OOP Documentation 2nd assignment

Name: Giorgi Gulbatashvili
Date: 24.05.2023
Neptun code: W9NJYS
Group: 10

Task 2

In the hydrological cycle of the Earth, various areas affect the weather as well as areas are also affected by various weathers. Areas involved in the simulation: plain, grassland, lakes region. Each area has a name, and the amount of water stored in the certain area is also given in km3 . The humidity of the air over the areas is also given in percentage.

The possible types of weather are the following: sunny, cloudy, rainy, depending on the humidity of the air. In case the humidity exceeds 70%, the weather gets rainy and the humidity decreases to 30%.

In case the humidity is between 40-70%, the calculation of the chance of rainy weather is: (humidity-30)*3,3%, otherwise the weather is cloudy. Humidity below 40% leads to sunny weather.

> In the following, we declare how the certain areas respond to the different type of weathers. First the amount of water stored by the area varies then the weather will be affected. There is no type of areas with negative amount of water stored.
>
> In case the type is plain, if the weather is sunny, the amount of water will be decreased by 3 km3 ; if cloudy, it will be decreased by 1 km3 ; for rainy weather it will be increased by 20 km3 . The humidity of the air is increased by 5%. If the amount of the stored water is greater than 15 km3 , the plain area changes into grassland.
>
> In case of type grassland: in sunny weather, the amount of water is decreased by 6 km3 , for cloudy it will be decreased by 2 km3 , but and for rainy, it will be increased by 15 km3 . The humidity of the air is increased by 10%. The area becomes lakes region obtaining amount of water over 50 km3 , whereas in case the amount of stored water goes below 16 km3 , the area changes to plain.
>
> In case of type lakes region: in sunny weather, the amount of water is decreased by 10 km3 , for cloudy it will be decreased by 3 km3 , for rainy it will be increased by 20 km3 . The humidity will be increased by 15%. Beyond an amount of water of 51 km3 the area changes into grassland.

The program reads data from a text file. The first line of the file contains a single integer N indicating the number of areas. Each of the following N lines contains the attributes of an area separated by spaces: the owner of the area, the type of the area, and the amount of water stored by the area. In the last line, the humidity of the air is given in

percentage. The type is identified by a character: P – plain, G – grassland, L – lakes region.

