Murad Huseynov **2nd assignment/9. Task** 3rd June 2023
RX15MW
rx15mw@inf.elte.hu
Group 10

Task

Hobby animals need several things to preserve their exhilaration. Cathy has some hobby animals: fishes, birds, and dogs. Every animal has a name and their exhilaration level is between 0 and 100 (0 means that the animal dies). If their keeper is in a good mood, she takes care of everything to cheer up her animals, and their exhilaration level increases: of the fishes by 1, of the birds by 2, and of the dogs by 3.

On an ordinary day, Cathy takes care of only the dogs (their exhilaration level does not change), so the exhilaration level of the rest decreases: of the fishes by 3, of the birds by 1. On a bad day, every animal becomes a bit sadder and their exhilaration level decreases: of the fishes by 5, of the birds by 3, and of the dogs by 10.

Cathy's mood improves by one if the exhilaration level of every alive animal is at least 5.

Every data is stored in a text file. The first line contains the number of animals. Each of the following lines contain the data of one animal: one character for the type (F – Fish, B – Bird, D-Dog), name of the animal (one word), and the initial level of exhilaration.

In the last line, the daily moods of Cathy are enumerated by a list of characters (g – good, o – ordinary, b – bad). The file is assumed to be correct.

Analysis

Independent objects in the task are the animals. They can be divided into 3 different groups: fishes, birds, and dogs.

All of them have a name and an exhilaration level. It can be examined what happens when their keeper's mood changes for a specific day. Keeper's mood affects the animal's exhilaration in the following way:

Fish:

Mood	Exhilaration change
good	+1
ordinary	-3
bad	-5

Bird:

Mood	Exhilaration change
good	+2
ordinary	-1
bad	-3

Dog:

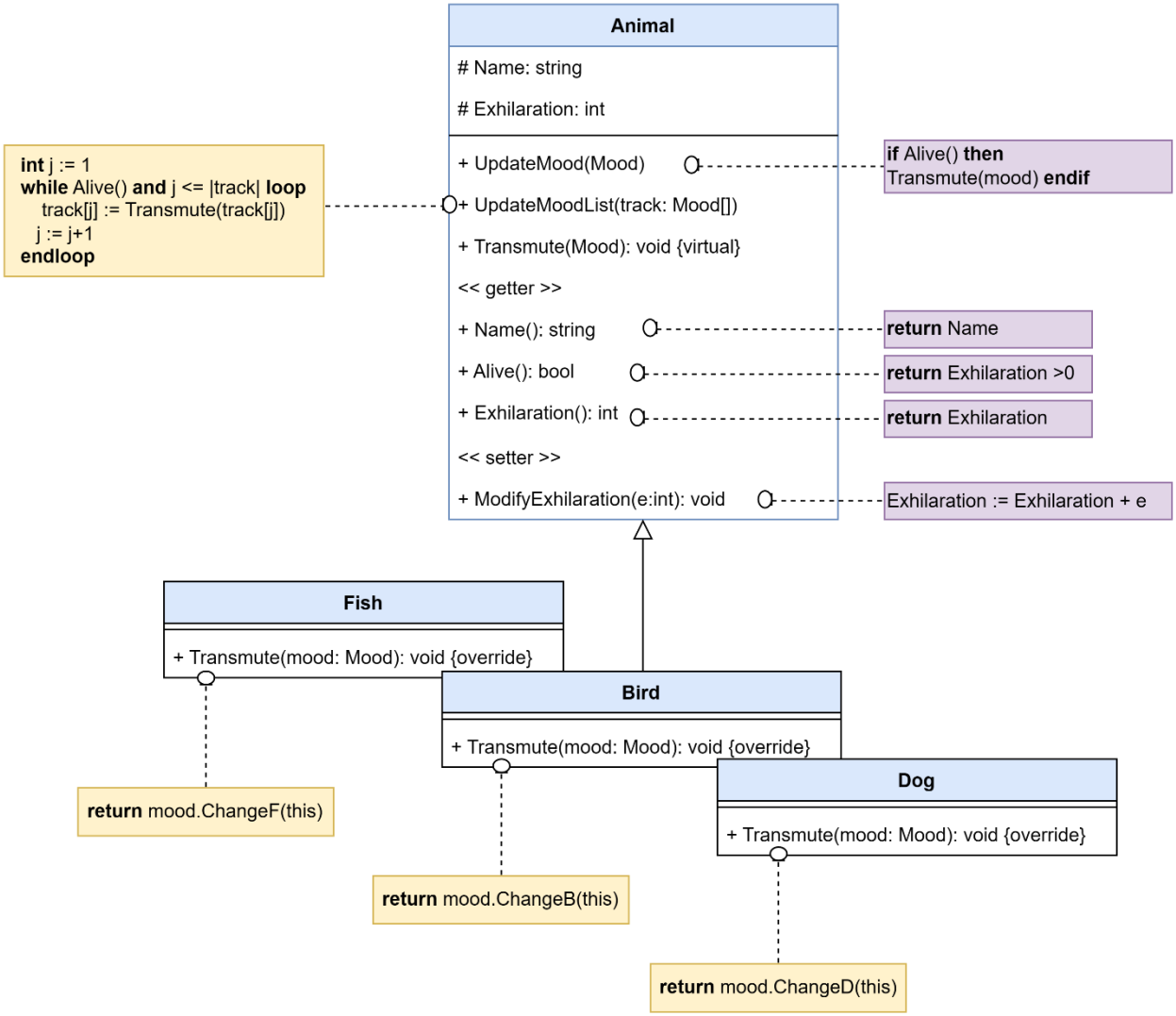
Mood	Exhilaration change
good	+3
ordinary	0
bad	-10

