OOP Khongorzul Khenchbish: Documentation, 3rd assignment 1.

Khongorzul Khenchbish 4th May 2021
R8HP78
r8hp78@inf.elte.hu
Group 1

Task

We are simulating the animals of the tundra. There are colonies of prey and predator animals. The number of animals in a colony affect the number of animals in other colonies. There are three predator species: the snowy owl, the arctic fox and the wolf. There are three kinds of prey: the lemming, the arctic hare and the gopher.

If the number of prey animals increase, predators can reproduce more quickly. If the number of prey is very large, most of them will wander away because they cannot find enough food. If the number of predators is large, the number of the prey decreases quicker as they are preyed upon.

Each colony has a name, a species, and the number of animals in the colony. The prey species are affected by the different predator species as follows. The number of animals in their own colony changes first, then they influence the predators.

Lemming**:** If they are preyed upon by a predator colony, the number of animals in their colony decreases by four times the number of animals in the predator colony. The number of animals in their colony doubles every second turn. If there are more than 200 animals in the colony, the number of animals in the colony decreases to 30.

Hare**:** If they are preyed upon by a predator colony, the number of animals in their colony decreases by double the number of animals in the predator colony. The number of animals in their colony grows by 50 percent (to one and a half times their previous number) every second turn. If there are more than 100 animals in the colony, the number of animals in the colony decreases to 20.

Gopher**:** If they are preyed upon by a predator colony, the number of animals in their colony decreases by double the number of animals in the predator colony. The number of animals in their colony doubles every fourth turn. If there are more than 200 animals in the colony, the number of animals in the colony decreases to 40.

Predators choose and attack a prey colony randomly in each turn. If there are not enough animals in the attacked colony (for example, there are not four times the number of predators in a lemming colony), the number of predators also decreases: every fourth predator out of the ones who didn't get prey perishes. Predators have offsprings every eighth turn. Normally, the snow owls have 1 offspring per 4 animals, the foxes have 3 offsprings per 4 animals, and the wolves have 2 offsprings per 4 animals.

The program should read the colonies from a text file. The first line contains the number of prey and predator colonies separated by a space. Each of the next lines contains the data of one colony separated by space: their name, their species, their starting number of animals. The species can be: o - owl, f - fox, w - wolf, l - lemming, h - hare, g - gopher.

Simulate the process until the number of animals in each predator colony decreases to below 4, or the number of predators doubles compared to its starting value. Print the data of each colony in each turn.

The program should ask for the name of the input file and display its contents. You can assume that the input file is correct. A possible input:

```
3 3
lem1 l 86
lem2 l 90
hare1 h 26
go g 12
hungry w 12
feathery o 6
```

Analysis

Independent objects in the task are the prey and predators colonies. There are three predator species: ***the snowy owl, the arctic fox and the wolf***. There are three kinds of prey: ***the lemming, the arctic hare and the gopher***.

All of them have a name, number of animals, how many times hunted. It can be examined what happens any predator hunts the prey. The hunting effects the preys and the predators in the following way:

Prey: Lemming

conditions	number of preys(n)
every 2nd turn	x 2
n exceed 200 (n > 200)	- 30
each attack	- 4 x (number of predators)

Prey: Hare

conditions	number of preys(n)
every 2nd turn	x 1.5
n exceed 100 (n > 100)	- 20
each attack	- 2 x (number of predators)

Prey:Gopher

conditions	number of preys(n)
every 4th turn	x 2
n exceed 200 (n > 200)	- 40
each attack	- 2 x (number of predators)

Predator. Snowy owl

conditions	number of predators(m)
offspring on every 8th turn	+ (m/4)
n < (4 * m)	- (m/4)

Predator. Arctic fox

conditions	number of predators(m)

