**OOP Bekbolot kyzy Asel: documentation, 2nd assignment 1.**

**Bekbolot kyzy Asel 3. Assignment** 17 June 2023

QECV0G

qecv0g@inf.elte.hu

Group 10

**Task description:**

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.

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 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 5

X 0.8

C 3

X 4

OOOOSSTSSOO

**OOP Bekbolot kyzy Asel: documentation, 2nd assignment 2.**

**Bekbolot kyzy Asel 3. Assignment** 17 June 2023

QECV0G

qecv0g@inf.elte.hu

Group 10

**Analysis:**

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

All of them have a (name) and thickness that can be got. It can be examined what happens when they got into the atmospheric variables. Getting into these variables effects the thickness states of the layers like in the table above.

**Plan2**

To describe the layers, 4 classes are introduced: base class *Layer* to describe the general properties and 3 children for the concrete types: *Oxygen, Ozone, CarbonD. Regardless* the type of the layers, they have several common properties, like the “Name” and “thickness”, the getter of the thickness is “getTHS()” and also “Combine()” method to merge the layers of the same type. “Traverse()” method modifies the properties of the layers, like its thickness. Both “Combine()” and “Traverse()” methods are abstract, so the implementations are in the children classes, which guarantees us the correct changes.

For the description of the variables, Variable class is created. Three other classes: Other, Sunshine, and Thund will inherit from Variable. The class Variable will have three abstract methods: changeOz(), changeCB(), changeO() and Simulation().

So, every variable has 3 methods that show how layers change according to it.

Here we will use **singleton design pattern** as we do not have to instantiate objects of classes Other, Sunshine, and Thunder many times. Simulation() method takes the list of arrays and traverses around each atmospheric variable.

According to the table, conditionals have to be used in which the type of the variable is examined, however, in case we will want to add a new type of variable, then all of the Layer subclasses have to be modified. To avoid this, we use **Visitor design pattern**. The Variable class acts as the visitor, defining methods for modifying layers based on their types.

**OOP Bekbolot kyzy Asel: documentation, 2nd assignment 3.**

**Bekbolot kyzy Asel 3. Assignment** 17 June 2023

QECV0G

qecv0g@inf.elte.hu

Group 10

Simulation(layers[])

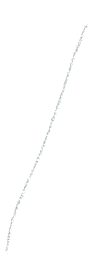
A picture containing text, screenshot, font, line

Description automatically generatedA picture containing text, screenshot, font, document

Description automatically generated



Method Traverse() in the subclasses expects an atmospheric variable, to call the Simulation() method

A picture containing text, handwriting, font, screenshot

Description automatically generated

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.

**OOP Bekbolot kyzy Asel: documentation, 2nd assignment 4.**

**Bekbolot kyzy Asel 3. Assignment** 17 June 2023

QECV0G

Qecv0g@inf.elte.hu

Group 10

Outer loop, where we traverse through variables array and apply Simulation method. In case there is one perished gas component, the loop stops.

A = *variables*: Variablem,  *layers*: Layer*n*

Pre = *layers = layers0* ∧ *variables = variables0*

Post = *variables = variables0* ∧ ∀i∈[1..m] : *variables*[i] = Simulation( layers) ∧ didPerish(layers)=false



For the Simulation method:

A = *layers*: Layer*n*

Pre = *layers = layers0*

Post = <prevLayer> =Traverse(layer[j]) ∧ layers = ⊕ <newLayer> and <prevLayer>

<newLayer> (j=1..n) newLayer.Satisfy()

prevLayer.Satisfy()

Satisfy() function is written to simplify all the calculations and avoid extra details for the postcondition.

Analogy ~ Conditional summation

Sum ~ layers

m..n ~ 1..m

f(e) ~ <newLayer> and <prevLayer>

cond(e) ~ Satisfly()

(+, H, 0) ~ (⊕, Layern , layers)

We have to consider several cases:

\*if in the above layers of the current layer, there exists identical layer to newly transformed layer, then we merge them. In case there is no identical layer above, we check the thickness of this layer. It should be >0.5km

-if yes, then we put this new layer on top of the all layers

-if not, then this new layer perishes

\*If the current layer’s thickness will ever be < 0,5, it should either merge with one of the above layers, or perish.

In the structogram below, all the steps are given:

**OOP Bekbolot kyzy Asel: documentation, 2nd assignment 5.**

**Bekbolot kyzy Asel 3. Assignment** 17 June 2023

QECV0G

Qecv0g@inf.elte.hu

Group 10

A picture containing text, handwriting, paper, ink

Description automatically generated

**OOP Bekbolot kyzy Asel: documentation, 3rd assignment 6.**

**Bekbolot kyzy Asel 3. Assignment** 20 May 2022

QECV0G

Qecv0g@inf.elte.hu

Group 5

**Testing:**

Grey-box testing

Length-based

1. One variable changes 0 layers
2. One variable changes 1 layer
3. On variable changes many layers

First-last

1. first layer of the layers changes properly depending on the types of variables
2. last layer of the layers changes properly depending on the types of variables

**Examination of function simulation**

INITIAL LAYER

1) merges into the identical layer

2) becomes a new layer

3) perishes

NEWLY GENERATED LAYER

1) merges into the identical layer

2) becomes a new layer

3) perishes