|  |  |  |
| --- | --- | --- |
| **Muhammad Eman Aftab**  IJE4R1 | **2nd assignment/0. Task** | 11th February 2023 |
| [ije4r1@inf.elte.hu](mailto:ije4r1@inf.elte.hu) |  |  |
| Group 1 |  |  |

# Task

4. 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 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.

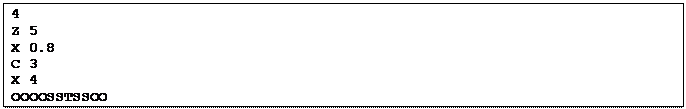
|  |  |  |  |
| --- | --- | --- | --- |
|  | thunderstorm | sunshine | other |
| ozone | - | - | 5% turns to oxygen |
| oxygen | 50% turns to ozone | 5% turns to ozone | 10% turns to carbon di-  oxide |
| carbon dioxide | - | 5% turns to oxygen | - |

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. Sample input:



# Analysis1

Independent objects in the task are the layers. 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 they simulate an atmospheric variable. It effects the layer in the following way:

Ozone:

|  |  |
| --- | --- |
| Atmospheric Variables | Effect |
| Thunderstorm | - |
| Sunshine | - |
| Other | 5% turns to oxygen |

Oxygen:

|  |  |
| --- | --- |
| Atmospheric Variables | Effect |
| Thunderstorm | 50% turns to ozone |
| Sunshine | 5% turns to ozone |
| Other | 10% turns to carbondioxide |

Carbon-Dioxide:

|  |  |
| --- | --- |
| Atmospheric Variables | Effect |
| Thunderstorm | - |
| Sunshine | 5% turns to oxygen |
| Other | - |

1 This part may be skipped. It is enough to show the tables of traverse in the Planning section.

# Plan2

To describe the Layers, 4 classes are introduced: base class *Layer* to describe the general properties and 3 children for the concrete species: *Ozone*, *Oxygen*, and *Carbon-dioxide*. Regardless the type of the Layers, they have several common properties, like the type (*\_type*) and the Thickness (*\_thickness*), the getter of its type (*getType()*), if its thickness is modified (*ModifyThickness()*) and it can be examined what happens when it simulate a layer. This latter operation (*Simulate()*) modifies the thickness of the layer and *changes* the thickness. *Simulate()* can be implemented on the level of the concrete classes as its effect depends on the type of the layer. Therefore, the general class *Layers* is going to be abstract, as method *Simulate()* is abstract and we do not wish to instantiate such class.

General description of the Atmospheric Variable is done in the base class *Atmospheric Variable* from which concrete AtmosphericVariables are inherited: *Thunderstorm*, *Sunshine*, and *Other*. Every concrete variable has three methods that show how a Ozone, a Oxygen, or a Carbon-dioxide changes during simulation.

The special layer classes initialize the type and the thickness through the constructor of the base class and override the operation (*Simulate())* in a unique way. Initialization and the override are explained below in structograms. According to the structogram, in method *Simulate()*, conditionals could be used in which the type of the layer 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 layers, as all of the methods *Simulate()* in all of the concrete layers classes should be modified. To avoid it, the Visitor design pattern is applied where the atmospheric variable classes like Thunderstorm, Sunshine and Others are going to have the role of the visitor. We will Take input with the help of text-file-reader which we pass as an input to our helper class Atmosphere that takes text-file-reader and populates the Layers and Atmospheric Variables and Then run Simulatelayers() Function where we simulate all atmospheric variables on each layer end will get the result after each simulation. Also we will create Atmospheric variables. In atmospheric variables Thunderstorm, Sunshine, and Others, the Singleton Design pattern have to be done this, ensures that only one instance of each class can exist at a time.

2 Plain text explanation is not necessary for the student documentations

A paper with writing on it

Description automatically generated

A paper with writing on it

Description automatically generated

A piece of paper with writing on it

Description automatically generated

# Testing

Grey box test cases:

*Layer Count Based Testing:*

1. length-based:
   * zero layer
   * one layer
   * more layers

*Layer on Atmospheric Variable Testing :*

1. length-based:
   * one layer on a zero-atmospheric variables
   * one layer on a one-atmospheric variable traverses properly
   * one layer on a longer atmospheric variables
2. first and last:
   * first atmospheric variable traverses properly depending on the layer
   * first atmospheric variable traverses properly depending on the layer

*Examination of function simulate()*

Five different cases depending on the layers and the atomospheric variables.