Jamolov Isroilbek **2nd assignment/4. Task** 15th May 2023
DXFV5Y
dxfv5y@inf.elte.hu
Group 4

Task

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 layers react 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 the number of layers is the triple of the initial number of layers or is less than three. 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 5
X 0.8
C 3
X 4
OOOOSSTSSOO
```

Analysis

Independent objects in the task are the layer of gases. They can be divided into 3 different groups: Ozone, Oxygen and Carbon Dioxide.

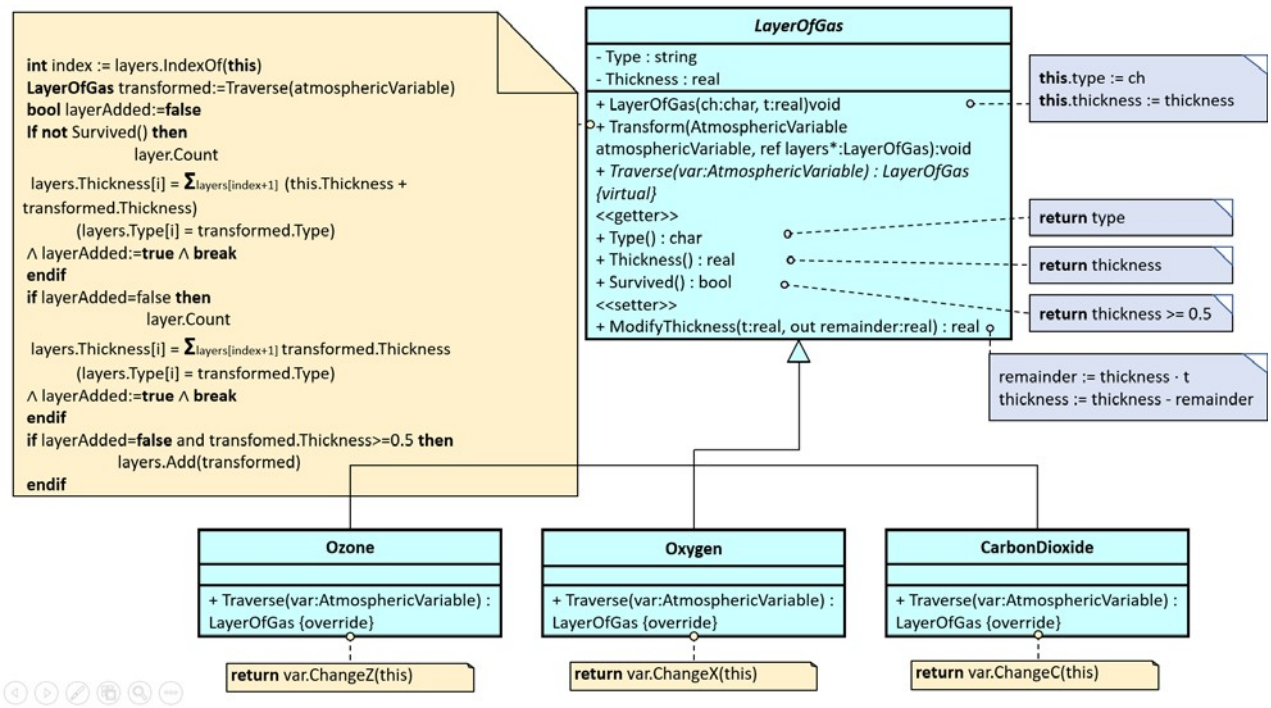
All of them have a type and a thickness that can be got. It can be examined what happens when a prt of one layer changes into another layer due to atmospheric variables. Changing their type and thickness effects the layer of gas and the atmospheric variables in the following way:

	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	-

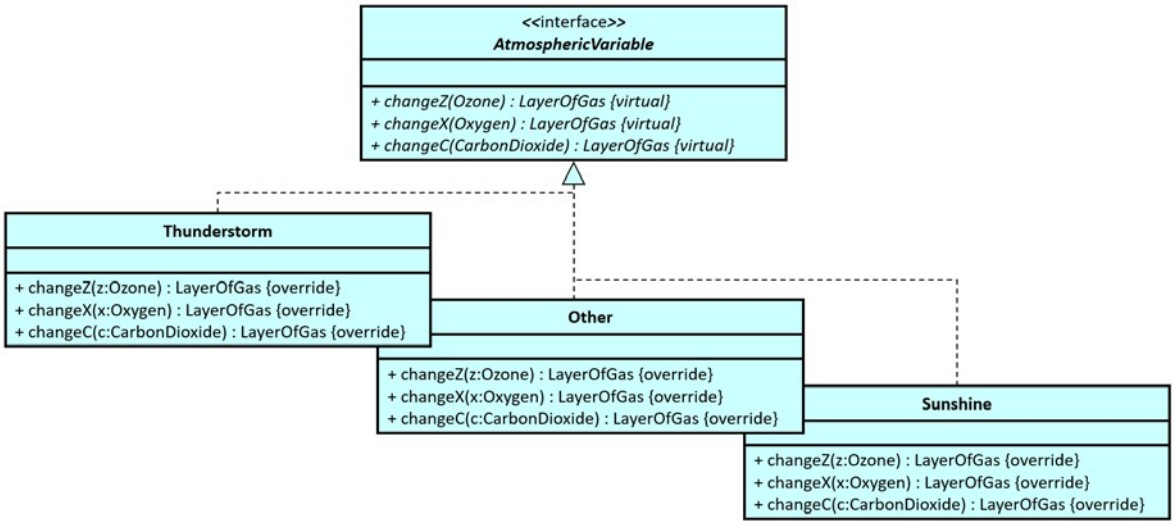
Plan

To describe the layers of gases, 4 classes are introduced: base class *LayerOfGas* to describe the general properties and 3 children for the concrete types: *Ozone*, *Oxygen*, and CarbonDioxide. Regardless the type of the layer of gases, they have several common properties, like the type (*type*) and the thickness (*thickness*), the getter of its name (*type()*), if it is survived (*survived()*) and it can be examined what happens when it changes into another layer due to atmospheric variables. This latter operation (*Traverse()*) modifies the thickness of the layer of gas and *changes* the athmospheric variable. Operations *survived()* and *type()* may be implemented in the base class already, but *Traverse()* just on the level of the concrete classes as its effect depends on the type of the layers of gases. Therefore, the general class *LayerOfGas* is going to be abstract, as method *Traverse()* is abstract and we do not wish to instantiate such class.

General description of the atmospheric variables is done the base class *AtmosphericVariable* from which concrete atmospheric variables are inherited: *Thunderstorm*, *Sunshine*, and Other. Every concrete atmospheric variable has three methods that show how an Ozone, an Oxygen, or a Carbon Dioxide changes during changing the part of one layer into