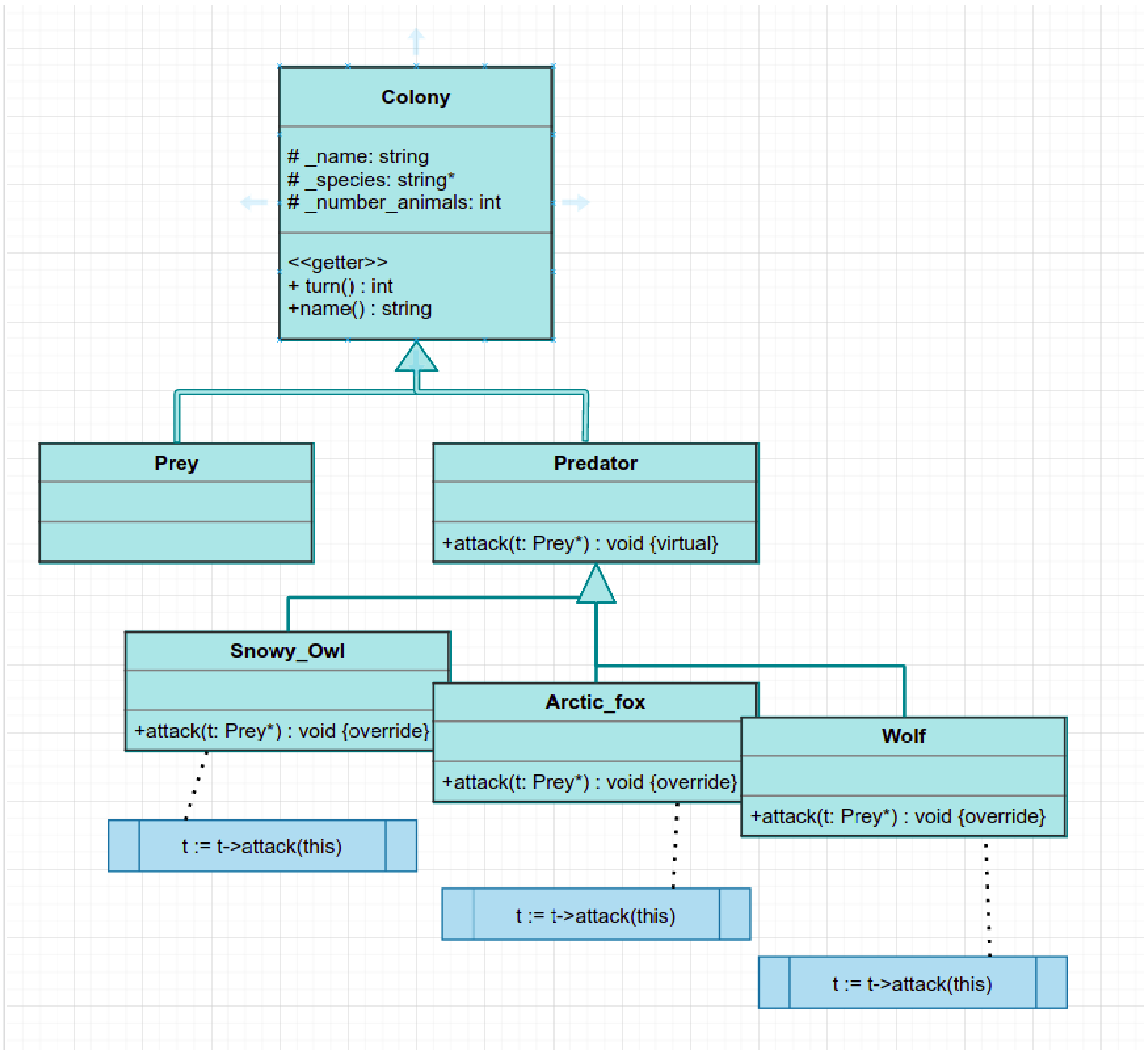
offspring on every 8th turn	+ (3 x m/4)
n < (2 * m)	- (m/4)

