Guohao Lin **3. assignment/Task 3** May 2020
IW3XV9
iw3xv9@inf.elte.hu
Group 1

Task

3. Layers of gases are given, with certain type (ozone, oxygen, carbon dioxide) and thickness, affected by atmospheric variables (thunderstorm, sunshine, other effects). When a part of one layer changes into another layer due to an athmospheric variable, the newly transformed layer ascends and engrosses the first identical type of layer of gases over it. In case there is no identical layer above, it creates a new layer on the top of the atmosphere. In the following we declare, how the different types of layersreact to the different variables by changing their type and thickness. No layer can have a thickness less than 0.5 km, unless it ascends to the identical-type upper layer. In case there is no identical one, the layer perishes.

.

	thunderstorm	sunshine	other
ozone	-	-	5% turns to oxygen
oxygen	50% turns to ozone	5% turns to ozone	10% turns to carbon dioxide
carbon dioxide	-	5% turns to oxygen	-

The program reads data from a text file. The first line of the file contains a single integer N indicating the number of layers. Each of the following N lines contains the attributes of a layer separated by spaces: type and thickness. The type is identified by a character: Z - ozone, X - oxygen, C - carbon dioxide.
The last line of the file represents the atmospheric variables in the form of a sequence of characters: T - thunderstorm, S - sunshine, O - others. In case the simulation is over, it continues from the beginning.

The program should continue the simulation until one gas component totally perishes from the atmosphere. The program should print all attributes of the layers by simulation rounds!

The program should ask for a filename, then print the content of the input file. You can assume that the input file is correct. Sample input:

```
4
Z 50
X 80
C 30
X 40
OOOOSSTSSOO
```

Analysis[1]

Independent objects in the task are the gases. They can be divided into 3 different groups (All inherited from the parent class): Ozone, Carbon, Oxygen.

The following table shows how different types of gases will change due to the influence of different atmospheres：

	thunderstorm	sunshine	other
ozone	-	-	5% turns to oxygen
oxygen	50% turns to ozone	5% turns to ozone	10% turns to carbon dioxide
carbon dioxide	-	5% turns to oxygen	-

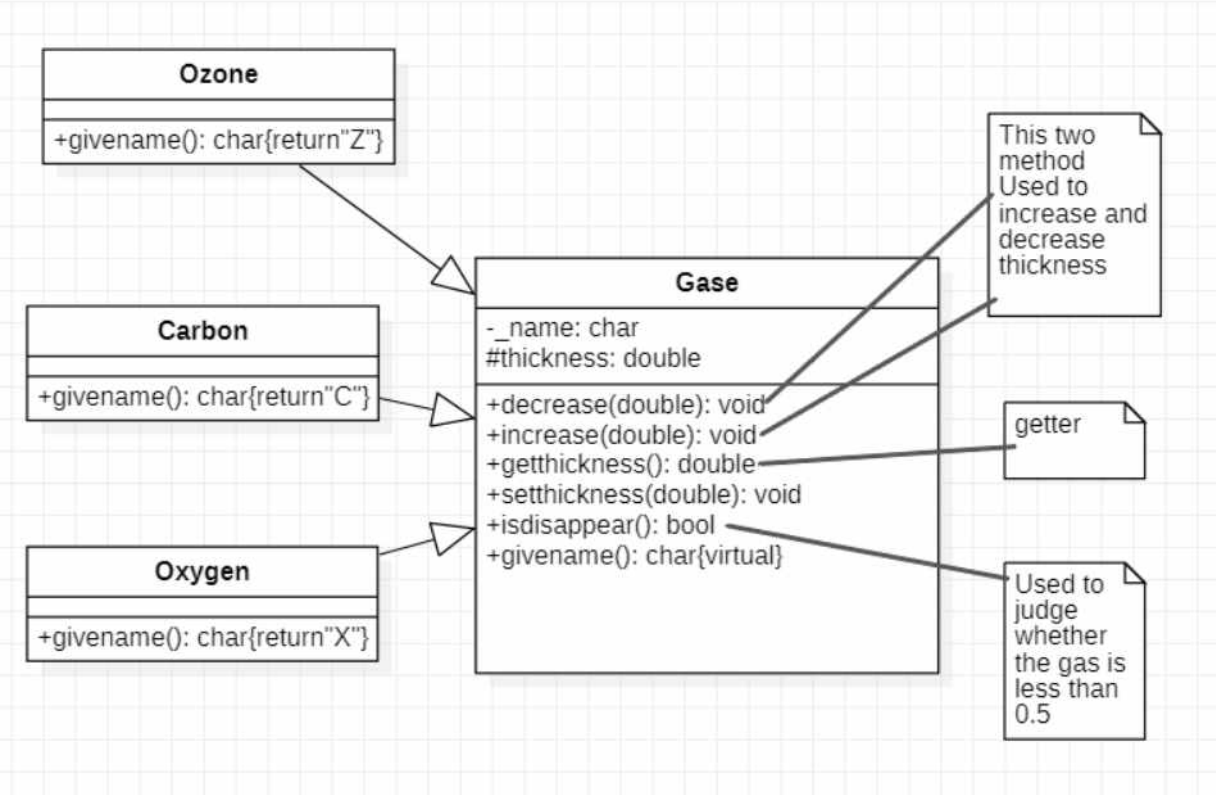
Plan[2]

