OOP. Shokhrukh Nigmatillaev. Documentation of 2nd Assignment. Task 10.

Shokhrukh Nigmatillaev **2. Assignment/ Task 9** 16.05.2024

APVAVZ

apvavz@inf.elte.hu

Group 6

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 animals 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. Name the animal of the lowest level of exhilaration which is still alive at the end of the simulation. If there are more, name all of them!

A possible input file:

```
3
F Nemo 50
B Hedwig 70
D Lassie 20
oooooggbgbgoogg
```

Analysis

We have 3 different independent animals in our task. They are: **fish, bird, dog.** All of them have **animalName** and **exhilarationLevel** attributes. In our program we also introduced IMood interface that has 3 childs and implements different actions according to the type of animal and type of mood. They are: Bad, Ordinary, Good.

Bad:

animalName	exhilarationLevel
Fish	-5
Bird	-3
dog	-10

Ordinary:

animalName	exhilarationLevel
Fish	-3
Bird	-1
dog	0

Good:

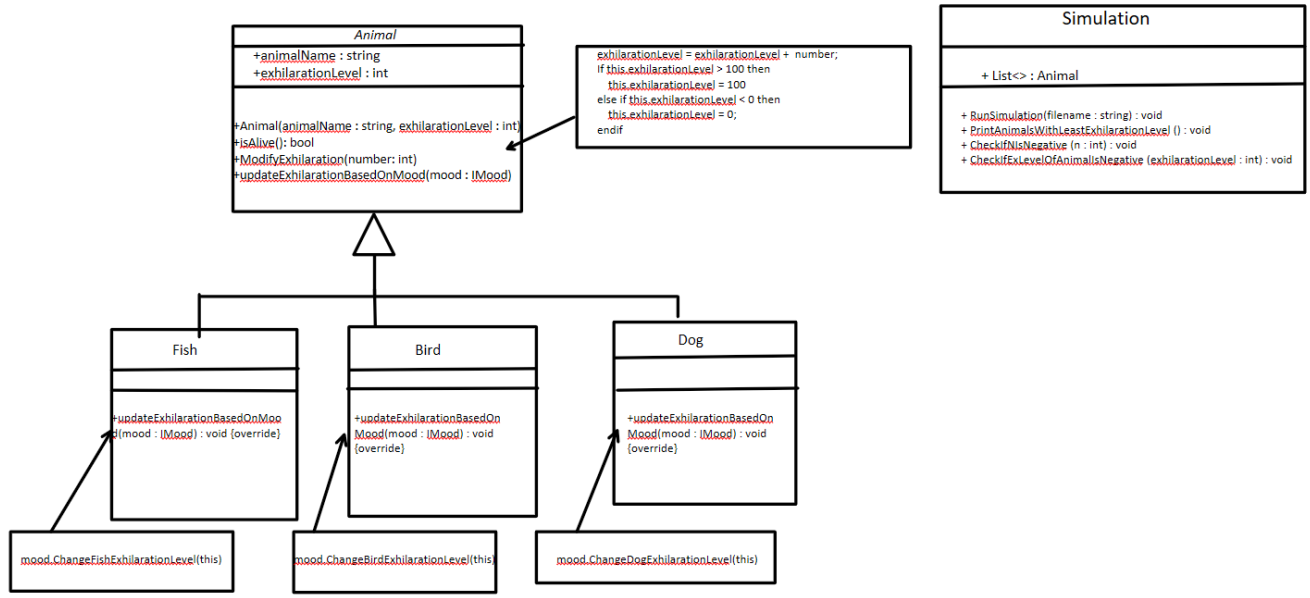
animalName	exhilarationLevel
Fish	+1
Bird	+2
dog	+3

