Nazifa Nazrul Rodoshi
W8CZNE
w8czne@inf.elte.hu
Group 4

2nd Assignment Documentation

24th May 2023

Task

Create a program that continues the simulation until each area of the output is of the same type. The program prints all attributes of the areas by simulation rounds.

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.

The program should continue the simulation until each area has the same type. The program should print all attributes of the certain areas by simulation rounds.

Analysis

To note, there are two distinct parts at play here, the areas and the weather. There will be an area parent class and a weather parent interface; not all members of the area class will need to be implemented by the relevant child classes but all members of the weather counterpart will need to implemented by its child classes. Both the area and the weather have three types of entities. The area subclasses will inherit from the area superclass while the weather sub interfaces will implement from the weather base interface. The humidity will affect the weather.

For ensuring the existence of only one instance of certain classes, the Singleton design pattern is used.

Weather	**Plain**	**Grassland**	**Lakes Region**
Sunny	Water decreases by 3km^3	Water decreases by 6km^3	Water decreases by 10km^3
Cloudy	Water decreases by 1km^3	Water decreases by 2km^3	Water decreases by 3km^3
Rainy	Water increases by 20km^3	Water increases by15km^3	Water increases by 20km^3

The above table shows the specific changes in water amount for the different areas based on the changing weather conditions.

Area	Water level>15km^3	Water level>16km^3	Water level>51km^3
Plain	To Grassland	-	-
Grassland	-	To Plain	-
Lakes Region	-	-	To Grassland

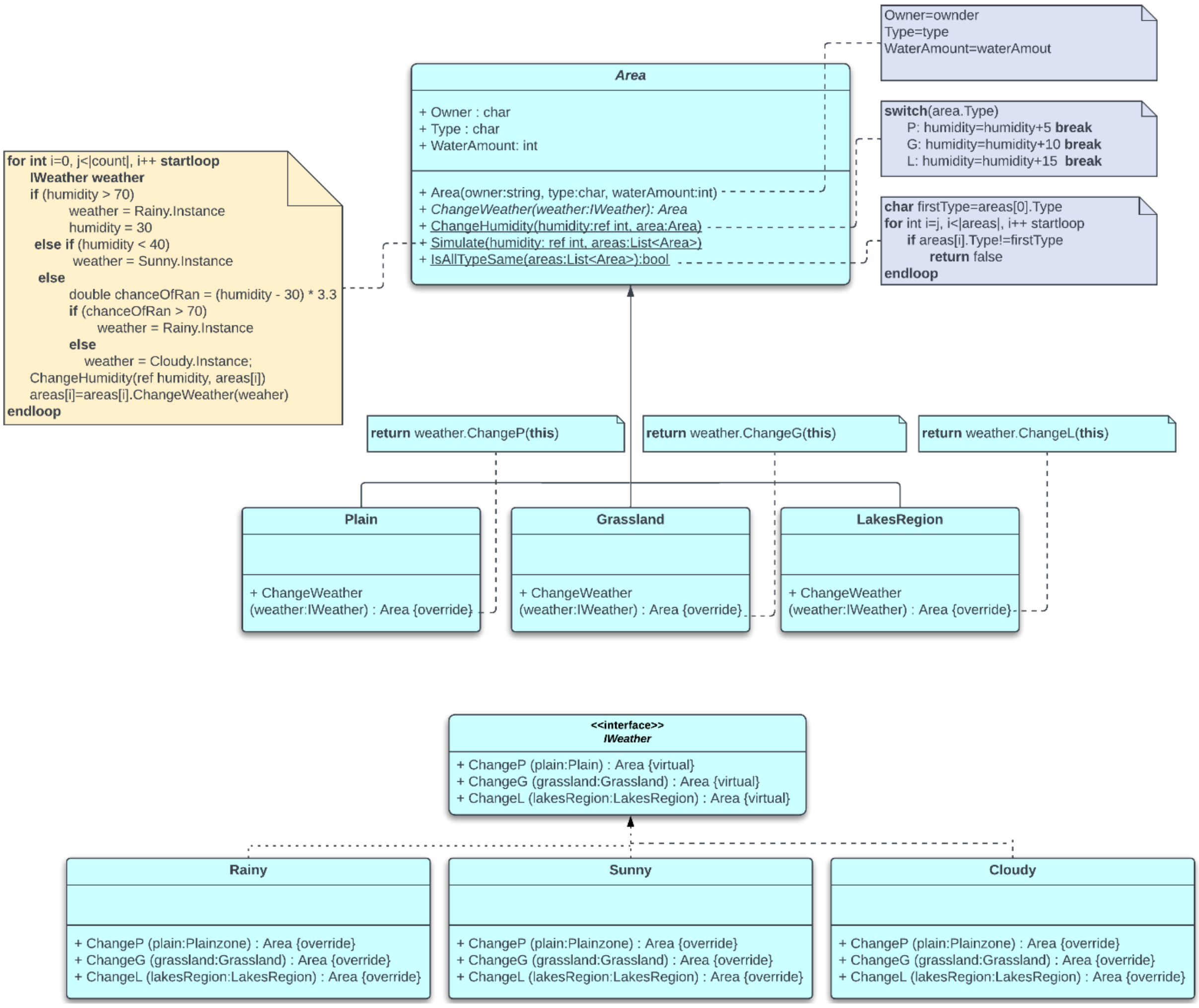
(*Note: If the water level is above 50 km^3 , the Grassland will turn to Lakes Region*).
The above table shows the which areas changes to which type, depending on their water level, after being affected by the weather (effect showed in the first figure).

