Chyong Artur **2nd assignment/4. Task** 19th May 2023
XOG5D9
xog5d9@inf.elte.hu
Group 11

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 atmospheric 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 at 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 di-oxide
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.

Analysis[1]

Independent objects in the task are the layers. They can be divided into 3 different groups: ozone, oxygen, carbon dioxide.

All of them have a name and a thickness that can be got. It can be examined what happens when they get affected by a variable from the list of the variables. They get affected in the following way:

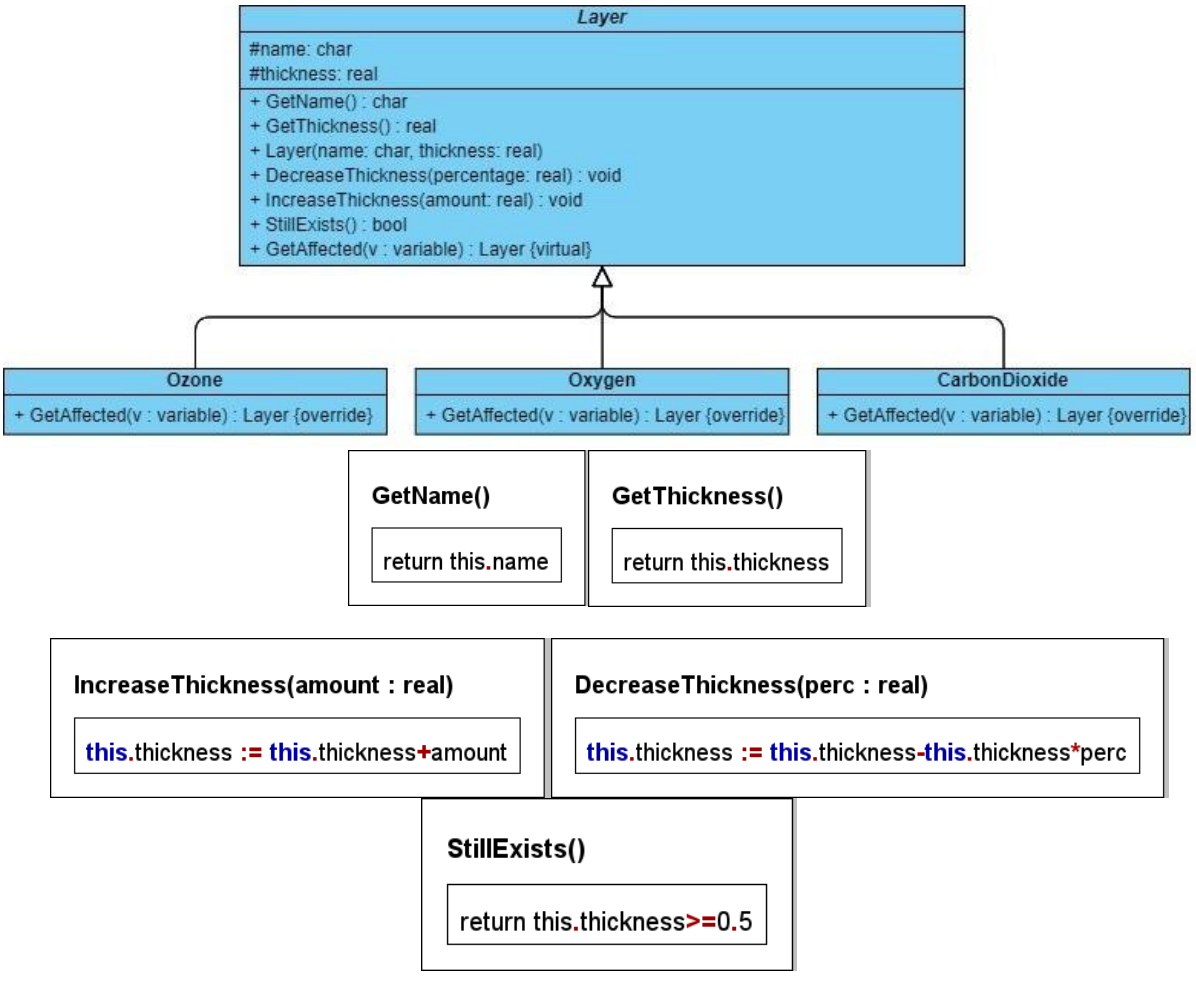
	thunderstorm	sunshine	other
ozone	-	-	5% turns to oxygen
oxygen	50% turns to ozone	5% turns to ozone	10% turns to carbon di-oxide
carbon dioxide	-	5% turns to oxygen	-

Plan[2]

To describe the layers, 4 classes are introduced: base class *Layer* to describe the general properties and 3 children for the concrete layers: *Ozone*, *Oxygen*, and *CarbonDioxide*. Regardless the type of the layers, they have several common properties, like the name (*name*) and the thickness (*thickness*), and methods, like the getters for these properties (*GetName*(), *GetThickness*()), the checker if it still exists(*StillExists*()) and the way it gets affected (*GetAffected*()) by the variable (described further). This latter operation (*GetAffected*()) modifies the thickness of the layer based on the variable it goes through. Operations *StillExists*() and getters can be implemented in the base class, but *GetAffected*() just on the level of the concrete classes as its effect depends on the type of the layer. Therefore, the general class Layer is going to be abstract, as method *GetAffected*() is abstract and we do not wish to instantiate such class.