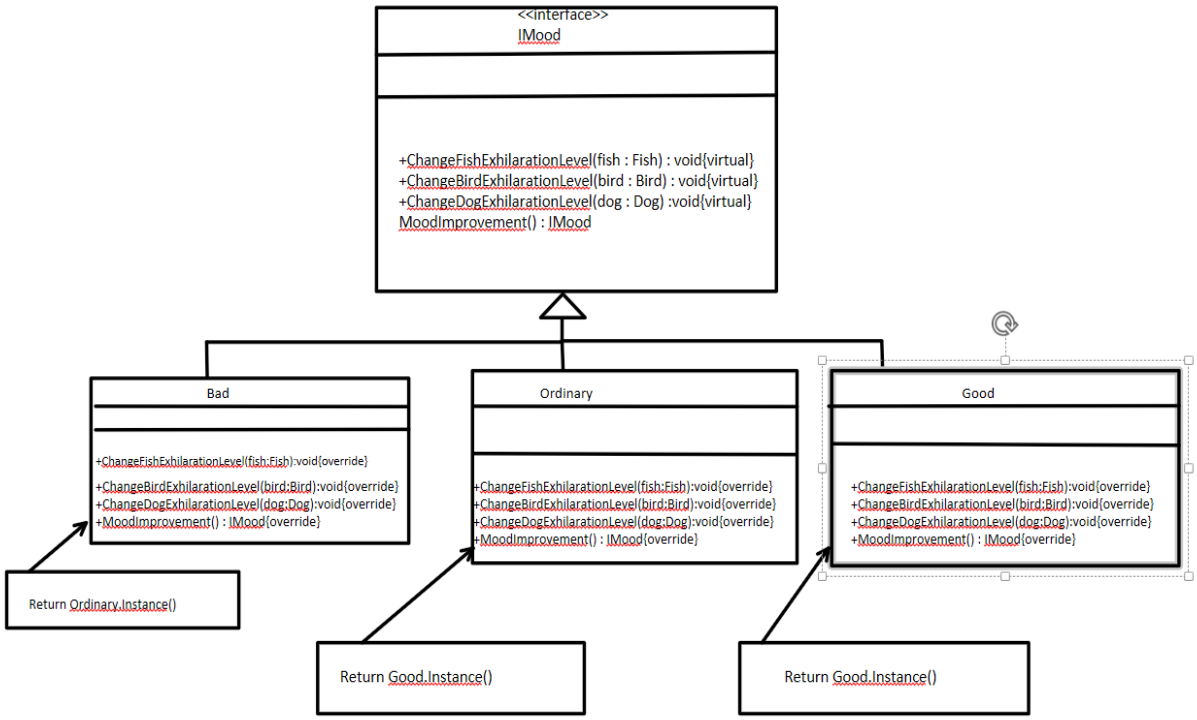
Plan

To describe whole program with animals and moods, I introduced abstract Animal class and 3 children classes inherited from it taking all attributes(string animalName, int exhilarationLevel) and methods (bool *isAlive()*, void *ModifyExhilaration()* and abstract void *updateExhilarationBasedOnMood())*. In children classes I am overriding *updateExhilarationBasedOnMood()* method taking parameter IMood object and according to that given parameter we call method to change exhilarationLevel of the current animal.

In IMood interface I created void methods that are used in the children classes that are Bad, Good and Ordinary. Each class has 4 methods that are being used with different parameters. So for fish class we have different implementation in sense of increasing or decreasing exhilarationLevel of it. And this is all similar implemented for each class. In this whole implementation **Sigleton** design pattern was used and objects of mood cannot be initialized more than once.

In Simulation class our whole simulation is going on. We create List of type Animal as an attribute. We take filename as an input and then continue reading from a file. While reading from file I introduced some Possible error cases that for example amount of animals cannot be negative number or exhilarationLevel of animals cannot be negative. For exhilaration level I originally put bounds as (0..100]. Then we continue in our code and changeExhilarationLevel of the animals in current day according to data given in the file. Then if all animals' exhilarationLevel is greater than 5, we improve mood by 1 unit measurement. After processing all moods and animals, we create another List type of animal in another *PrintAnimalsWithLeastExhilarationLevel*() where we list animals with the LeastAmountOfExhilaration level. And fot output we print all animals from this newly created list.





