Ali Kazim
OOP Practice
Z0w7jf
z0w7jf@inf.elte.hu

2. Assignment

Group IV

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 die). 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
ooooggbgbgoogg
```

Analysis[1]

Independent objects in the task are three animals. They are divided into three different group: Dogs, Birds, and Fishes.

All of them have the same Exhilaration, which we can get from a getter. We can know what can happen to the Exhilaration of these animals when the mood of Cathy changes. Mood effect the exhilaration as shown in the following tables:

Bird:

Mood	Exhilaration

Good	+2
Ordinary	-1
Bad	-3

Fish:

Mood	Exhilaration
Good	+1
Ordinary	-3
Bad	**-5**

Dog:

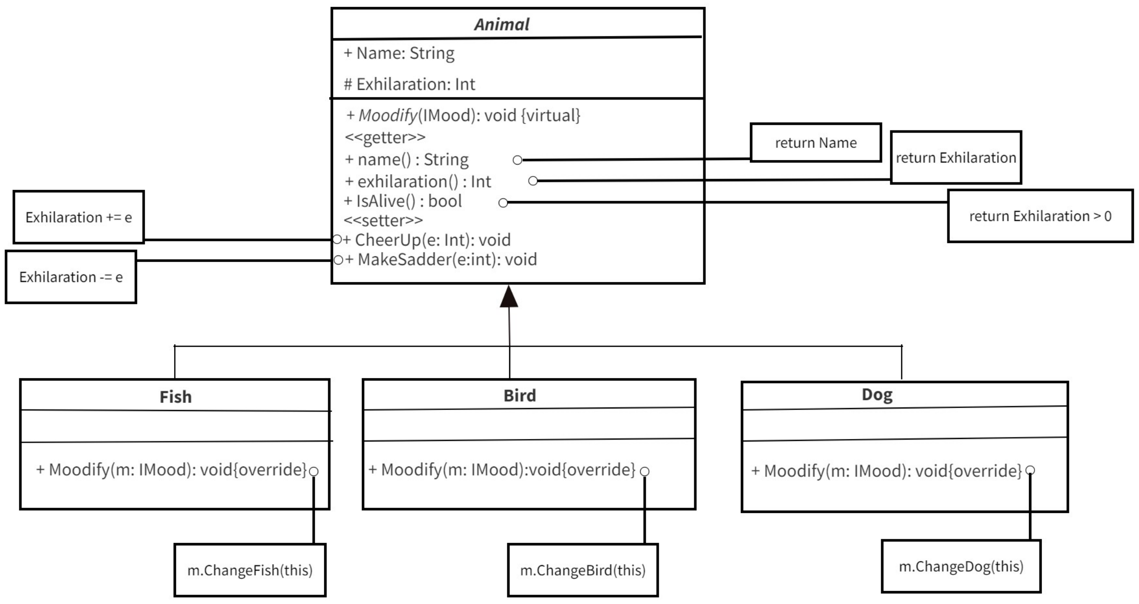
Mood	Exhilaration
Good	+3
Ordinary	-
Bad	**-10**

Plan[2]

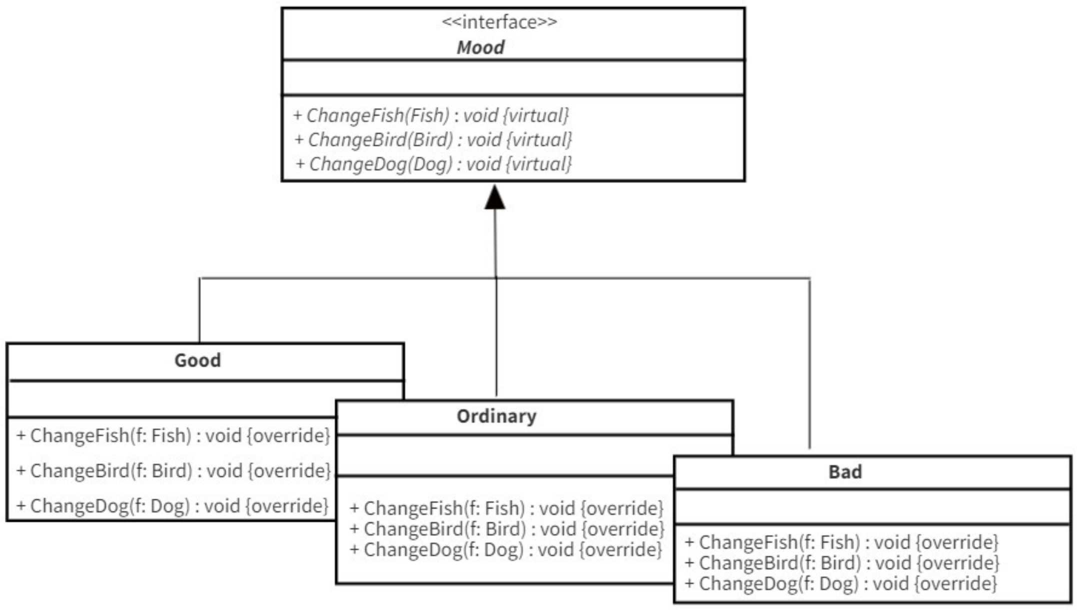
To put the animals inside the simulation, we will introduce four classes. The base class, Animal, will serve as the parent class, outlining the general properties shared among its three children: Fish, Bird, and Dog. These classes share common attributes like name and Exhilaration, and methods to retrieve these properties. The *isAlive()* function determines if the animal is still living. While operations like *CheerUp()* and MakeSadder() may already be implemented in the parent class, Moodify() is specific to each animal type. Therefore, the Animal class will be abstract, as *Moodify()* is abstract, preventing direct instantiation of this class.

The general description of moods is done by the parent class *Mood,* from which other moods are inherited: *Good, Ordinary* and *Bad*. Every concrete mood has three methods that show how the exhilaration changes according to the mood of Cathy.

