Documentation of OOP
2nd Assignment
Task 6:
Simulation of Tundra Animals

Prepared By: Ammar Yildirim
Neptun Code: FA27QM
Gmail : fa27qm@inf.elte.hu
Group 4

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 each of the prey colonies becomes extinct or the number of prey animals quadruples compared to its starting value. Print the data of each colony in each turn.

Analysis

Prey Reproduction:

- Lemming: they double their previous size in every 2nd turn.
- Hare: they increase their size by one and a half every 2nd turn.
- Gopher: they double their previous size in every 4th turn.

When preys are being eaten:

- Lemming: decrease by 4 times the size of the predators.
- Hare: decrease by 2 times the size of the predators.
- Gopher: decrease by 2 times the size of the predators.

Restrictions:

- Lemming: if there are more than 200, their number decreases to 30.
- Hare: if there are more than 100, their number decreases to 20.
- Gopher: if there are more than 200, their number decreases to 40.

Predators Perishing:

- If there are not enough preys in the colony they attack, their size is divided by the ratio the preys' size decreases when they are attacked.
    - For example, normally if a predator of any kind attacks lemming, lemmings size decreases by 4 times the size of the predator. However, if there are not (4*size of predator) lemmings, then the predators size is divided by 4.
    - The ratio 4 is 2 for Hare and Gopher.

Predator Reproduction:

- All predators reproduce every 8th turn.
- Owl: 1 offspring per 4 animals.
- Fox: 3 offspring per 4 animals.
- Wolf: 2 offspring per 4 animals.

Plan

In this task there are two types of animals, preys and predators. We will present both with a separate class. Regardless if it is a prey or a predator colony, a colony has the ability to reproduce, as well as it has a size and a name property. Hence, we will create an abstract *Colony* class, which will provide the properties and the skeleton for the *Reproduce* function. On broad terms, the *Reproduce* function works as follows: If the right time has come, then we calculate the new size. However, the right time and the calculation method changes for each colony, so we define them in the child classes.

A **prey** colony has 3 duties: reproduction, restriction for maximum size, and being hunted by a predator.

Prey colonies have a common calculation method *(CalculateReproduction())* for reproduction in which we multiply the size by a coefficient*(PreyReproductionRatio()),* but the coefficient is different for every prey colony. So, we provide a skeleton for *CalculateReproduction()* in the prey class, and will override *PreyReproductionRatio()* if necessary, in the child classes.

For the restriction*(CheckNumber()),* it is said that when a prey colony reaches a certain size*(MaxSize()),* its size should be set to a new lower size*(NewSize()).* Again, we use template design pattern and provide the general algorithm in *CheckNumber(),* we also give some return values for *MaxSize()* and *NewSize(),* which generally hold true but will be overridden for certain classes.

Finally, when a prey colony is hunted by a predator colony *(HuntedBy()),* the predator has to be passed as a parameter. The prey's size is decreased by a certain coefficient*(Multiplier()),* and if there are not enough preys for predators, the predator's size is divided by the same coefficient. Like the previous functions, *HuntedBy()* is defined generally, and *Multiplier()* is given a value which holds true in most cases but will be overridden for certain classes.