another and how the are explained in Section Analysis. According to the tables, in method *Traverse()*, conditionals could be used in which the type of the atmospheric variables would examined. Though, the conditionals would violate the SOLID principle of object-oriented programming and are not effective if the program might be extended by new atmospheric variables, as all of the methods *Traverse()* in all of the concrete creature classes should be modified. To avoid it, the Visitor design pattern is applied where the atmospheric variable classes are going to have the role of the visitor.



Methods *Traverse()* of the concrete creatures expect an atmospheric variable object as an input parameter as a visitor and call the methods which corresponds to the layer of gases of the layers.



All the classes of the atmospheric variables are realized based on the Singleton design pattern, as it is enough to create one object for each class.

In the specification, the layers's part changing into another layer due to atmospheric variables is denoted by function *Traverse* : *Layer* × *AtmosphericVariable*^{m} → *Layer* × *AtmosphericVariable*^{m} which gives the changed layeror layer itself depending whether it perishes or not. i^{th} version of the atmosphericVariable is denoted by $atmosphericVariable_i$, which the program is not going to show, it is going to be just a temporal value of variable *atmosphericVariable*.

A = *atmosphericVariable*: *AtmosphericVariable*^{m}, *layers*: *LayerOfGas*^{n}, *Survived*: *LayerType**

layerType = rec(Type:Char, Thickness:Double)

Pre = $layers = layers_0 \wedge atmosphericVariable = atmosphericVariable_0$

Post = $\forall i \in [1..n]: layers[i], atmosphericVariable_i = Transform(layers_0[i], atmosphericVariable_{i-1}) \wedge survived = \bigoplus_{ii=1..nn} < layers[ii].Type, layers[i].Thickness > layers[i].survived()$

Concatenation of the layers (after changing the atmosphericVariable) and 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, or in case there is no identical layer above, it creates a new layer on the top of the atmosphere), step by step are two Summations just as the assortment of the survived layers. As all of them are based on the same enumerator, they can be merged into the same loop (*i=1 .. n*).