The child classes of Animal class, initialize the name and exhilaration through the constructor of parent class and override the *Moodfy()* method in a special way. According to the given tables, in method *Moodify(),* several conditional can be used for the simulation. Putting many conditional can break the SOLID principle of object oriented programming and are not effective and efficient if the program is extended new Mood types, because we need to modify every *Moodify()* method in the concrete children classes. To avoid such situation, we use the Visitor design pattern, which is applied where the Mood classes are going to have the role of visitor.



Moodify() is expecting mood object as input parameter as a visitor and call the methods which corresponds to the animals.



<<interface>>
Mood
+ ChangeFish(Fish) : void {virtual}
+ ChangeBird(Bird) : void {virtual}
+ ChangeDog(Dog) : void {virtual}
Good
+ ChangeFish(f: Fish) : void {override}
+ ChangeBird(f: Bird) : void {override}
+ ChangeDog(f: Dog) : void {override}
Ordinary
+ ChangeFish(f: Fish) : void {override}
+ ChangeBird(f: Bird) : void {override}
+ ChangeDog(f: Dog) : void {override}
Bad
+ ChangeFish(f: Fish) : void {override}
+ ChangeBird(f: Bird) : void {override}
+ ChangeDog(f: Dog) : void {override}

Specification:

A = *mood:* Mood^n, *animals*: Animal^n, *alive:* String*

Pre = $animals = animals_0 \wedge mood = mood_0$

Post = $mood = mood_n \wedge$

$\forall i \in [1..n]: animals[i], mood_i = Moodify(animals_0[i], mood_{i-1}) \wedge$

$alive = \oplus_{i=1...n} < animals[i].name >$

animals[i].alive()

Analogy: First component of Moodify() function

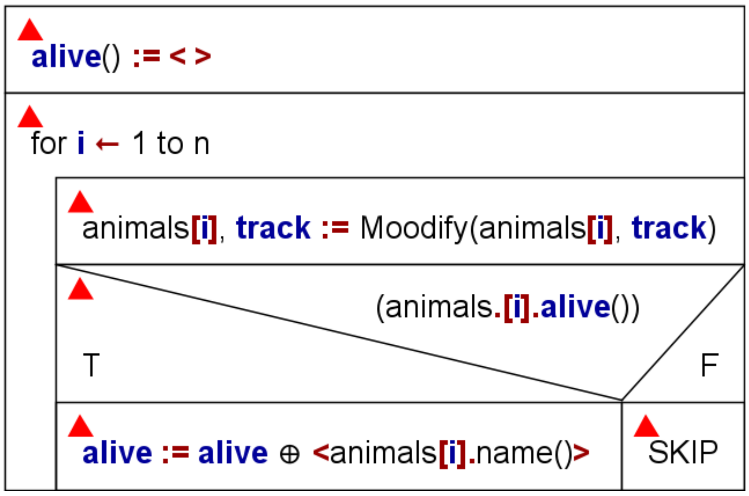
enor(E)	*i = 1 . . . n*
f(e)	$Moodify(animals[i], mood)_1$
s	*animals*
H, + , 0	*Animals* , ⊜, animals[i]*

Second component of *Moodify()* function where $a ⊜ b ::= b$

enor(E)	*i = 1 . . . n*
f(e)	$Moodify(animals[i], mood)_2$
s	*animals*
H, + , 0	*Mood* , ⊜, mood*

enor(E)	*i = 1 . . . n*

f(e)	*<animals[i]> if animals[i].alive()*
s	*alive*
H, + , 0	*animal* , ⊕, <>*

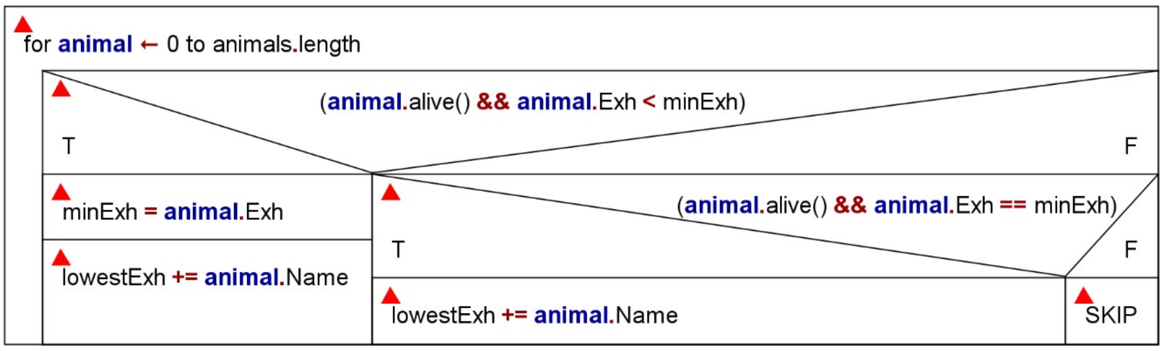