Plan

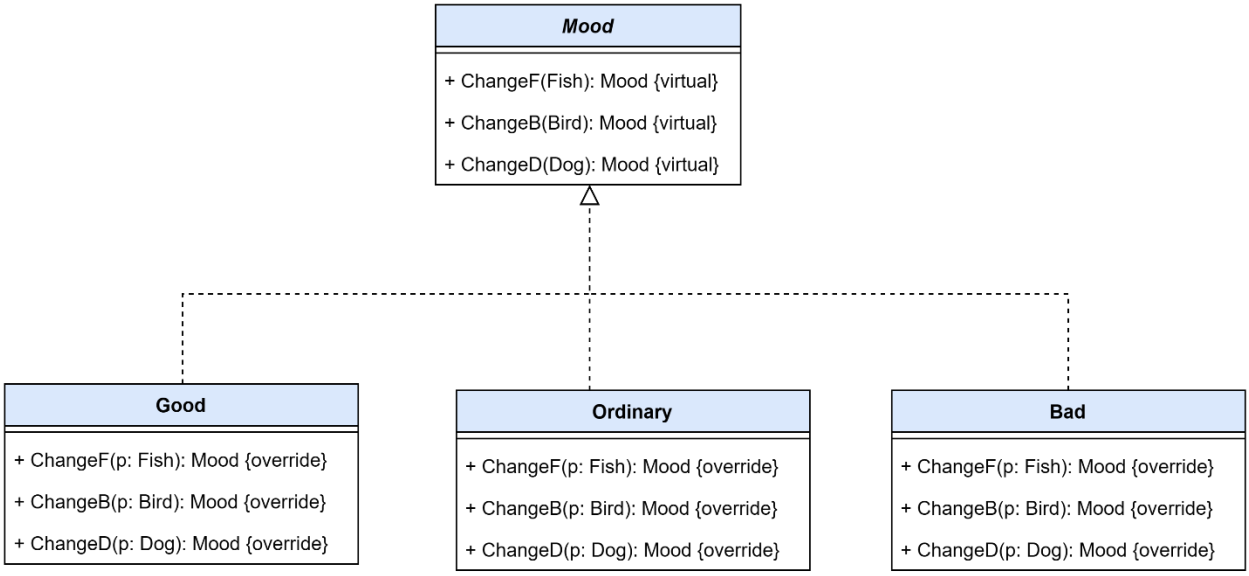
To describe the animals, 4 classes are introduced: base class *Animal* to describe the general properties and 3 children for the concrete species: *Fish*, *Bird*, and *Dog*. Regardless the type of the animals, they have several common properties, like the name (*Name*) and the power (*Exhilaration*), the getter of its name, the getter and protected setter of its exhilaration level, if it is alive (*Alive()*) and it can be examined what happens when the mood of the keeper changes. This latter operation (*Transmute()*) modifies the exhilaration level of the animal. Operations *Alive()* and *Name()* may be implemented in the base class already, but *Transmute()* just on the level of the concrete classes as its effect depends on the species of the animal. Therefore, the general class *Animal* is going to be abstract, as method *Transmute()* is abstract and we do not wish to instantiate such class.

General description of the moods is done the base class *Mood* from which concrete moods are inherited: *Good*, *Ordinary*, and *Bad*. Every concrete mood has three methods that show how a mood of a Fish, a Bird, or a Dog changes during the mood of the keeper changes.

The special animal classes initialize the name and the exhilaration level through the constructor of the base class and override the operation *Transmute()* in a unique way. Initialization and the override are explained in Section Analysis. According to the tables, in method *Transmute()*, conditionals could be used in which the mood of the keeper would be 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 moods, as all of the methods *Transmute()* in all of the concrete creature classes should be modified. To avoid it, the Visitor design pattern is applied where the ground classes are going to have the role of the visitor.