Predator. Wolf

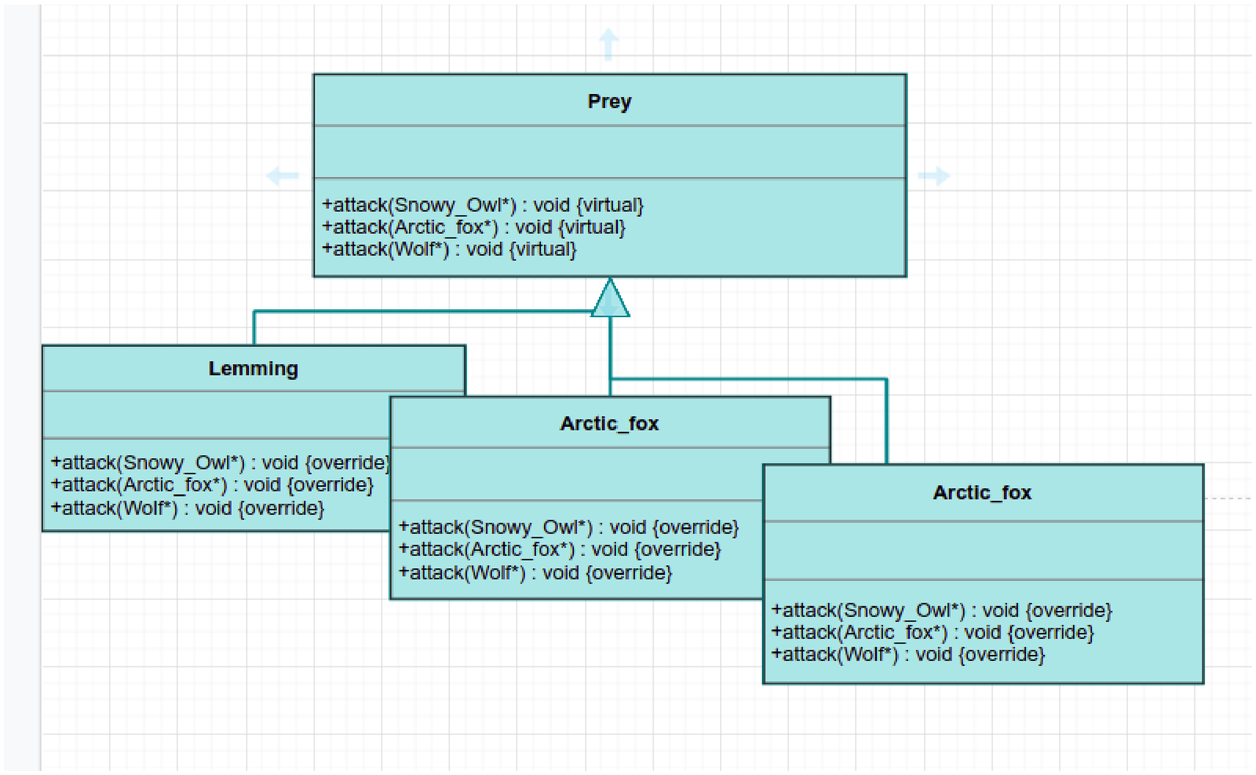
conditions	**number of predators(m)**
offspring on every 8th turn	+ (2 x m/4)
n < (2 * m)	- (m/4)

Plan

To describe the creatures, 9 classes are introduced: base class *Colony* to describe the general properties and 2 children for the types of colony: Prey and Predator. Regardless of the types of colonies the species have several common properties. The name (_name) and the power (_power), the getter of its name (name()), getter for number of turn (turn()) and it can be examined what happens when predator species attack prey species. This latter operation (attack()) modifies the number of alive animals and the turn of the prey, predator colony. Operations turn() and name() may be implemented in the base class already, but attack() just on the level of the concrete classes as its effect depends on the species of the preys and predators. Therefore, the general class Colony is going to be abstract, as method attack() is abstract and we do not wish to instantiate such class. The special colony classes initialize the name and the power through the constructor of the base class and override the operation attack() in a unique way. According to the tables, in method attack(), conditionals have to be used in which the current number of animal is. Though, the conditionals are not effective if the program might be extended by new conditionals, as all of the methods transmute() in all of the concrete colony classes have to be modified. To avoid it, design pattern Visitor is applied where the ground classes are going to have the role of the visitor.



Methods attack() of the concrete predator colony expect a prey object as an input parameter as a visitor and calls the methods which corresponds to the species of the predator.


Prey
+attack(Snowy_Owl*) : void {virtual}
+attack(Arctic_fox*) : void {virtual}
+attack(Wolf*) : void {virtual}
Lemming
+attack(Snowy_Owl*) : void {override}
+attack(Arctic_fox*) : void {override}
+attack(Wolf*) : void {override}
Arctic_fox
+attack(Snowy_Owl*) : void {override}
+attack(Arctic_fox*) : void {override}
+attack(Wolf*) : void {override}
Arctic_fox
+attack(Snowy_Owl*) : void {override}
+attack(Arctic_fox*) : void {override}
+attack(Wolf*) : void {override}

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

A = $preys: Prey^{m},\ predators: Predator^{n},\ l: bool$

Pre = $preys\ =\ preys_{0}\ \wedge\ predators\ =\ predators_{0}$

Post =

$\forall i\ \in\ [1..n],\ \forall randIndex\ \in\ [1..m]: predator[i]\ -\!\!>\ attack(preys[randIndex])\ \wedge\ l\ =\ \wedge^{n}_{i=1}\ cond$

where:

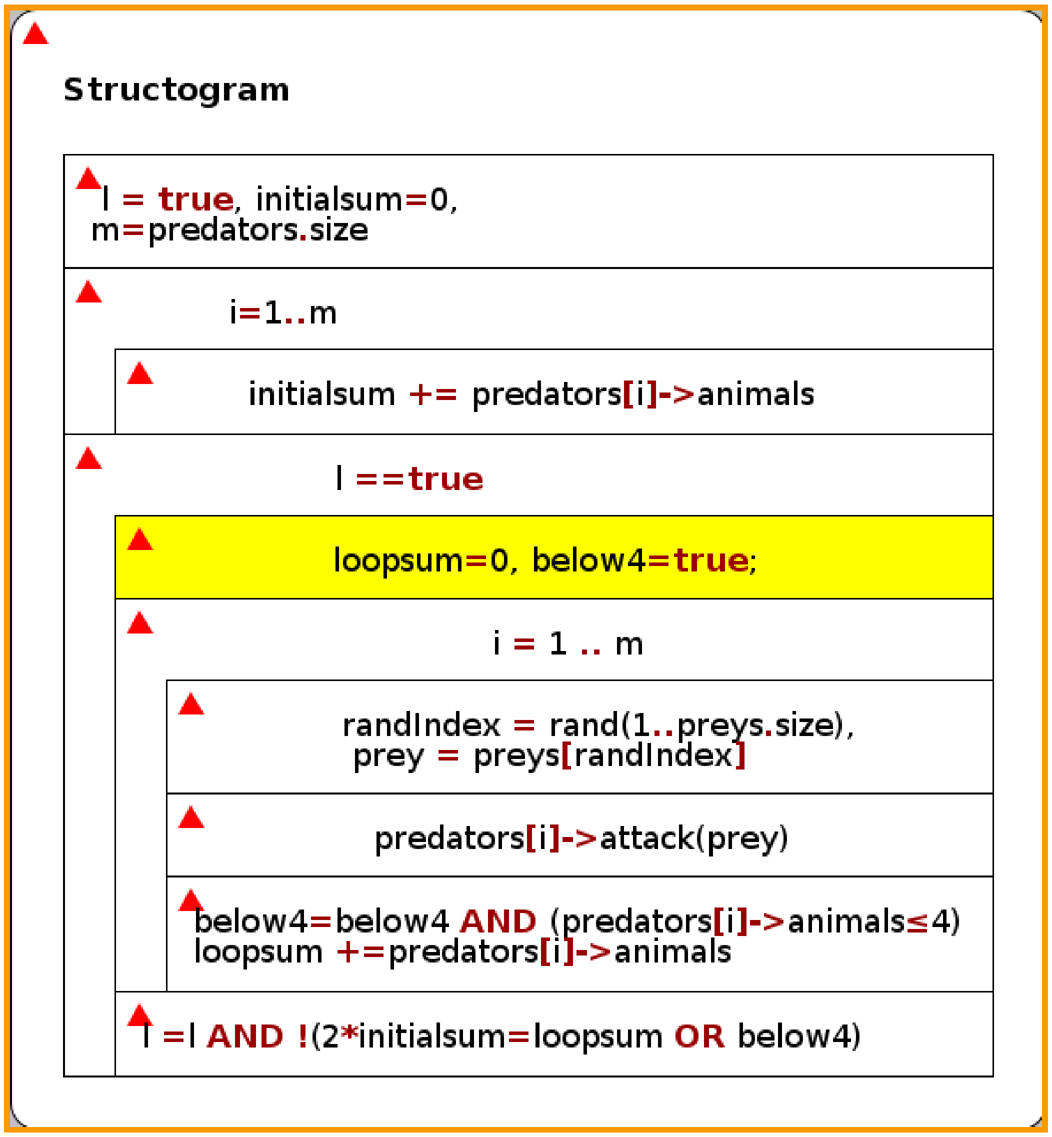
$$cond\ =\ !\,(predator[i].animal\ =\ (2\ *\ \sum_{i=1}^{n} predators[i])\ \vee\ below4)$$

$$below4\ =\ \wedge^{n}_{i=1} predators[i].animals\ <\ 4$$

Analogy

enor(E)	i = 1 .. n
f(e)	attack(predators[i], prey[randIndex])

s	predators
H, and, 0	Predators*, ∧, <>



Structogram

```
l = true, initialsum=0,
m=predators.size
i=1..m
    initialsum += predators[i]->animals
l ==true
    loopsum=0, below4=true;
    i = 1 .. m
        randIndex = rand(1..preys.size),
        prey = preys[randIndex]
        predators[i]->attack(prey)
        below4=below4 AND (predators[i]->animals≤4)
        loopsum +=predators[i]->animals
    l =l AND !(2*initialsum=loopsum OR below4)
```

Testing

Grey box test cases:

Main Loop (Linear search)

Length-based:

1. file without predator
2. one predator
3. more predator

First and Last:

1. first predator attacked, if satisfies stopping condition
2. last predator attacked, if satisfies stopping condition

Examination of function attack()
Nine different cases depending on the random chosen prey.

1. prey offspring
2. predator offspring
3. attack function