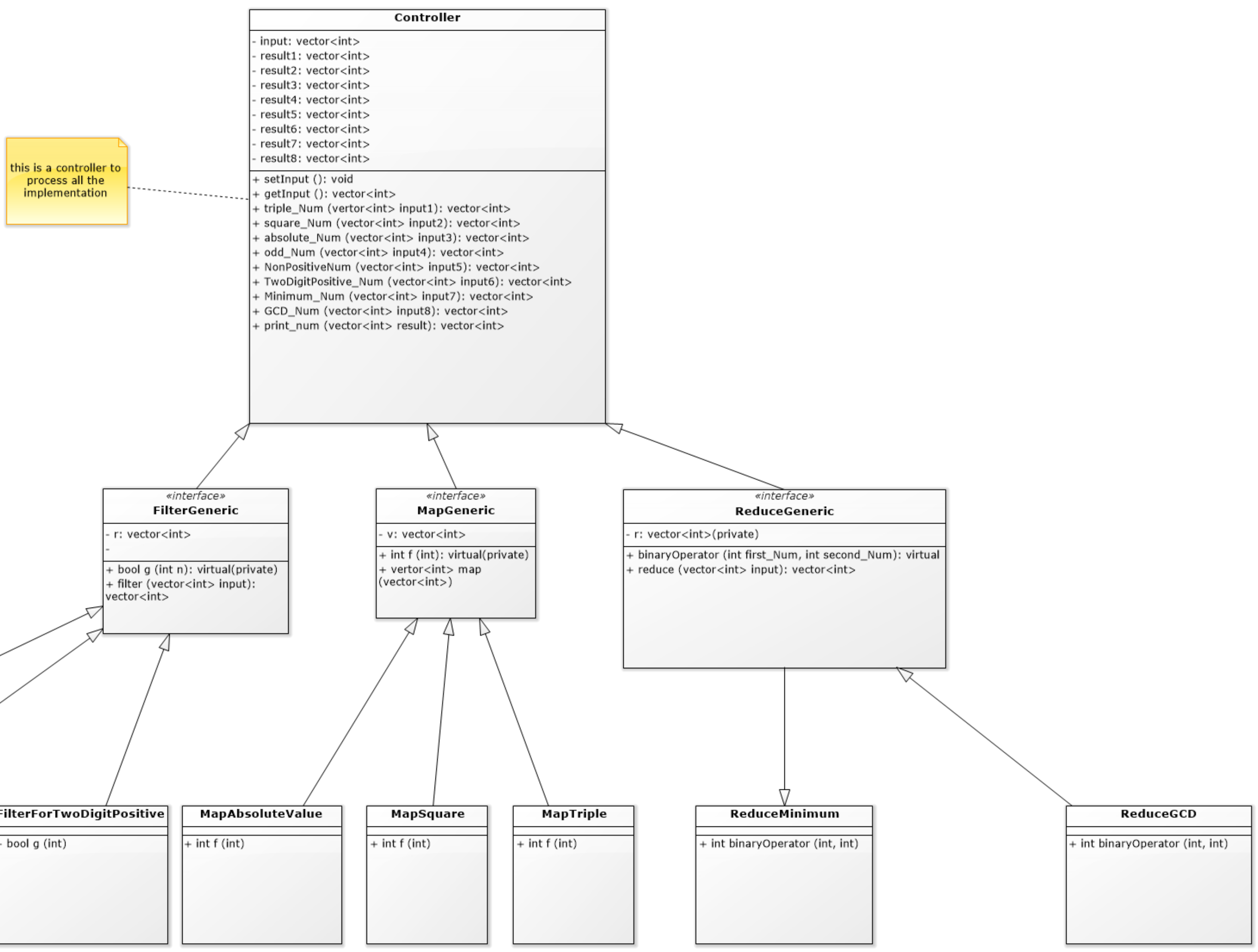
Class Diagram


this is a controller to process all the implementation
Controller
- input: vector<int>
- result1: vector<int>
- result2: vector<int>
- result3: vector<int>
- result4: vector<int>
- result5: vector<int>
- result6: vector<int>
- result7: vector<int>
- result8: vector<int>
+ setInput (): void
+ getInput (): vector<int>
+ triple_Num (vertor<int> input1): vector<int>
+ square_Num (vector<int> input2): vector<int>
+ absolute_Num (vector<int> input3): vector<int>
+ odd_Num (vector<int> input4): vector<int>
+ NonPositiveNum (vector<int> input5): vector<int>
+ TwoDigitPositive_Num (vector<int> input6): vector<int>
+ Minimum_Num (vector<int> input7): vector<int>
+ GCD_Num (vector<int> input8): vector<int>
+ print_num (vector<int> result): vector<int>
«interface»
FilterGeneric
- r: vector<int>
-
+ bool g (int n): virtual(private)
+ filter (vector<int> input): vector<int>
«interface»
MapGeneric
- v: vector<int>
+ int f (int): virtual(private)
+ vertor<int> map (vector<int>)
«interface»
ReduceGeneric
- r: vector<int>(private)
+ binaryOperator (int first_Num, int second_Num): virtual
+ reduce (vector<int> input): vector<int>
FilterNonPositive
+ bool g (int)
FilterOdd
+ bool g (int)
FilterForTwoDigitPositive
+ bool g (int)
MapAbsoluteValue
+ int f (int)
MapSquare
+ int f (int)
MapTriple
+ int f (int)
ReduceMinimum
+ int binaryOperator (int, int)
ReduceGCD
+ int binaryOperator (int, int)

Controller

Attribute:

1) Input //initial an input at the beginning
2) Reesult1//initial a result1 which store the data from the getInput(int)
3) Result2 // initial a result2 to store the value from the triple_Num(int)
4) Result3 // initial a result3 to store the value from the square_Num(int)
5) Result4 // initial a result4 to store the value from the abs()
6) Result5 //initial a result5 to store the value from the odd NUm()
7) Result6// initial a reslut6 to store the value from the NonPositive_Num()
8) Result7 // initial a result7 to store the value from the the TwoDigit_num()
9) Result8 // initial a result8 to store the value from the reduce()

Behaviour

1) setInput()// we use cin to get the input test sample
   we need to be careful an issue that we need to ignore the comma
2) getInput() //return the input back to the the main.cpp
3) triple_Num(input) // this is the multiplication (3*x)
4) Square_Num(input) // this is the square(x^2)
5) Absolute_Num(input) // this is the |x|
6) Odd_Num(input) // select the odd number
7) NonPositiveNum(input)// select the Nonpositive Number
8) TwoDigitPositive_Num(input) //select the Two Digit Positive Number