We continue the simulation until each area has the same type. The program should print all attributes of the certain areas 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
Mr Bean L 86
Mr Green G 26
Mr Dean P 12
Mr Teen G 35
98
```

Analysis

Independent objects in this task are Areas. they can be divided in 3 different types: Plain, Grassland and Lake area.All of them have name of owner(hereinafter **Name**), some reserve of water(hereinafter **Water**), percentage of humidity(hereinafter **Humidity**).

W**ater** is effected by **Weather** in each traversal (hereinafter one **cycle**).

Humidity is effected based on **type of area** in each cycle and after rainy cycle.

Type of area changes based on **Water** in some cycles.

Weather is determined based on percentage of **Humidity** and changes after some cycles.

All of the effects on area as follows:

Plain:

Weather	Water Change	Humidity change	Humidity Change	Area Change
Sunny	-3	5		becomes Grassland if Water >15
Cloudy	-1			
Rainy	20		set to 30	

Grasslang:

Weather	Water Change	Humidity change	Humidity Change	Area Change
Sunny	-6	10		becomes Lake if Water >50; becomes plain if Water <16;
Cloudy	-2			
Rainy	15		set to 30	

Lake:

Weather	Water Change	Humidity change	Humidity Change	Area Change
Sunny	-10	15		becomes Grassland if Water <50;
Cloudy	-3			
Rainy	20		set to 30	

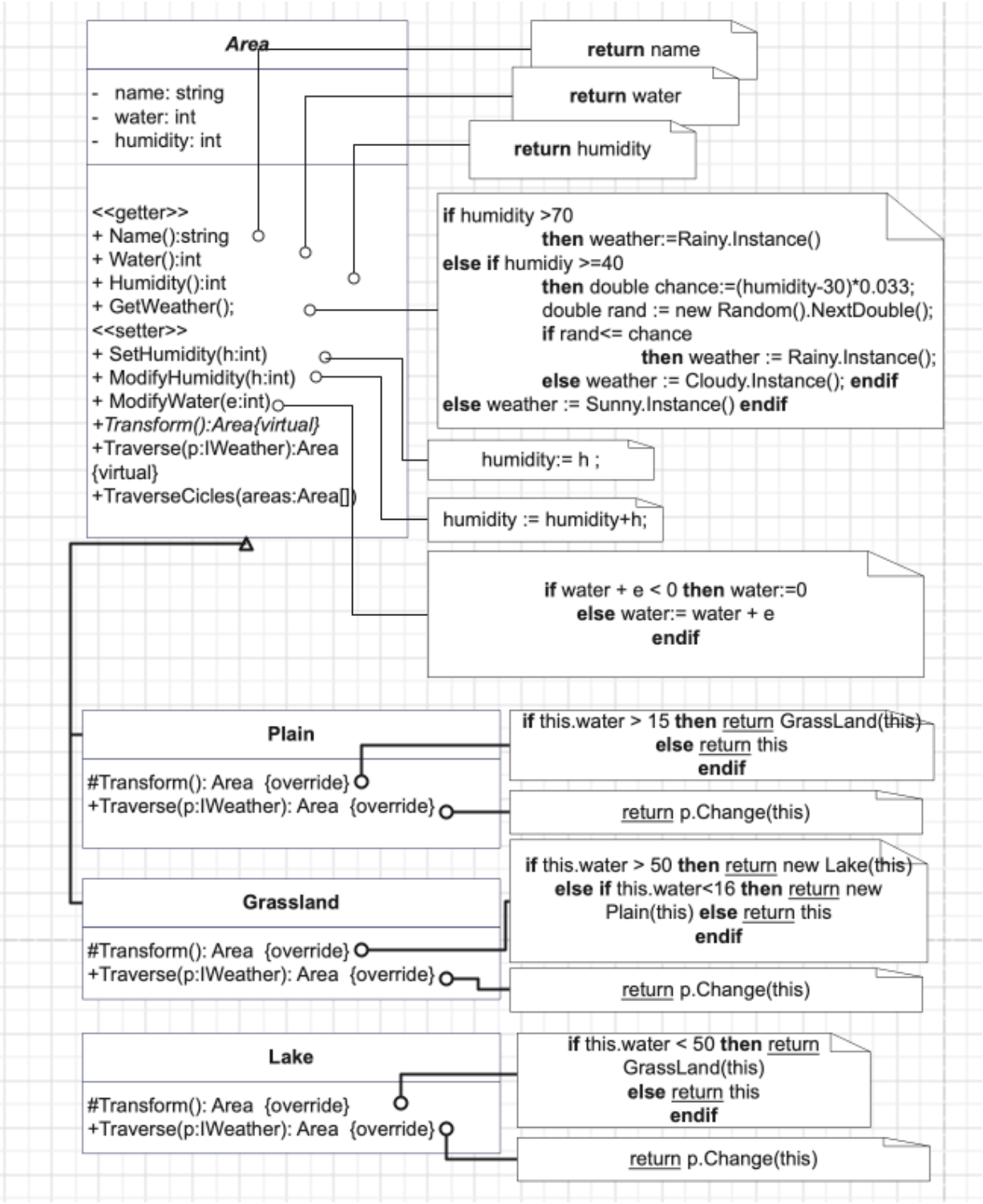
Plan

To describe the Areas 4 classes are introduced: base class Area to describe the general properties and 3 children for the concrete types: Plain, Grassland and Lake; Regardless the type of Area they have several common properties, like (name), (water) and (humidity) which are initialised through the constructor (Area(name:String , water: int)) of the base class. Also regardless the type of Area they have common methods like getters for the fields(in C# properties) (Name),(Water),(Humidity), (Weather), Copy-constructor(Area(a:Area)) and other methods effecting properties of Area:(setWeather()), (setHumidity(a : int)), (modifyWater(a: int)), (modifyHumidity(a:int)). All of the methods described so far can be implemented in base class. Abstract methods (Transform()) (which returns instance of Area) and (Traverse()) can be implemented just on the level of the concrete classes as its effect depends on type of Area. Therefore general class Area will be abstract as well.