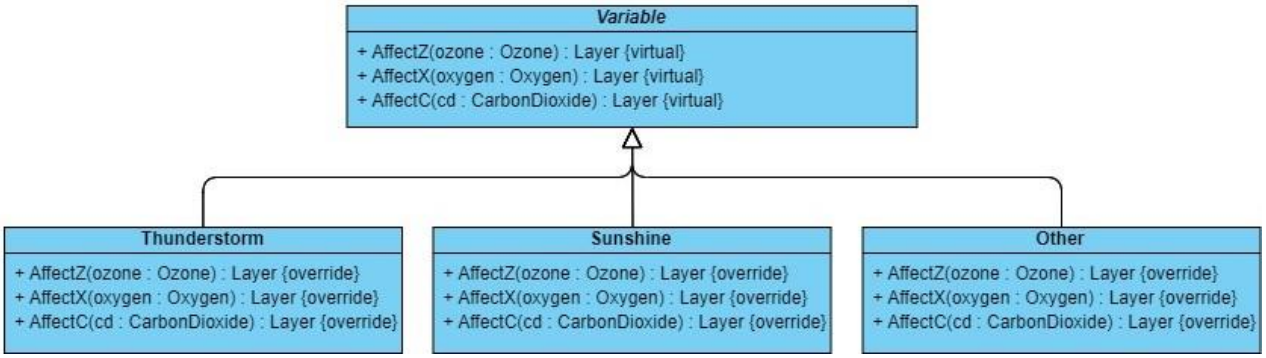
General description of the variables is done. There is the base class Variable from which concrete variables are inherited: *ThunderStorm*, *Sunshine*, and *Other*. Every concrete variable has three methods that show how *Ozone*, *Oxygen*, and *CarbonDioxide* get affected by the variables and their thickness changes.

The special layer classes initialize the name and the thickness through the constructor of the base class and override the operation *GetAffected*() in a unique way. Initialization and the base class are explained in Section Analysis. According to the tables, in method *GetAffected()*, conditionals could be used in which the type of the variable would be 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 variable types, as all of the methods *GetAffected()* in all of the concrete layer classes should be modified. To avoid it, the Visitor design pattern is applied where the variable classes are going to have the role of the visitor.



{Ozone} GetAffected(v : variable)	{Oxygen} GetAffected(v : variable)	{CarbonDioxide} GetAffected(v : variable)
return v.AffectZ(this)	return v.AffectX(this)	return v.AffectC(this)

Methods *GetAffected()* of the concrete layer expect a variable object as an input parameter as a visitor and call the methods which corresponds to the type of the layer.



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

The change is denoted and performed by functions Affect depending on the variable given as a parameter.

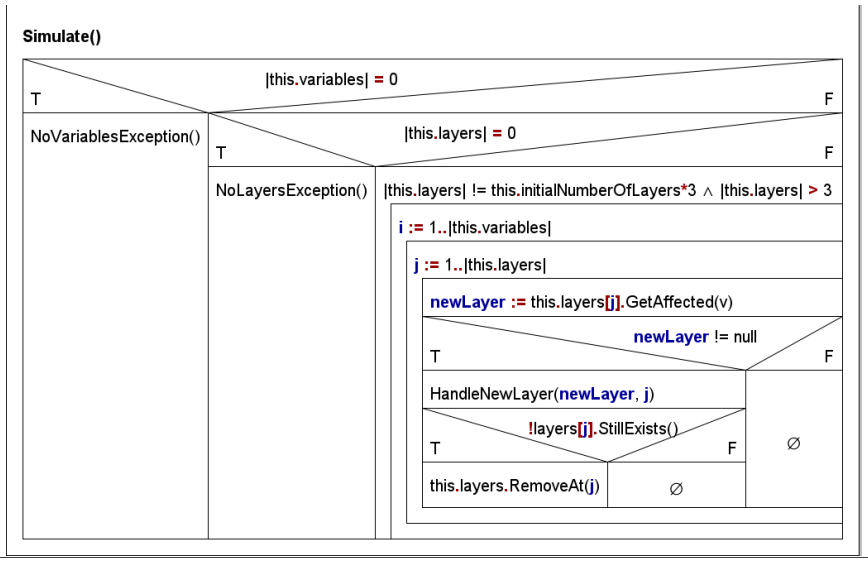
A = layers: $Layer^m$, variables: $Variable^n$

