Sarun Kumar **Assignment 2 / Task 7** **30-04-2022**
IWCYYD
IWCYYD@INF.ELTE.HU

Task

Different kinds of plants live on a planet. If the nutrient of a plant runs out (its nutrient level becomes zero), the plant wastes away. There are three kinds of radiation on the planet: alpha, delta, no radiation. The different species of plants react to radiation differently. The reaction involves a change in the nutrient level of the plant and the radiation the next day. The radiation of the next day will be alpha radiation if the sum of the demand for alpha radiation over all plants is greater than the sum of the demand for delta radiation by at least three. If the demand for delta radiation is greater by at least three than the demand for alpha radiation, the radiation will be delta. If the difference is less than three, there will be no radiation. There is no radiation the first day.

Each plant has a name (string), a nutrient level (int), and a boolean that denotes whether it's alive. The plant species are wombleroot, wittentoot and woreroot. The different plant species react to the different radiations as follows. The level of nutrients changes first. After that, the plant can influence the radiation of the next day if it's still alive.

Wombleroot: Alpha radiation makes the nutrient level increase by 2, no radiation makes it decrease by 1, and delta radiation makes it decrease by 2. It demands alpha radiation by a strength of 10 regardless of the current radiation. This plant also wastes away if its nutrient level increases above 10.

Wittentoot: Alpha radiation makes the nutrient level decrease by 3, no radiation makes it decrease by 1, delta radiation makes it increase by 4. This plant demands delta radiation with strength 4 if its nutrient level is less than 5, with strength 1 if its nutrient level is between 5 and 10, and doesn't influence the radiation if its nutrient level is greater than 10.

Woreroot: Its nutrient level increases by 1 if there is alpha or delta radiation, and decreases by 1 if there is no radiation. Doesn't influence the radiation of the next day.

Simulate the ecosystem of plants and give the name of the strongest plant which is still alive after n days. Print all the data of the plants and the level of radiation on each day.

The program should read the data of the simulation from a text file. The first line contains the number of plants. Each of the next lines contains the data of one plant: its name, its species, and its starting nutrient level. The species can be: wom - wombleroot, wit - wittentoot, wor - woreroot. The last line of the file contains n, the number of days as an int. The program should ask for the filename and display the contents of the file. You can assume that the input file is correct. A possible input file:

```
  4
 Hungry  wom 7
Lanky wit 5
Big wor 7
Tall wit 3
10
```

Analysis[1]

Independent objects in the task are the plants. They can be divided into 3 different groups: Wombleroot ,Wittentoot and Woreroot.

All of them have a name and a nutrient level that can be get . It can be examined what happens when they face a type of radiation. Radiation effects on plants in the following way:

Wombleroot:

Radiation	Nutrient level change	Radiation demand (if alive)
Alpha	+ 2	10
Delta	-2	10
No Radiation	-1	10

Wittentoot:

Radiation	Nutrient level change	Radiation demand(if alive)
Alpha	-3	4(level<5) or 1(10>=level>=5)
Delta	+4	4(level<5) or 1(10>=level>=5)
No Radiation	-1	4(level<5) or 1(10>=level>=5)

Woreroot :