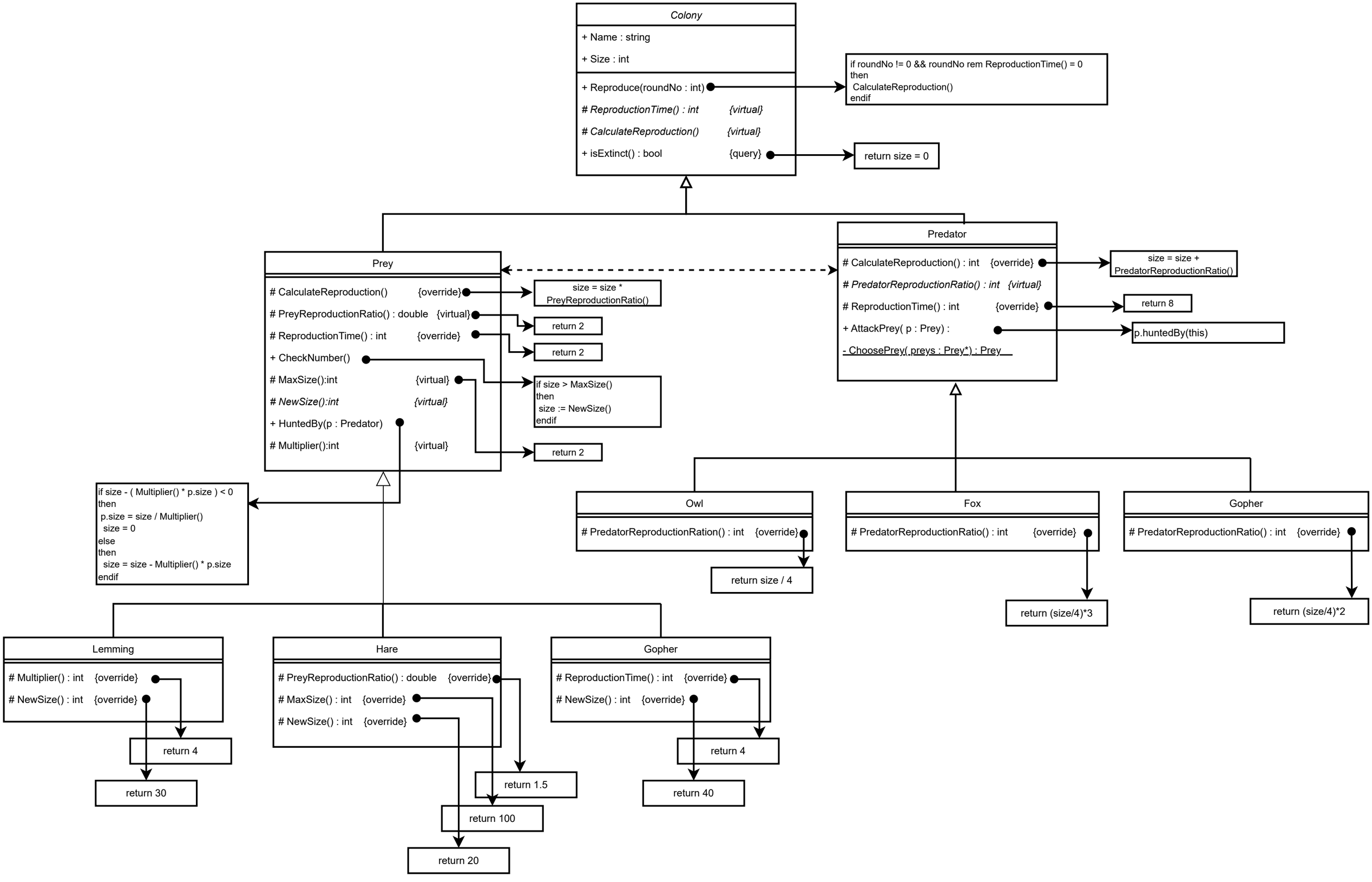
We need 3 child classes for the prey class: Lemming, Hare, and Gopher. They initialize their names and sizes using the constructor of the base class. Thanks to the template design pattern, we only override the return values of a few methods inherited from the parent class.

A **predator** colony has 2 duties: reproduction and hunting a prey.

Predator colonies have a common calculation method *(CalculateReproduction())* for reproduction in which we add to the size a certain ratio*(PredatorReproductionRatio()),* but the ratio is different for every predator colony. So, we provide a skeleton for *CalculateReproduction(),* and create an abstract PredatorReproductionRatio(), which will be overridden for the child classes.

Hunting a prey consists of two steps: Randomly choosing a prey and Attacking a prey. When a predator colony attacks a prey colony*(AttackPrey()),* it simply passes itself as a parameter to the *HuntedBy()* method of the prey.

We need 3 child classes for the predator class: Owl, Fox, Wolf. They initialize their names and sizes using the constructor of the base class. Thanks to the template design pattern, we only override the *PredatorReproductionRatio()* for each of them and we are done.


