做改变。
- 而且这样设计的好处是我们也可以很方便对某一个部门进行操作，如显示全公司的员工和现实某个部门下的员工，我们只需要用不同对象的同一个方法就能实现转换。这让我在编写删除员工UI时，只显示该部门的员工提供了极大的方便。
- 使用sort函数和自定义比较方法实现对工资的排序。用该方法也能根据其他排序（如员工id等）

程序框图

公司职员管理系统UML图



公司职员管理系统流程图



运行效果

- 主菜单

```
Windows PowerShell

-----
OPTIONS:
-> (1) List all employees(Salary ascending).
-> (2) Add employee.
-> (3) Delete employee.
-> (4) Set bonus.
-> (5) Set sales.
-> (6) Save data.
-> (7) Exit.
-----
```

- 员工列表 (工资升序)

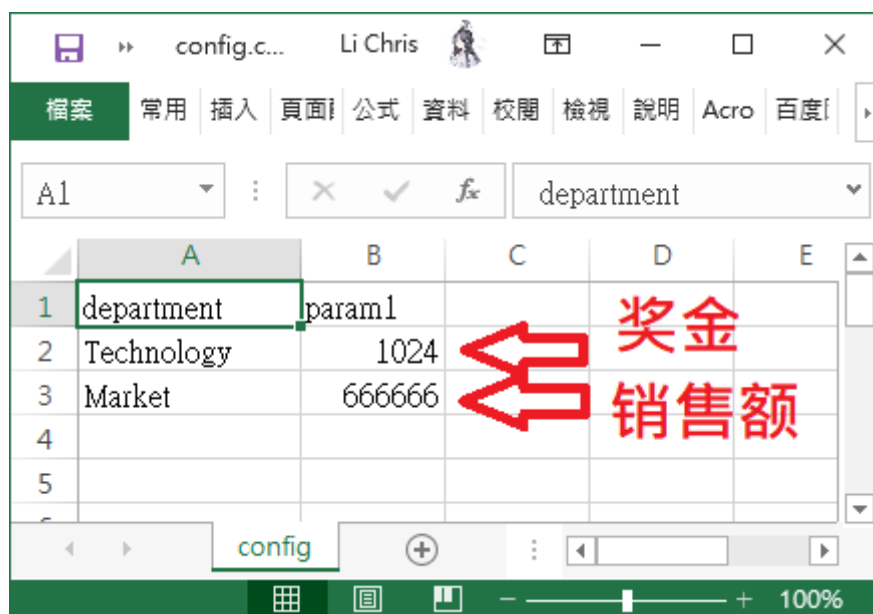
```
Windows PowerShell

# STAFF MANAGEMENT SYSTEM
=====
* List all employees(Salary ascending)
=====

  -index      -name      -department  -level      -position      -salary
->         5      Zhou      Market      1      Salesperson      6666.66
->         3      Sun      Technology  2      Technical staff    7024
->         4      Li      Market      3      Sales manager     11333.3
->         2      Qian     Technology  3      Technical manager  13024
->         1      Zhao     Administrative 4      General manager   20000

-----
Please press 'q' to return...
```

- 公司部门配置文件



- 员工信息存档

	A	B	C	D	E	F
1	department	index	name	level	position	salary
2	Administrative	1	Zhao	4	General manager	20000
3	Technology	2	Qian	3	Technical manager	13000
4	Technology	3	Sun	2	Technical specialist	7000
5	Market	4	Li	3	Sales manager	9000

总结分析

- 使用聚合的方法，每个 `Company` 对象作为一个节点，对他们进行连接可以很直观的表现出某些树形的结构。并且能够使用递归等方法能够方便的对他们进行遍历。
- 由于计算工资的方法每个部门不同，但是只是 `Company` 类中的一个小功能，而其余大量的代码都是可以复用的。我们可以通过继承的方法重载这个方法，用多态方便的实现我们的程序。