OOP Documentation 2nd Assignment

Torogeldiev Azamat Documentation 02.06.2023HVICMC
hvicmc@inf.elte.hu
Group 5

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 layerof 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 typeand thickness.

No layer can have a thickness less than 0.5 km, unless it ascends to the identical-type upper layer. In case there isno 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 fileis correct. Sample input:

```
4
Z 5
X 0.8
C 3
X 4
OOOOSSTSSO
```

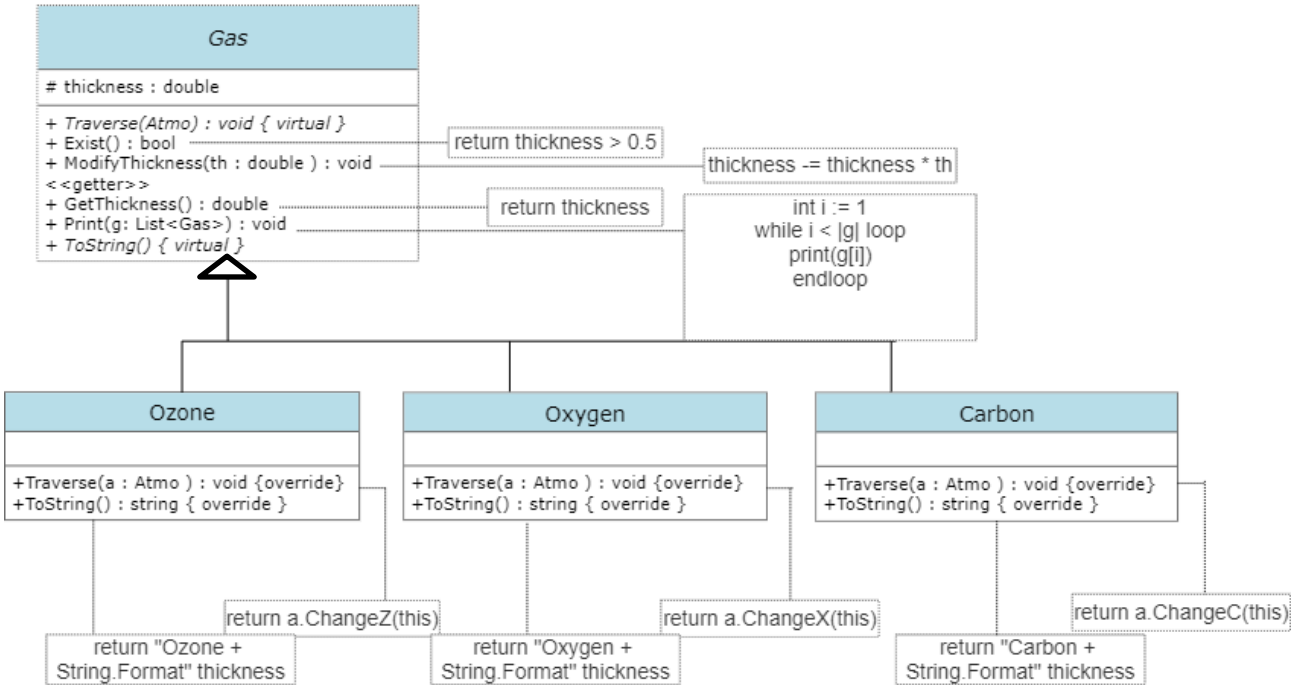
Analysis1

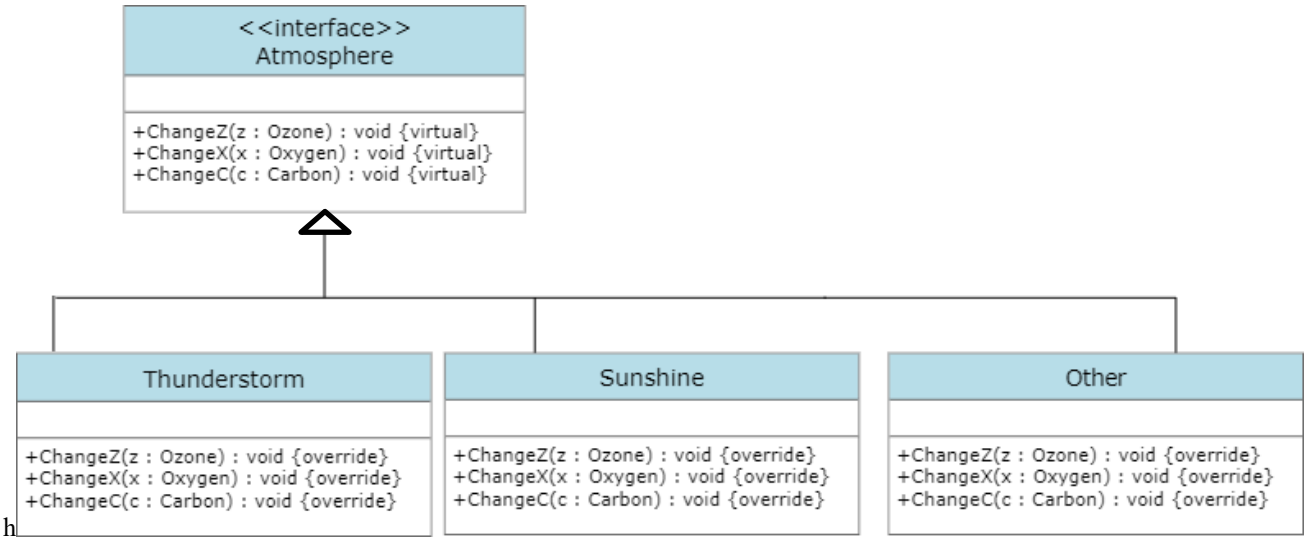
Independent objects in the task are the Gases. They can be divided into 3 different groups: Ozone, Oxygen and Carbon Dioxide. All of them have a thickness that can be got. It can be examined what happens when atmospheric variable effects gases layer.