To describe the gases, 3 classes are introduced: base class *Gas* to describe the general properties and 3 children for the concrete species: *Ozone, Oxygen,* and *Carbon.* Regardless the type of the gas, they have several common properties, like the name *(_name)* and the thickness *(thickness),* the getter of its thickness (*getthickness()*), two methods which can change the thickness(*discrease(), increase()*), a Boolean function which can judge whether this layer should disappear(*isdisappear()const*) and a set method to set the thickness(*thickness()*).

Because we will have three different kinds of gases, and we want these gases to have the above methods, we will inherit the *Gaes* class and override the *givename ()* method. The *givename ()* method will appear as a virtual function in the *Gase* class.

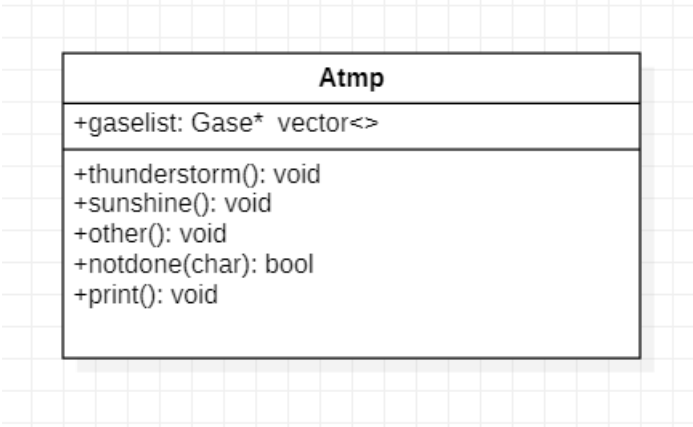
Three different atmospheric conditions can be directly completed in the base class(*Atmp*) with corresponding methods. They are *thunderstorm(), sunshine() and other()*. Corresponding to thunderstorm, sunshine, and other. We don't need any parameters to pass into these methods, because these methods will judge for themselves how each gas will change in the specific atmosphere, and then achieve the goal by changing the collection in the *Atmp* class.

According to the tables, in methods sunshine() ,thunderstorm() and other() conditionals have to be used for a specified gas, otherwise these three functions cannot judge the gas by themselves.



In atmospheric class which called Atmp, we have a vector used to store the pointers come from each layer of gas. We also need a boolean function(*notdone(char*):bool*) which to traversing the vector to determine whether any of oxygen, carbon dioxide and ozone have disappeared from the vector, in this case we will stop. Three methods named *thunderstorm (), sunshine (), and other ()* are used to change the gas pointer stored in the vector in a specified atmospheric environment. We also need a printing method that can print all gases pointed by the pointer in the vector after each simulation.

Among the three methods related to atmospheric variables, we should add a check method, which is used to determine and eliminate those layers with a thickness less than 0.5km in the gas pointed by the pointer in the vector after a single function call.