Colony
+ Name : string
+ Size : int
+ Reproduce(roundNo : int)
# ReproductionTime() : int {virtual}
# CalculateReproduction() {virtual}
+ isExtinct() : bool {query}
if roundNo != 0 && roundNo rem ReproductionTime() = 0
then
CalculateReproduction()
endif
return size = 0
Prey
CalculateReproduction() {override}
PreyReproductionRatio() : double {virtual}
ReproductionTime() : int {override}
+ CheckNumber()
# MaxSize():int {virtual}
# NewSize():int {virtual}
+ HuntedBy(p : Predator)
# Multiplier():int {virtual}
size = size * PreyReproductionRatio()
return 2
return 2
if size > MaxSize()
then
size := NewSize()
endif
return 2
if size - (Multiplier() * p.size) < 0
then
p.size = size / Multiplier()
size = 0
else
then
size = size - Multiplier() * p.size
endif
Predator
CalculateReproduction() : int {override}
PredatorReproductionRatio() : int {virtual}
ReproductionTime() : int {override}
+ AttackPrey(p : Prey) :
- ChoosePrey(preys : Prey*) : Prey
size = size + PredatorReproductionRatio()
return 8
p.huntedBy(this)
Owl
PredatorReproductionRation() : int {override}
return size / 4
Fox
PredatorReproductionRatio() : int {override}
return (size/4)*3
Gopher
PredatorReproductionRatio() : int {override}
return (size/4)*2
Lemming
Multiplier() : int {override}
NewSize() : int {override}
return 4
return 30
Hare
PreyReproductionRatio() : double {override}
MaxSize() : int {override}
NewSize() : int {override}
return 1.5
return 100
return 20
Gopher
ReproductionTime() : int {override}
NewSize() : int {override}
return 4
return 40

Specification

A = (preys : Prey*, predators: Predator*, finished: Prey*)

Pre = (preys = preys' $\wedge$ predators = predators')

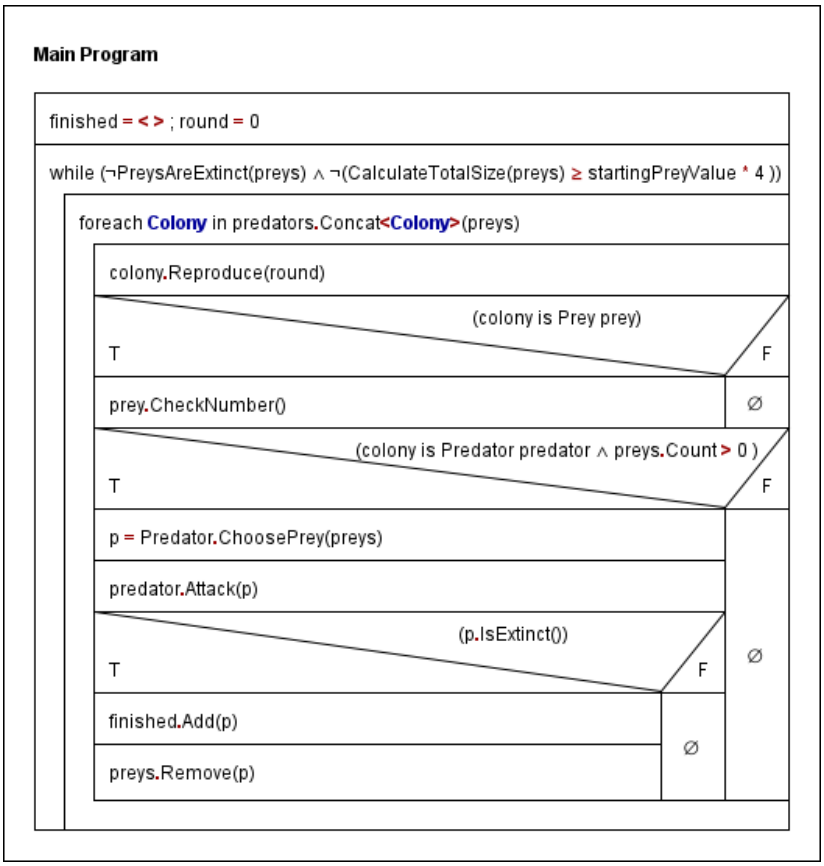
Post = ($\forall$i$\in$[1..n]: ($\forall$ j $\in$ [1..m]: predators[i].Attack(preys[j]) $\wedge$

finished = $\bigoplus_{\substack{e \in \text{preys} \\ e.isExtinct()}}$ <e>)

Analogy

Enor(E)	preys
F(e)	<e>
Cond(e)	e.isExtinct()
S	Finished
H, +, 0	Prey*, $\bigoplus$, <>

Algorithm



Testing

1) **Checking Reproduction**
    - Check for each predator and prey colony if they increase by the right margin at the right round
    - Check that they do not increase in size if the wrong round is passed
2) **Checking Restriction**
    - Check for each prey that if they have exceeded the threshold, their number is decreased
    - Check that the function does not modify the size of the prey colony if they have not exceeded the threshold
3) **Check Attacking**
    - Check for each predator what happens when they attack a prey colony. Two cases should be checked:
        1. If there are enough preys, then only the size of the prey is modified
        2. If there are not enough preys, both the size of prey and predator is modified