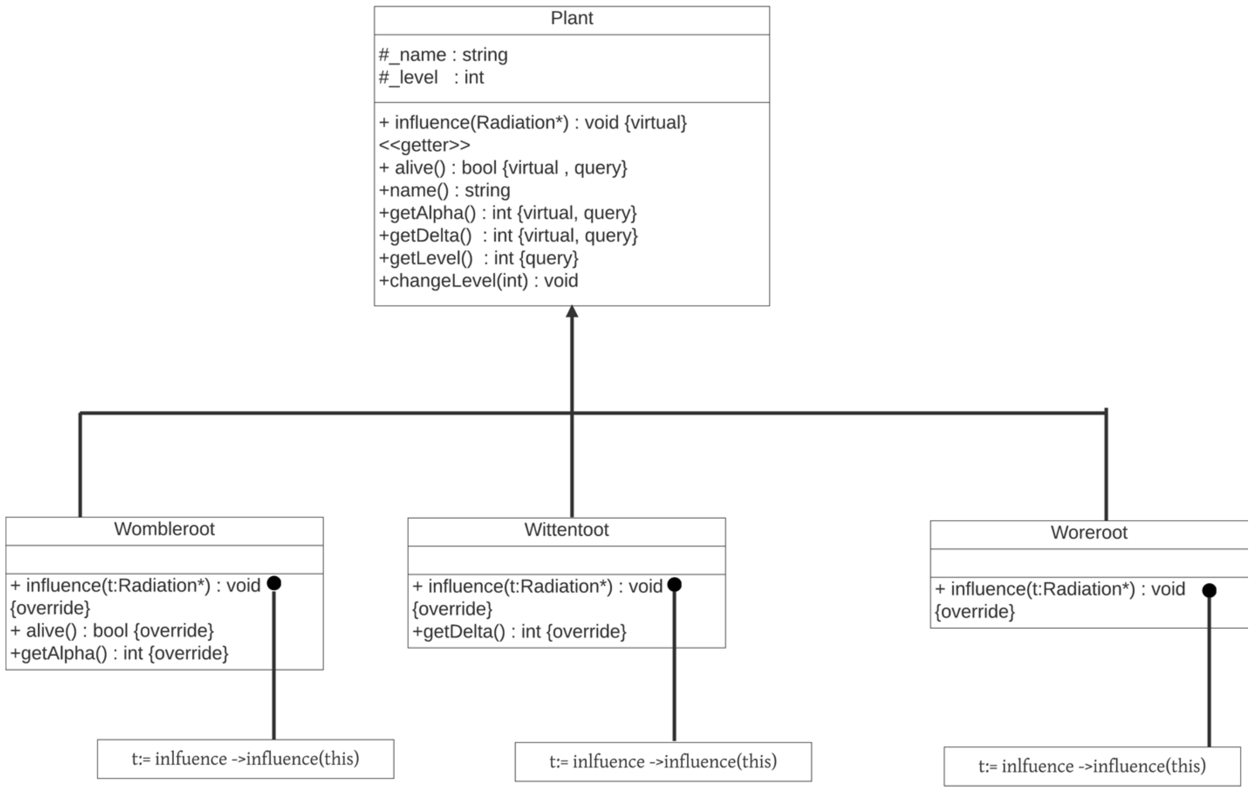
Radiation	Nutrient level change	Radiation demand(if alive)
Alpha	+1	0
Delta	+1	0
No Radiation	-1	0