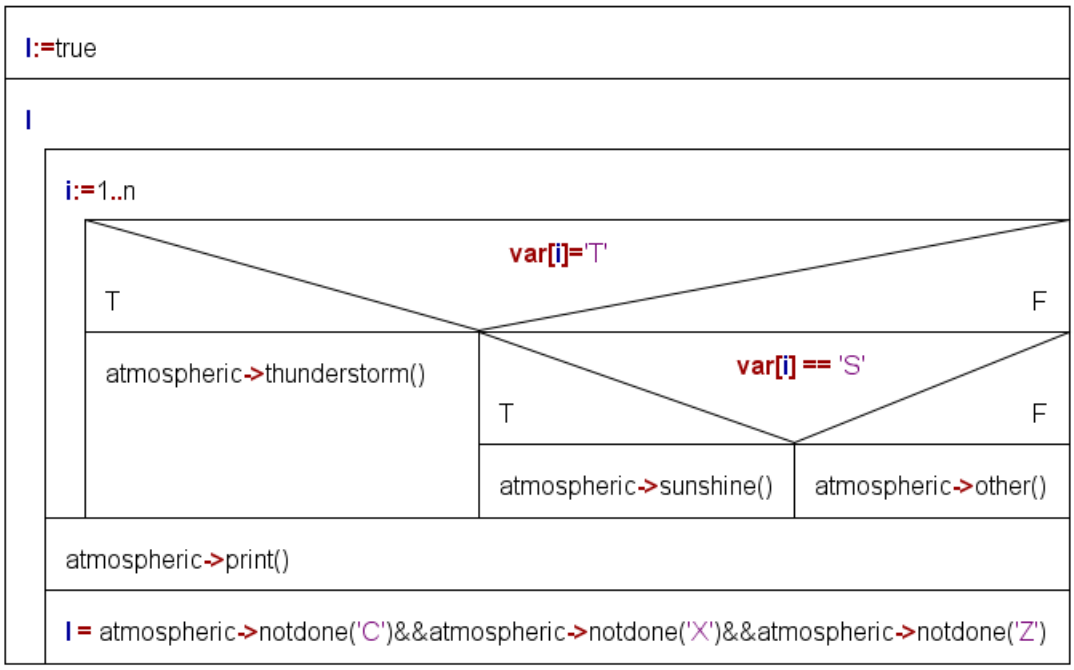


In the main method, we need to convert each layer of information in the file into the corresponding gas extension class and store its pointer in the vector of the instantiated atmosphere class. We also need to convert the atmospheric variables in the last line of the file to a char array. We can assume that this array is named var. Then we call the specified function(*thunderstorm(), sunshine(),other()*) in the instantiated atmospheric class (*Atmp*) according to the atmospheric variable specified in the var. After looping through all the atmospheric variables in var, we call the *print ()* method in the instantiated atmospheric class (Atmp) to print out the information of each layer (gas name, thickness). Then use the *notdone ()* method in this class to determine whether each specific type of gas has disappeared in all layers (disappeared in the vector). If true we stop the simulation, if not we simulate all the atmospheric variables in var again.

A = var: $char^n$, atmospheric: Atmp, l :L

Pre : var := var' , atmospheric: = atmospheric' , l := true

main



Testing

Grey box test cases:

When testing, we judge whether the program is correct by returning the count of simulations.

1. Original file test (input.txt)

This file is given in the task.

2. Repeated gas subclass test (in2.txt, in3.txt, in4.txt)

Used to determine whether each subclass is properly initialized, and whether the program can be correctly identified.

3. Test other methods (in5.txt, in6.txt, in13.txt)

Test whether the other method can perform the correct operation for these three different kinds of gas.

4. Test sunshine method (in7.txt, in8.txt, in14.txt)

Test whether the sunshine method can perform the correct operation for these three different kinds of gas.

5. Test thunderstorm method (in9.txt, in15.txt, in16.txt)

Test whether the thunderstorm method can perform the correct operation for these three different kinds of gas.

6. Test three gas subclasses (in10.txt, in11.txt, in12.txt)

Used to test whether the three gas subclasses can be correctly handled by all atmospheric variables methods and procedures.