Classes Plain,Grassland and Lake initialise name, water through the constructor of the base class and humidity with setter. They implement method Traverse() and Transform() according to the tables. According to the tables, in method Traverse(), conditionals could be used in which the type of the Weather 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 ground types, as all of the methods Traverse() in all of the concrete Weather classes should be modified. To avoid it, the Visitor design pattern is applied where the Weather classes are going to have the role of the visitor.

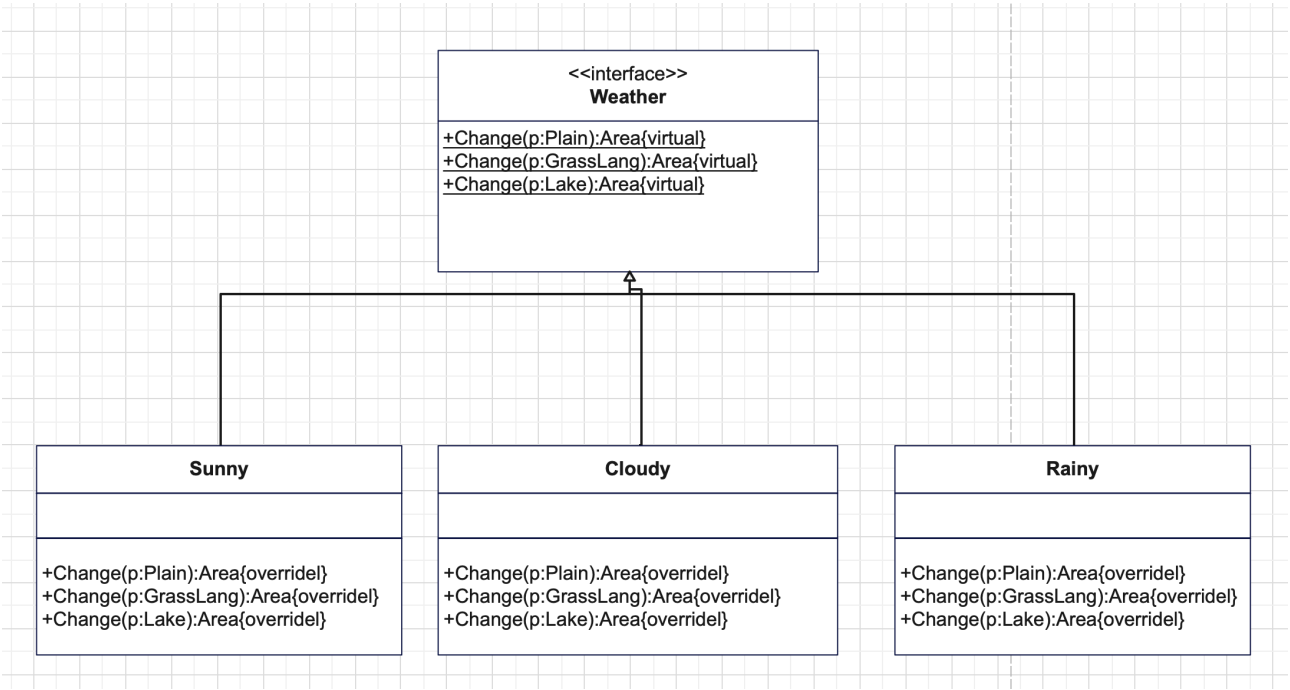
General description of the Weather is done the base class Weather from which concrete grounds are inherited: Sunny, Cloudy, Rainy. Every concrete Weather has three methods that show how a Plain, GreenLand and lake ares changes during cycle and how the Weather changes, too.

UML Diagram(Area)



Methods Traverse() of the concrete Area expect a Weather object as an input parameter as a visitor and call the methods which corresponds to the specific area and weather.

UML Diagram(Weather)



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

We are using two enumerators:
AreaEnumerator traverses list of Areas and after first() operation returns true with current() if first and second element have the same type. After Next() operation Current returns boolean again if second and third element have the same types, and so on until it traverses all list.

CycleEnumerator traverses boolean values returned from AreaEnumerator and sums logical values with logical and (&&) operator. First() operator traverses AreaEnumerator once. next() operation is used to traverse AreaEnumerator again. Current returns True after any complete traversal(after First() or Next()) if all the data types of the modified areas in original list are the same, same time End() is set to true, to finish traversal.

enor(Area) // AreaEnumerator in code

Area*	First()	Next()	Current()	End()

Areas:Area*
b:bool
n: N
count: N
	count := 0;
	areas[count] :=
	areas[count].Traverse(areas[count].GetWeather());