Plan2

To describe the layers of gases, 4 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 common properties, like the thickness(_thickness), if it is exist (exists()) and it can be examined what happens when atmospheric variable effects. This latter operation (Traverse()) modifies the thickness of the gas layer and adds new layers if it should. Operation exist() may be implemented in the base class already, but Traverse() just on the level of the concrete classes as its effect depends on the species of the gas layer. Therefore, the general class Gas 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 Atmo from which concrete grounds are inherited: Thunderstorm, Sunshine, and Other. Every concrete atmosphere has three methods that show how an Ozone, an Oxygen, or a Carbon ,changes during day it. The special gas classes initialize the thickness through the constructor of the base class and override the operation Traverse() in a unique way. Initialization and the override are explained in Section Analysis. According to the table, in method Traverse(), conditionals could be used in which the type of the atmosphere 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 Gas classes should be modified. To avoid it, the Visitor design pattern is applied where the atmospheric classes are going to have the role of the visitor.

Methods Traverse() of the concrete creatures expect an atmosphere object as an input parameter as a visitor and call the methods which corresponds to the species of the gas





h

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

In the specification, it is necessary to calculate with the n+1 versions of the gas layers as every

Atmospheric variable changes it. The 0th version is the initial gases list. The effect on one creature is denoted

by function traverse : Gas × $Atmo^m$ → × Gas × $Atmo^m$ which gives the changed gas. i^{th} version of the gas layers list is denoted by $gases^i$, which the program is not going to show, it is going to be just a temporal value of variable gases.

A = Atmosphere: $atmo^m$, gas: gas^n , exist: String*

Pre = gas = gas_0 ^ gases = $gases_0$

Post = gases = $gases_n$^

i∈ [1..n]: gas[i], $gases_i$ = transform(gas_0[i], $gases_i$-1) ^

exist = ⨁i=1..n< gas, gases[i]. thickness >

creatures[i].alive()

Concatenation of the gases (after crossing n days) and transmuting the atmosphere step by

step are two Summations just as the assortment of the existing gases. As all of them are based

on the same enumerator, they can be merged into the same loop (i=1 .. n).

first component of the value of function Transofrm()

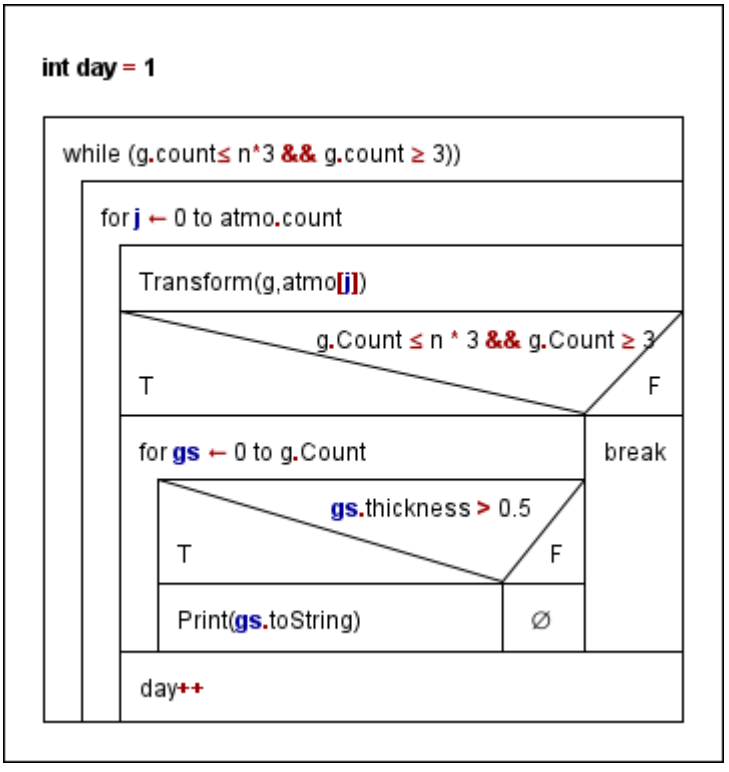
Enor(E)	i = i..n
f(e)	Transform(gases[i],amto)$_1$
s	gases
H, +, 0	Gas*, ⊖, gases[i]

second component of the value of function Transform()