Plan

For the areas, 4 classes are made: the abstract superclass *Area* and 3 subclasses for the concrete areas: *Plain*, *Grassland* and *LakesRegion*. They all share common functions, namely: *ChangeHumidity()*, *Simulate()* and *IsAllTypeSame()*. Only the *ChangeWeather()* is made abstract because it needs to be implemented depending on the classes.

Similary, the *IWeather* super interface has 3 sub interfaces; *Rainy*, *Sunny* and *Cloudy* all of which implements all the *IWeather* functions. Here, singleton design pattern is used.

UML


Area
+ Owner : char
+ Type : char
+ WaterAmount: int
+ Area(owner:string, type:char, waterAmount:int)
+ ChangeWeather(weather:IWeather): Area
+ ChangeHumidity(humidity:ref int, area:Area)
+ Simulate(humidity: ref int, areas:List<Area>)
+ IsAllTypeSame(areas:List<Area>):bool
Owner=ownder
Type=type
WaterAmount=waterAmout
switch(area.Type)
P: humidity=humidity+5 break
G: humidity=humidity+10 break
L: humidity=humidity+15 break
char firstType=areas[0].Type
for int i=j, i<|areas|, i++ startloop
if areas[i].Type!=firstType
return false
endloop
for int i=0, j<|count|, i++ startloop
IWeather weather
if (humidity > 70)
weather = Rainy.Instance
humidity = 30
else if (humidity < 40)
weather = Sunny.Instance
else
double chanceOfRan = (humidity - 30) * 3.3
if (chanceOfRan > 70)
weather = Rainy.Instance
else
weather = Cloudy.Instance;
ChangeHumidity(ref humidity, areas[i])
areas[i]=areas[i].ChangeWeather(weaher)
endloop
return weather.ChangeP(this)
return weather.ChangeG(this)
return weather.ChangeL(this)
Plain
+ ChangeWeather
(weather:IWeather) : Area {override}
Grassland
+ ChangeWeather
(weather:IWeather) : Area {override}
LakesRegion
+ ChangeWeather
(weather:IWeather) : Area {override}
<<interface>>
IWeather
+ ChangeP (plain:Plain) : Area {virtual}
+ ChangeG (grassland:Grassland) : Area {virtual}
+ ChangeL (lakesRegion:LakesRegion) : Area {virtual}
Rainy
+ ChangeP (plain:Plainzone) : Area {override}
+ ChangeG (grassland:Grassland) : Area {override}
+ ChangeL (lakesRegion:LakesRegion) : Area {override}
Sunny
+ ChangeP (plain:Plainzone) : Area {override}
+ ChangeG (grassland:Grassland) : Area {override}
+ ChangeL (lakesRegion:LakesRegion) : Area {override}
Cloudy
+ ChangeP (plain:Plainzone) : Area {override}
+ ChangeG (grassland:Grassland) : Area {override}
+ ChangeL (lakesRegion:LakesRegion) : Area {override}

Specification

In the specification, it is required to calculate the n+1 versions the of the IWeather's weather variables as every areas part changes. The 0^{th} version is the initial weather variable. The changing of the areas is given by the function *ChangeWeather*(): area × weather^{m} → area × *weather*^{m}; giving the changed area (or a new area depending on the water level). The i^{th} version of the weather is denoted by the $IAtmosphere_i$, which the program won't display, it will simply be a temporal value of the variable IWeather.