	print(areas[count])
	count++;
	Next();
		if (count = elements){count++;return; }
		areas[count] :=
		areas[count].Traverse(areas[count].GetWeather());
		print(areas[count])
		count++;
			return GetType().Equals(areas[count - 1].GetType());	count = elements + 1;

A=(Areas:Area$^{[1..n]}$)
pre=(|Areas|>1)
post = (Areas[1] = Areas[1].Traverse(Area.GetWeather()) ∧ (Areas[2] = Areas[2].Traverse(Area.GetWeather())∧ Areas'= Area[1]∪ Area[2]∪ [Area[2..n]]∧

a(Areas':Area$^{[m..n]}$)
pre=(|Areas$^{[m..n]}$|>0)
post = (Areas[m] = Areas[m].Traverse(Area.GeWeather()) ∧ Areas"= Area$^{[1..m-1]}$∪ Area$^{[m]}$∪ Area$^{[m+1..n]}$
//where Area[1] to Area[m-1] is already traversed elements by enumerator

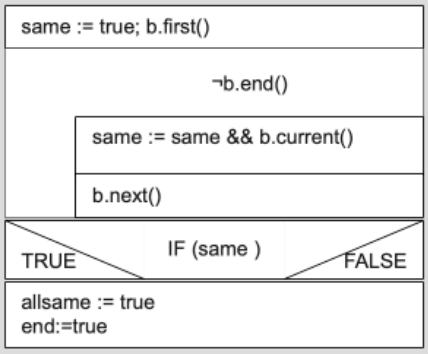
enor(L) // CycleEnumerator in code

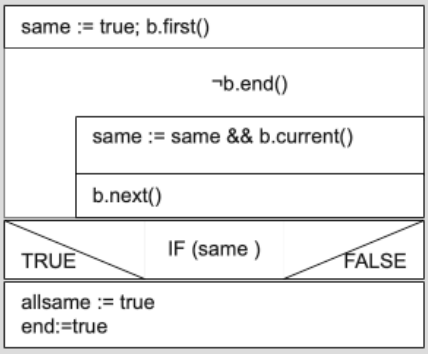
L*	First()	Next()	Current()	End()

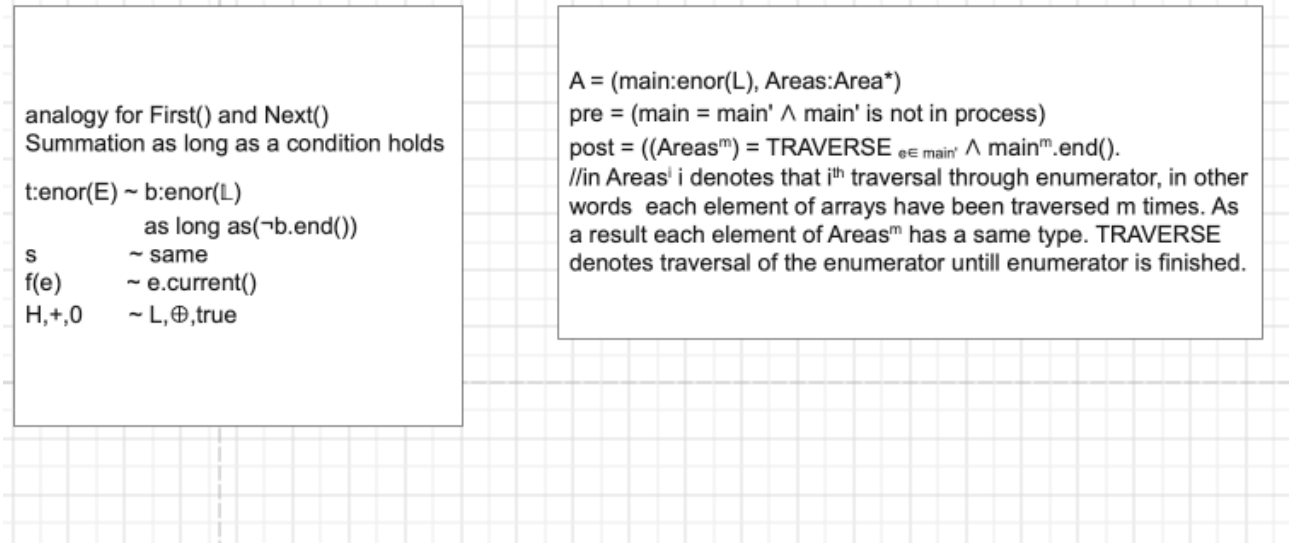
b:enor(L)
Allsame:L
same: L
end: L
		if (count = elements){count++;return; }
		areas[count] :=
		areas[count].Traverse(areas[count].GetWeather());
		count++;
			return
			allsame
				return
				end

A = (b.enor(L)^0, same: L,)
pre = (b = b' ∧ b' is not in process)
post = ((same, b^1) = $\oplus_{e\in b'}$ e.current() (¬b.end())∧ b^0.end().
//in b^i i denotes that enumerator has been traversed ith times. Since we had to travers every area inconditionally we used summation design pattern in code.⊕ denotes summation(with logical and) of logical values.

A = (b.enor(L)^m, same: L,)
pre = (b = b' ∧ b' is not in process (bmm^{m-1}.end())
post = ((same, b^{m1}) = $\oplus_{e\in b'}$ e.current() (¬b.end())∧ b^m.end().
//in b^i i denotes ith traversal through enumerator. Since we had to travers every area inconditionally we used summation design pattern in code.⊕ denotes summation(with logical and) of logical values.







We call main enumerator(main.enor(L)) in main program, which traverses the areas as long as type of areas are not the same.

Testing

Grey box test cases:

1)length based:

- Two land
- More lands
- Check if first land is traversed proper
- check if last land is traversed properly

2)checking constructors
3)checking getters, setters
4)checking Traverse methods
5) checking Traversecicles method