Methods *Transmute()* of the concrete animals expect a mood object as an input parameter as visitor and call the methods which corresponds to the type of the animal.



All the classes of the mood 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 track as every animal changes it. The 0^{th} version is the initial track. The change of the mood of the keeper is denoted by function *Transmute* : $Animal \times Mood^m \rightarrow Mood^m$ which gives the changed mood. i^{th} version of the track is denoted by $track_i$, which the program is not going to show, it is going to be just a temporal value of variable *track*.

A = $track: Mood^m, animals: Animal^n, minAlive: String^*$

Pre = $animals = animals_0 \wedge track = track_0$

Post = $track = track_n \wedge$

$$\forall i \in [1..n]: animal[i], track_i = UpdateMoodList(animals_0[i], track_{i-1}) \wedge$$

$$(l, elem) = \bigwedge_{e \in animals} animals[j].Alive() \wedge animals[j].Exhilaration \geq 5 \wedge$$

$$\forall j \in [1..m]: ((\min, elem)) = MIN\ animals[j].Exhilaration \wedge$$

$$minAlive = \bigoplus_{\substack{i=1..n \\ creatures[i].alive()}} < animals[i].name >$$

Concatenation of the animals and transmuting moods step by step are two Summations just as the assortment of the alive animals and Minimum Search algorithmic pattern has been used to find the animals with the minimum level of exhilaration.

Analogy:

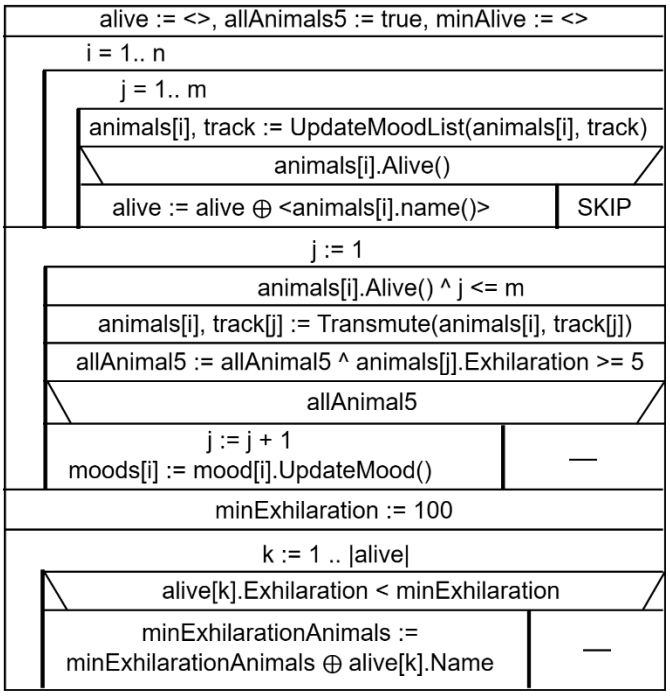
enor(E)	$i = 1 .. n$	
$f(e)$	$Transmute(animals[i], track)_1$	Value of function *UpdateMoodList()*
s	$Animal \times Mood$	
H, +, 0	$Animal^* \times Mood, \ominus, animals \times moods_0$	

enor(E)	$j = 1 .. m$
$f(e)$	$<animals[j]>$ if $animals[j].Alive()$
s	$Alive$
H, +, 0	$Animal^*, \wedge, false$

enor(E)	$j = 1 .. m$
$f(e)$	$animals[j].Exhilaration \geq 5$
s	L
H, +, 0	$Animal^*, \wedge, false$

enor(*E*)	*i = 1 .. n*
f(*e*)	<*animals*[*i*]>
s	<*minAlive*>
H, +, 0	*Animal**, ⊕, <>

For each day the keeper will have ith mood and the exhilaration level of animals being taken care by the keeper (1 .. mth) will be change according to the mood. If the exhilaration level of the animal reaches below 0, the animal dies so it will be skipped, if the exhilaration level is above 0, that animal will be added to the collection alive. Afterwards, the animal which are alive will be checked whether their exhilaration levels are above 5 and if all of them have more then 5, the mood of the keeper increases by 1. In the outer loop there is the minimum search which has a minimum exhilaration initialized to the maximum exhilaration level, and we will go through the animals which are alive and compare their exhilaration level with the initial minimum exhilaration level and the animal(s) with the least exhilaration(s) which are alive will be stored.



Testing

Grey box test cases:

1. Checking the changes in exhilaration of animals in different moods
2. Checking different mood improvements
3. Checking if an animal is alive after exhilaration level is 0 or lower
4. Checking the change of exhilaration of animals using a list of moods
5. Checking the mood changes of animal using a list of animals