$$\mathbf{A} = \mathrm{weather}: \mathrm{IWeather}^{m}, \mathrm{areas}: \mathrm{Area}^{n}, \mathrm{same}: \mathrm{records}^{*}, \mathrm{humidity}: \mathrm{int}$$

$$\mathrm{records}^{*} = \mathrm{rec}(\mathrm{Owner}: \mathrm{string}, \mathrm{Type}: \mathrm{char}, \mathrm{WaterAmount}: \mathrm{int})$$

$$\mathbf{Pre} = \mathrm{weather} = \mathrm{weather}_0 \wedge \ \mathrm{areas} = \mathrm{areas}_0 \wedge \mathrm{humidity} = \mathrm{humidity}_0$$

$$\mathbf{Post} = \mathrm{weather} = \mathrm{IWeather}_n$$

$$\wedge \forall i \in [1..n]: \mathrm{area}[i], \mathrm{atmosphere}_i = \mathrm{Simulate}(\mathrm{humidity}, \mathrm{area}_0[i])$$

$$\wedge \ \mathrm{same} = \bigoplus_{i=1+n_{(!\mathrm{Area.IsAllTypeSame(area)})}} < \mathrm{area}[i].\mathrm{Owner}, \mathrm{area}[i].\mathrm{Type}, \mathrm{area}[i].\mathrm{WaterAmount} >$$

Here, areas are concatenated after each iteration over the different weathers depending on the humidity level. The concatenation is done only if they are not of the same type. Below are the two summations and they can be merged as one loop (i = 1 … n)

Analogy:

enor(E)	i = 1 … n
f(e)	$\mathrm{Simulate}((humidity, \mathrm{area}_0[i])_1)$
s	humidity
H, +, 0	$\mathrm{IWeather}^{*}$, ⊜, weather

First component of the value of the function *Simulate()* function.

enor(E)	i = 1 … n
f(e)	$\mathrm{Simulate}((humidity, \mathrm{area}_0[i])_2)$
s	area
H, +, 0	Area^{*}, ⊜, area

Second component of the value of the function *Simulate()* function.

a ⊜ b ::= b

enor(E)	i = 1 … n
f(e)	$< \mathrm{area}[i] >$ if $!\mathrm{area}[i].\mathrm{IsAllTypeSame(area)}$

s	same
H, +, 0	Area*, $\oplus$, <>

The above are merged, to create one loop.

```
same:=< >
i=1..n
    weather,area[i]:=Simulate(humidity,area[i])
    !area[i].IsAllTypeSame(area)
    T                                                                   F
    same:=same⊕<area[i].Ownder,area[i].Type,area[i].WaterAmount>  |  skip
```

The i^{th} area is going to have *m+1* states as the areas changes depending on the weather which are in turn dependent on the humidity (which are affected by the changing areas from the previous ietration). The 0^{th} area is the given ($\text{area}_0[i]$). The m^{th} (i.e., the last state) is the situation when all areas are of same type..

The i^{th} area, before being changed because of the j^{th} weather is given by $\text{area}_{j-1}[i]$.Here, the j^{th} state is created by ChangeWeather( ) and the i^{th} state of the weather $\text{weather}_i[j]$.

Task to solve:

$$\forall j \in [1 \ldots m]: \text{area}[i], \text{weather}[j] = \text{ChangeWeather}(\text{weather}_i[j]) \wedge \text{area}[i] = \text{area}_m[i]$$

Below is only summation (concatenation) with enumerator j = 1 … m:

enor(E)	i = 1 … m
f(e)	ChangeWeather(weather[j])
s	area[j]
H, +, 0	Area*, $\ominus$, area[i]

first component of the value of function ChangeWeather ()

enor(E)	i = 1 … m
f(e)	ChangeWeather(weather[j])

s	weather[j]
H, +, 0	$IWeather^*$, $\oplus$, <>

Second component of the value of function ChangeWeather ()

Merging:

```
weather,area[i]:=Simulate(humidity,area[i])
j=1..m
    area[i],weather[j]:=ChangeWeather(weather[j])
```

Testing

Grey box test cases:

- Single Area Plain High Water Amount Low Humidity
- Single Area Plain High Water Amount Medium Humidity
- Single Area Plain High Water Amount Medium Humidity With Rainy Chance
- Single Area Plain High Water Amount High Humidity
- Single Area GrassLand Medium Water Amount Low Humidity
- Single Area GrassLand High Water Amount Medium Humidity
- Single Area GrassLand High Water Amount Medium Humidity With Rainy Change
- Single Area GrassLand High Water Amount High Humidity
- Single Area GrassLand High Water Amount High Humidity Change To Lakes Region
- Single Area GrassLand High Water Amount High Humidity Change To Plain
- Single Area LakesRegion Medium Water Amount Low Humidity
- Single Area LakesRegion High Water amount Medium Humidity
- Single Area LakesRegion High Water Amount High Humidity With Change Of Rain
- Single Area LakesRegion High Water Amount High Humidity
- Multiple Areas Combination
- Negative Water Amount