Analogy:

enor(*E*)	*i = 1 .. n*
f(*e*)	*Transform*(*layers*[*i*], *atmosphericVariable*)$_1$
s	*layers*
H, +, 0	*LayerOfGas**, ⊖, *layers*[*i*]

first component of the value of function *Transform()*

enor(*E*)	*i = 1 .. n*
f(*e*)	*Transform*(*layers*[*i*], *atmosphericVariable*)$_2$
s	*atmosphericVariable*
H, +, 0	*AtmosphericVariable**, ⊖, *atmosphericVariable*

second component of the value of function *Transform()*

a ⊖ *b* ::= *b*

enor(*E*)	*i = 1 .. n*
f(*e*)	< *layers[i]*> if *layers[i].survived()*
s	*survived*
H, +, 0	*LayerOfGas* *, ⊕, <>

By merging the above to the same loop, the solution is got:

survived := <>		
i =1 .. n		
	layers[i]. atmosphericVaraible:= Transform(layers[i], atmosphericVaraible)	
	layers[i].survived()	
	survived := survived ⊕ <layers[i].Type, layer[i].Thickness>	*SKIP*

The i^{th} layer is going to have m+1 states as a part of one layer changes into another layer due to an athmospheric variable which is, in essence, rebuilt (from $atmosphericVariable_{i-1}$ to $atmosphericVariable_i$). 0^{th} state of *layers[i]* is the given layer (*layers[i]*), while the m^{th} state is the layer after the number of elemenets of the layers will be less than 3 or more than triple of its initial number of elements ($layers_m[i] = layers[i]$). The i^{th} layer's part before changing to another layer due to the j^{th} atmosphericVariable is denoted by $layer_{j-1}[i]$ from which the j^{th} state is created by method *Transform()* ($layers_j[i]$) along with the i^{th} state of the atmosphericVariable $atmosphericVariable_i[j]$.
So, the task to be solved is:

$$\forall j \in [1..m]: \quad layers_j[i], atmosphericVariable[j] = Traverse(layers_{j-1}[i], transformation_{i-1}[j]) \quad \wedge$$
$$layers[i] = layers_m[i]$$

A layer's part's changing into another layer means reducing the thickness of the layer and adding that remainder part after transformation to the first identical layer, or in case of there is no identical layer and the remainder thickness is greater than 0.5km creating a new layer on top, step by step, while the concatenation of the traversed atmospheric variables is done, too. Both of them are based on Summation with the same enumerator ($j=1 .. m$):

enor(E)	$j = 1 .. m$	
$f(e)$	$Traverse(layers[i], atmosphericVariable[j])_1$	first component of the value of function *Traverse()*
s	$layers[i]$	
H, +, 0	*LayerOfGas**, ⊖, $layers[i]$	$a \ominus b ::= b$

enor(E)	$j = 1 .. m$	
$f(e)$	$Traverse(layers[i], atmosphericVariable[j])_2$	second component of the value of function *Traverse()*
s	*atmosphericVariable[j]*	
H, +, 0	*AtmosphericVariable**, ⊕, <>	

By merging them into the same loop:

***layers*[*i*], *atmosphericVariable*:=*Transform*(*layers*[*i*], *atmosphericVariable*)**

j = 1 .. m	
	layers[*i*], *atmosphericVariable* [*j*] := *Traverse*(*layers*[*i*], *atmosphericVariable* [*j*])

Testing

Grey box test cases:

1. length-based:
   - zero layer on a zero-long transformation
   - one layer on a zero-long transformation traverses properly
   - two layers on a zero-long transformation
2. correct inputs:
   - three layers and finishing the process until the number of layers are less than 3 (Expected: 2 layers left in the list)
   - three or more layers and finishing the process until the number of layers are triple of initial number of layers(Expected: triple or more layers left in the list)
3. negative thickness. (Expected: the layers should be perished)