```
alive() := < >
for i ← 1 to n
    animals[i], track := Moodify(animals[i], track)
    (animals.[i].alive())
    T: alive := alive ⊕ <animals[i].name()>
    F: SKIP
```

The Animal with the lowest Exhilaration:

Alive() function only checks if the exhilaration level is below zero or not. And If it is then it declares the animal as dead. Here, we want to make a method which will give us the animals which are having the lowest exhilaration.

int minExh := 0
lowestExh() := < >



```
for animal ← 0 to animals.length
    (animal.alive() && animal.Exh < minExh)
    T:
        minExh = animal.Exh
        lowestExh += animal.Name
    F:
        (animal.alive() && animal.Exh == minExh)
        T: lowestExh += animal.Name
        F: SKIP
```

Testing:

White Box Testing:

1) checkCheerUp: Tests if the CheerUp method of a fish increases its exhilaration by a specified amount.

2) checkMakeSadder: Ensures that the MakeSadder method of a bird decreases its exhilaration by a specified amount.

3) CheckModifyFish: Verifies that applying a good mood to a fish increases its exhilaration by 1.

4) CheckModifyBird: Confirms that applying a bad mood to a bird decreases its exhilaration by 3.

5) DogCheckModify: Checks if applying an ordinary mood to a dog keeps its exhilaration unchanged.

GoodMood_changeFish: Tests if changing the mood to good increases a fish's exhilaration by 1.