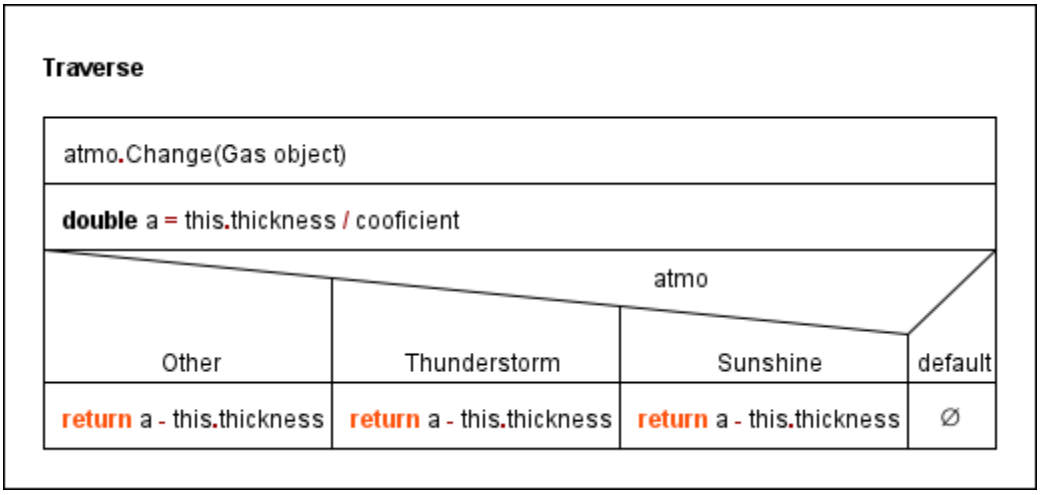
Enor(E)	i = i..n
f(e)	Transform(gases[i],amto)$_2$
s	amto
H, +, 0	Atmo*, ⊖, atmo

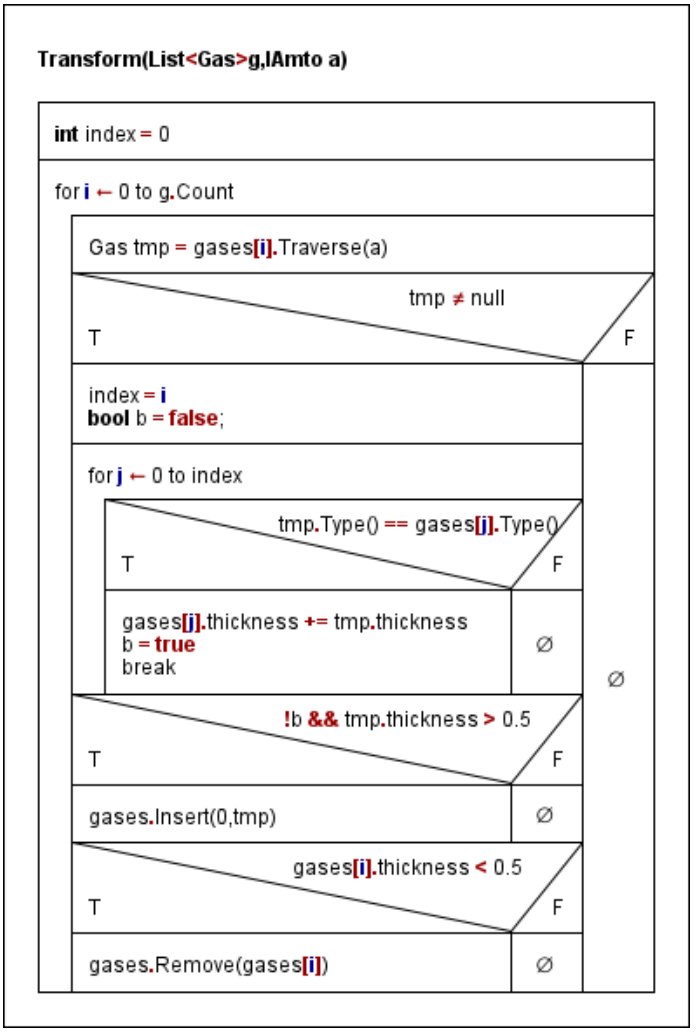
Enor(E)	i = i..n
f(e)	Transform(gases[i],amto)
s	Exist
H, +, 0	Gas*, ⊖, <>

Simulate days and we got a solution



In the Transform(), we change gas thickness, and add or remove gas that should exist or not, by calling function Traverse(). See the structure of the functions Transform() and Traverse() below.





Testing

Grey box test cases:

1)Gas Exists

2) function ModifyThickness works correctly

3)function Traverse works correctly

4)function Traverse works correctly (different variables)

5)Gas exist after Traverse

6)Transform method works correctly