Plan2

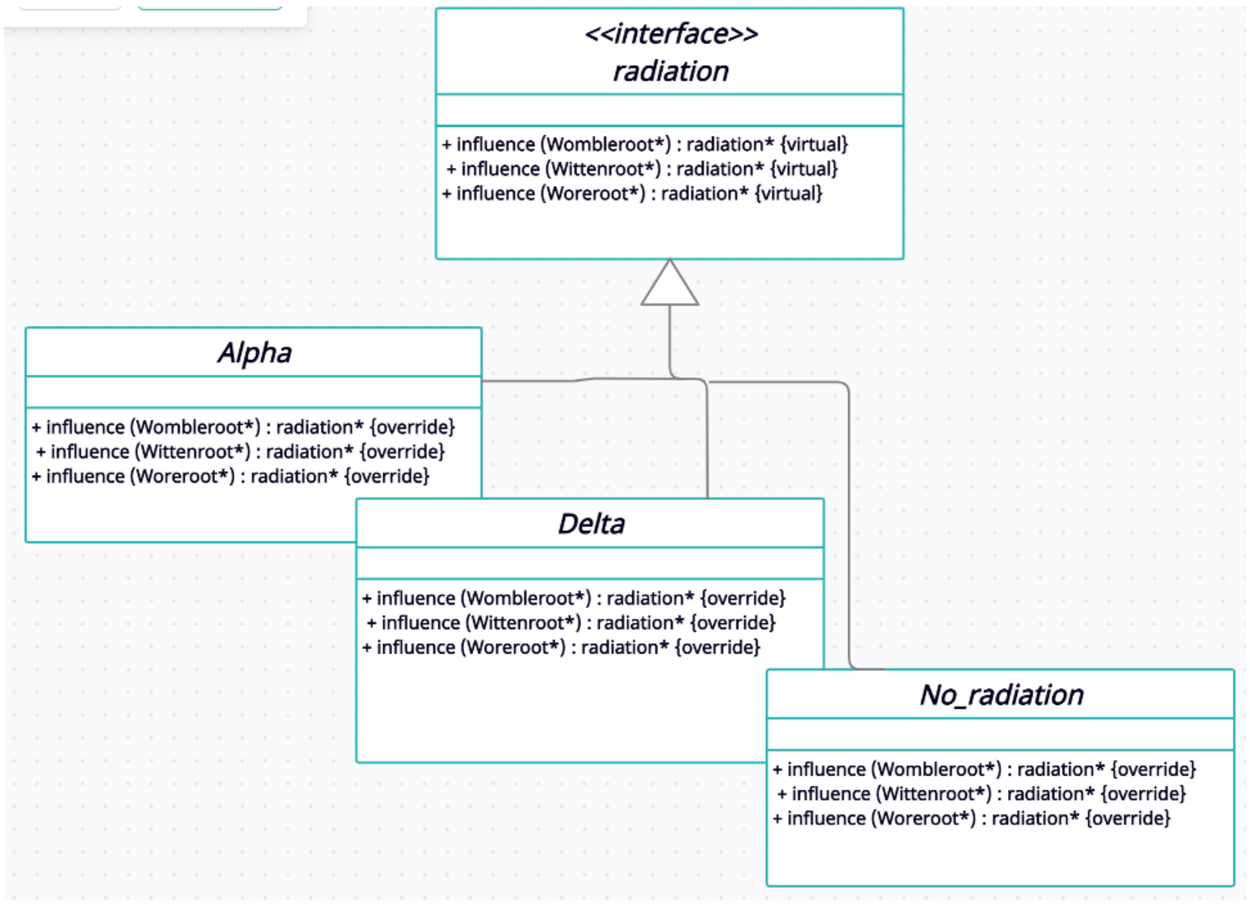
To describe the Plants, 4 classes are introduced: base class *Plant* to describe the general properties and 3 children for the concrete species: *Wombleroot*, *Wittentoot*, and *Woreroot*. Regardless the type of the plants, they have several common properties, like the name (*_name*) and the power (*_level*), the getter of its name (*name()*), if it is alive (*alive()*) and it can be examined what happens when it faces a radiation. This latter operation (*influence()*) modifies the nutrient level of the plant and may influence the radiation next day. Operations *alive()* and *name()* may be implemented in the base class already, but *influence()* just on the level of the concrete classes as its effect depends on the species of the plant. Therefore, the general class *Plant* is going to be abstract, as method *influence()* is abstract and we do not wish to instantiate such class.

General description of the radiations is done the base class *Radiation* from which concrete radiations are inherited: Alpha, *Delta*, and *No_radiation*. Every concrete radiation has three methods that show how a Wombleroot, a Wittentoot or a Woreroot changes during radiation and how the nutrient level changes, too. Objects are referred by pointers.

The special plants classes initialize the name and the power through the constructor of the base class and override the operation *influence()* in a unique way. Initialization and the override are explained in Section Analysis. According to the tables, in method *influence()*, conditionals have to be used in which the type of the radiation is examined. Though, the conditionals are not effective if the program might be extended by new radiation types, as all of the methods *influence()* in all of the concrete plants classes have to be modified. To avoid it, design pattern Visitor is applied where the Plants classes are going to have the role of the visitor.