9) Minimum_Num(input) // the minimum value from the list
10) GCD_Num(input) get the greatest common D from the list
11) Print_num()// print the result

MapGeneric

Attribute:

1) V : vector<int>// initial a vector call v to store the new list from the recursion

Behaviour:

2) Int f(int n)// this is the virtual function in mapping
3) Map(vector <int>) // this is main function and also the recursion function

Derived class:

1) MapSquare: int f(int n); // this is the implementation of square. It will return the square result.
2) Maptriple: int f(int n);// this is the implementation of the multiplication: 3*x;
3) MapAbsoluteValue: int f(int n)// this is the implementation of the absolute value.

ReduceGeneric:

Attribute:

1) r : initialize a vector to store the result

behaviour:

1) binaryOperator(int, int) // the virtual function to get the GCD and minimum
2) reduce(vector<int>) // the main function (recursion). First, we need to get the base case, if it's the last element of the list.
   Otherwise, we need get the value from the binaryOperator(). And push_back the value. After that, we need to erase the first two number.

Derived class:

1) ReduceMinimum : int binaryOperator(int, int) // this is the implementation to get the value of the minimum +69
2) ReduceGCD: int binaryOperator(int, int) //this is the implementation of the the GCD

   Using the Mod and forloop to get the gcd.

filterGeneric:

attribute:

1) V: this is the vector to store the result.

Behaviour:

1) Filter(): main function to get the result: set the base case (if the size is equal to 1), return the result. Else, get the bool g(int n), and erase the value from the input.

Derived class:

1) FilterNonPositive: bool g(int): if is not negative return turn true otherwise return false.
2) FilterOdd: bool g(int): if is odd return true,otherwise return false.
3) FilterForTwoDigitPositive: if it is two digit Positive return true, otherwise return false.

Testing

Input	Description	Output
6, -11, 53, -16, 73, 128, 105, 104, -71, -179, 102, 12, 21, -145, -99, 199, -156, -186, 43, -189	The random number	33 3
1 2 3 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	If the number is too small , maybe we cannot get the two digit number. And no result	No result in case of the crash, we need to give a if statement to make sure the program can be finished
1 32 43	Not enough number	We need to fill it up
A dsa sadasd	Illegal number or character	No output
!@@#$	Illegal input	No output
15, 72, 369, 243, 600, 471, 252, 201, 249, 180, 216, 576, 75, 60, 150, 543, 210, 45, 324, 369	Random number	15 15