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

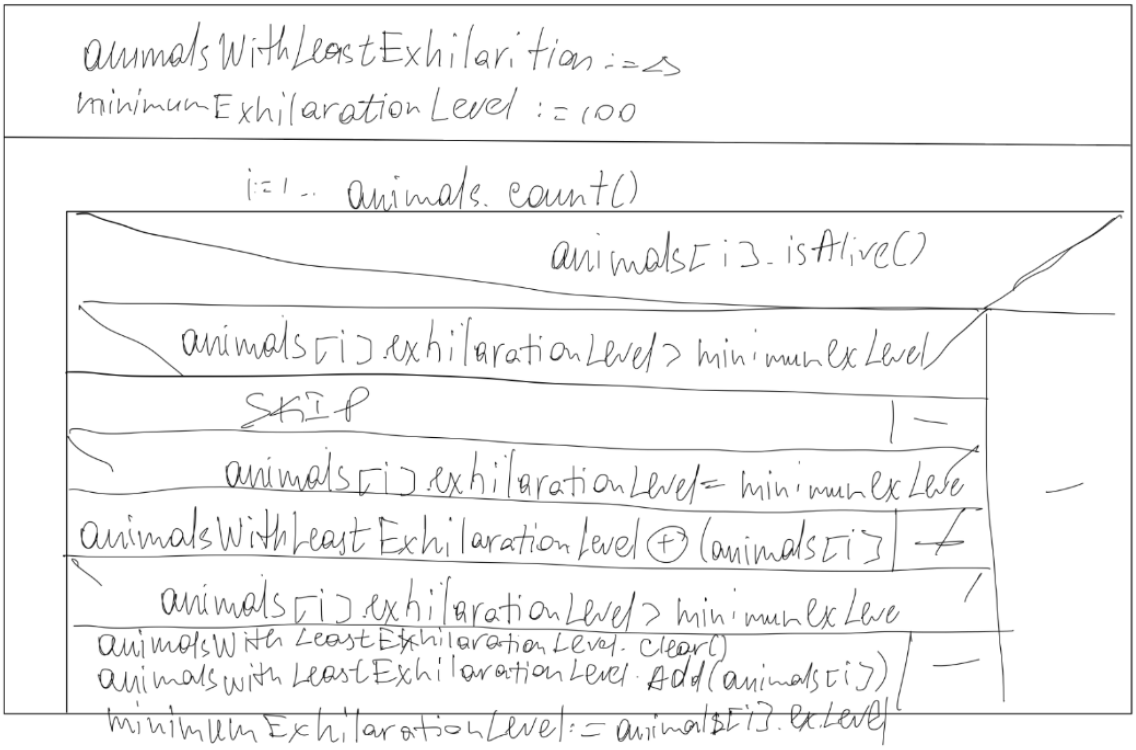
Specification:

A = (animals : Animal*, moods = IMood*, alive : bool)

Pre = (animals = animals' ^ moods = moods' ^ alive : true)

Post = animals ⊕ <Animal(animalName, exhilarationLevel)>(if ∀i∈[1..n]: animals[i].isAlive)
We check if animal is alive then add it by concatenation to our list.

Enor(E)	i = 1 .. n
s	animals
H, +, 0	Animal*, ⊕, <>



Algorithm of method of addirion to new List and ready to print with LeastExhilarationLevel animal(s)

Testing

White box test cases:

1. updateExhilarationBasedOnMoodTest()
in this method I check if moods have same effect on all animals and if their exhilarationLevel really changes with mood change
we declare all animal objects then all IMood objects and then start to use methods.

2. We check if given animals are alive. Checking if their exhilarationLevel is greater than 0.
We first declare Animal objects with different exhilarationLevel, I also added faulty value, so we can check and see correct values return true, faulty false – meaning not alive animal.

3. We check if triggered negativeAmountOfAnimals works passes test case
while reading from file if given number n is negative our error will be triggered and program catches it. Test cases indicates caught error. Special helper method was created for this test case.

4. Checking if the exhilarationLevel of animal in the file was given correctly
while reading from file if given animals' exhilaration level is negative we our error will be triggered program catches it. Test cases indicates caught error. Special helper method was created for this test case.

5. Checking method that improves mood Good -> Good, Ordinary -> Good, Bad -> Ordinary.
in this test case we initialize all moods and by order run MoodImprovement method. Bad improves to Ordinary, Ordinary improves to Good and Good returns itself.