Plant
#_name : string
#_level   : int
+ influence(Radiation*) : void {virtual}
<<getter>>
+ alive() : bool {virtual , query}
+name() : string
+getAlpha() : int {virtual, query}
+getDelta() : int {virtual, query}
+getLevel() : int {query}
+changeLevel(int) : void
Wombleroot
+ influence(t:Radiation*) : void {override}
+ alive() : bool {override}
+getAlpha() : int {override}
t:= inlfuence ->influence(this)
Wittentoot
+ influence(t:Radiation*) : void {override}
+getDelta() : int {override}
t:= inlfuence ->influence(this)
Woreroot
+ influence(t:Radiation*) : void {override}
t:= inlfuence ->influence(this)

Methods *influence()* of the concrete plants expect a radiation object as an input parameter as a visitor and calls the methods which corresponds to the species of the plant.



<<interface>>
radiation
+ influence (Wombleroot*) : radiation* {virtual}
+ influence (Wittenroot*) : radiation* {virtual}
+ influence (Woreroot*) : radiation* {virtual}
Alpha
+ influence (Wombleroot*) : radiation* {override}
+ influence (Wittenroot*) : radiation* {override}
+ influence (Woreroot*) : radiation* {override}
Delta
+ influence (Wombleroot*) : radiation* {override}
+ influence (Wittenroot*) : radiation* {override}
+ influence (Woreroot*) : radiation* {override}
No_radiation
+ influence (Wombleroot*) : radiation* {override}
+ influence (Wittenroot*) : radiation* {override}
+ influence (Woreroot*) : radiation* {override}

All the classes of the radiations 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 radiation as every plants get influenced by it. The 0th version is the initial radiation that is No_radaition. The facing of one radiation is denoted by function *influence* : $Radiation \times Plant^m \rightarrow Radiation \times Plant^m$ which returns the same radiation(but providing the demand based on change in nutrient level) and changes the nutrient level. i^{th} type of the Plant is denoted by $Plant_i$, which the program is not going to show, it is going to be just a temporal value of variable *Plant*.

A = *plants*: $Plant^m$, *radiations*: $Radiation^n$, *alivePlants*: *int**,
strongestPlant : int, strongestPlantName : string

Pre = $plants = plants_0 \wedge radiations = radiations_0$

Post = $\forall i \in [1..n]: plants[i], radiations_i = createAndComputeData(plants_0[i], radiation_{i-1}) \wedge$

$$alivePlants = \bigoplus_{\substack{i=1..n \\ plants[i].alive()}} < i > \wedge \text{ strongestPlant} = \underset{\substack{k \in alivePlants[j] \\ j \in 1..|alivePlants|}}{\text{MAX}} \text{plants[k]->getLevel()}$$

$\wedge$ strongestPlantName = plants[strongestPlant]

Concatenation of the plants(after facing the radiation for n days) and influencing the demand for radiation step by step are two Summations just as the assortment of the alive plants. As all of them are based on the same enumerator, they can be merged into the same loop (*i=1 .. n and m is the number of days*).

Analogy:

enor(E)	$i = 1 .. n$
$f(e)$	$createAndComputeData\ (plants\ [i], radiation)_1$
s	*plants*
H, +, 0	$Plant^*$, $\oplus$, <>

first component of the value of function *createAndComputeData()*

enor(E)	$i = 1 .. m$
$f(e)$	$createAndComputeData\ (plants\ [i], radiation)_2$
s	*radiation*
H, +, 0	$Plant^*$, $\ominus$, *radiation*

second component of the value of function *createAndComputeData()*

$a \ominus b ::= b$

enor(E)	$i = 1 .. n$
$f(e)$	$< i >$ if $plants[i].alive()$
s	*alivePlants*
H, +, 0	$Plant^*$, $\oplus$, <>