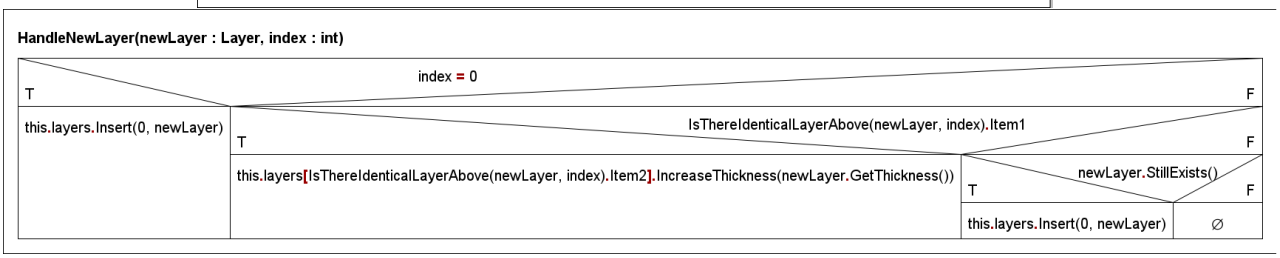
Pre = layers = $layers_0$ ∧ variables = $variables_0$

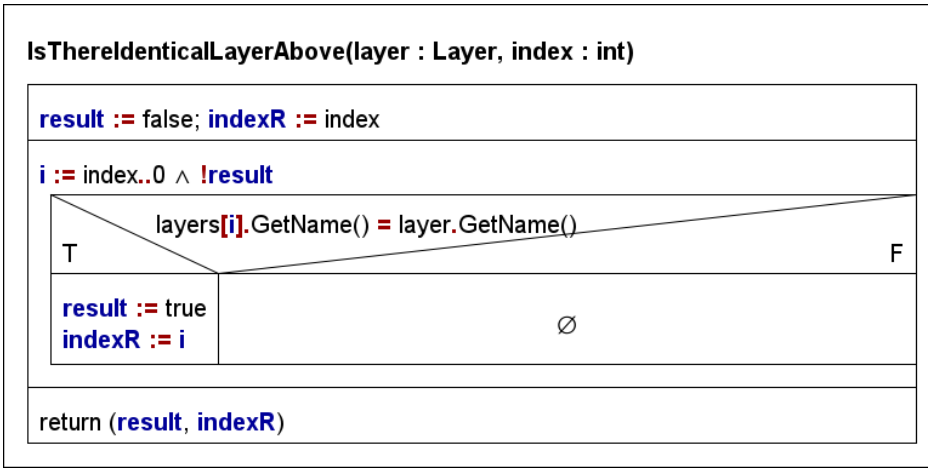
Post = ∀i∈[1..m], ∀j∈[1..n] : layer = layers[i].GetAffected(variables[j]) ∧ HandleNewLayer(layer, i)

|result|>3 ∧ |result| ≠ 3*m

∧ result = result ⊕ layer







A = layers: Layer^{m}, indexR: Z, layer: Layer

Pre = layers = layers$_0$ ∧ indexR = indexR$_0$ ∧ layer = layer$_0$

Post = (result, indexR) = ($\forall i \in [indexR..0]$: layers[i].GetName() $\overset{!indexR}{=}$ layer.GetName())

Testing

Testing the methods (black box testing)

1. Class *Variable,* testing *Affect* methods.
   3 initial different instances of variables (*ThunderStorm, Sunshine, Other*).
   3 layers of each type of layers with thickness 5.
   1) Test Ozone layer and how it gets affected by variables:
      - Thunderstorm: no effect, thickness doesn't change.
      - Sunshine: no effect, thickness doesn't change.
      - Other: new Layer of Oxygen is created with 5% of Ozone's thickness. Ozone's thickness has decreased by 5%.
   2) Test Oxygen layer and how it gets affected by variables:
      - Thunderstorm: new Layer of Ozone is created with 50% of Oxygen's thickness. Oxygen's thickness has decreased by 50%.
      - Sunshine: new Layer of Ozone is created with 5% of Oxygen's thickness. Oxygen's thickness has decreased by 5%.
      - Other: new Layer of Carbon Dioxide is created with 10% of Oxygen's thickness. Oxygen's thickness has decreased by 10%.
   3) Test Carbon Dioxide layer and how it gets affected by variables:
      - Thunderstorm: no effect, thickness doesn't change.
      - Sunshine: new Layer of Oxygen is created with 5% of Carbon Dioxide's thickness. Carbon Dioxide's thickness has decreased by 5%.
      - Other: no effect, thickness doesn't change.
2. Class Layer, testing *Increase* and *Decrease* thickness methods.
   3 layers of every type with thickness 5.
   1) DecreaseThickness() with 0.5 led to the decrease of the thickness by 50% for every layer.
   2) IncreaseThickness() wth 0.5 led to the increase of the thickness by 0.5 for every layer.

Testing based on the code (white box testing)

1. Generating and catching exceptions.