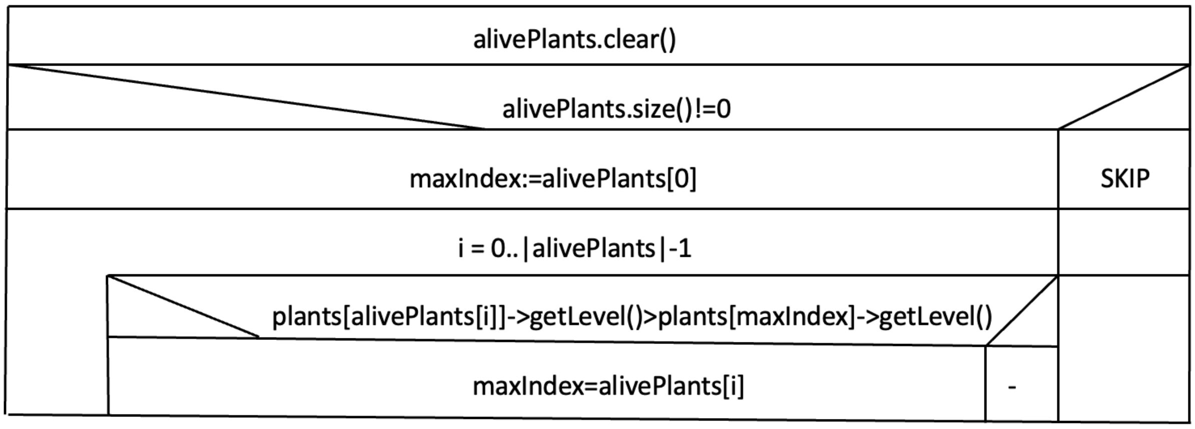
To collect radiations(after reading input file and collecting plants), we use way depicted below:

```
radiations.resize(days)
if Days>0
    radiations[0]:= No_Rad::instance()
else
    SKIP
j = 0..days-1
    alphaDemSum,deltaDemSum := 0, 0
    i = 0..|plants|
        if plants[i]->alive()
            alphaDemSum = alphaDemSum + plants[i]->getAlpha()
            deltaDemSum = deltaDemSum + plants[i]->getDelta()
            plants[i]->influence(radiations[j])
        else
            SKIP
    j+1<days
        alphaDemSum>=(3+deltaDemSum)
            radiations[j+1]:=Alpha_Rad::instance()
        deltaDemSum>=(3+alphaDemSum)
            radiations[j+1]:=Delta_Rad::instance()
        else
            radiations[j+1]:=No_Rad::instance()
```

By continuing with procedure given below, the solution can be achieved:

```
alivePlants := <>
i =1 .. n
    if plants[i].alive()
        alivePlants := alivePlants ⊕ i
    else
        SKIP
```

To get the final result, we can use result() function shown below,



```
alivePlants.clear()
if alivePlants.size()!=0
    maxIndex:=alivePlants[0]
    i = 0..|alivePlants|-1
        if plants[alivePlants[i]]->getLevel()>plants[maxIndex]->getLevel()
            maxIndex=alivePlants[i]
        else
            -
else
    SKIP
```

Testing

Grey box test cases:

Outer loop (Summation)

1. length-based:
   - zero plant
   - one plant
   - more plants
2. first and last:
   - first plants survives or not after facing the radiation for n days
   - last plants survives or not after facing the radiation for n days

Inner loop (Summation)

1. length-based:
   - one plant under radiation for zero-day
   - one plant under radiation for one day
   - one plant under radiation for more than one day
   - more plant under radiation for more than one day
2. first and last:
   - first radiation of getting influenced properly depending on the demand of radiation by the species of the plant
   - last radiation of getting influenced properly depending on the demand of radiation by the species of the plant
   -

Examination of function influence()

Different cases checking each radiation of n days.

Some cases for checking if the plant is alive.