11th May 2019

**3. assignment/10. Task**

**Sadi Mamedov**

LHXOTB

sadimamedov7@gmail.com

Group 2

# Task

*Hobby animals need several things to preserve their exhilaration. Steve has some hobby animals: tarantulas, hamsters, and cats. Every animal has a name and their exhilaration level is between 0 and 70 (0 means that the animals dies).*

*If their keeper is* ***joyful****, he takes care of everything to cheer up his animals, and their exhilaration level increases: of the tarantulas by 1, of the hamsters 2, and of the cats by 3.*

*On a* ***usual day****, Steve takes care of only the cats (their exhilaration level increases by 3), so the exhilaration level of the rest decreases: of the tarantulas by 2, and of the hamsters by 3. On a* ***blue day****, every animal becomes a bit sadder and their exhilaration level decreases: of the tarantulas by 3, of the hamsters by 5, of the cats by 7.*

***Steve’s mood improves by one if the exhilaration level of every 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 (T–Tarantula, H–Hamster, C–Cat), name of the animal (one word), and the initial level of exhilaration.*

*In the last line, the daily moods of Steve are enumerated by a list of characters (j–joyful, u–usual, b –blue). The file is assumed to be correct.*

***List the animals of the highest exhilaration level at the end of each day****.*

# Analysis1

Independent objects in the task are the animals. They can be divided into 3 different groups: Tarantulas, Hamsters and Cats.

All of them have a name and a power that can be got. It can be examined what happens when they pass period of days. Passing day affects the creature and the ground in the following way:

Tarantula:

|  |  |  |
| --- | --- | --- |
| mood | power change | mood change |
| blue | +1 | usual |
| usual | -2 | joyful |
| joyful | -3 | - |

1 This part may be skipped. It is enough to show the tables of swing in the Planning section.

Hamster:

|  |  |  |
| --- | --- | --- |
| mood | power change | mood change |
| blue | +2 | usual |
| usual | -3 | joyful |
| joyful | -5 | - |

Cat:

|  |  |  |
| --- | --- | --- |
| mood | power change | mood change |
| blue | -5 | usual |
| usual | -2 | joyful |
| joyful | +6 | - |

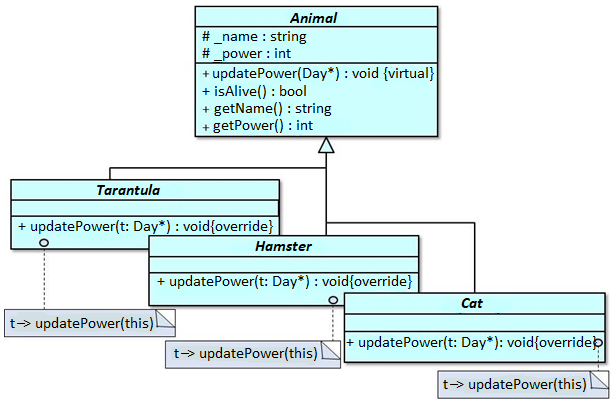
# Plan2

To describe the animals, 4 classes are introduced: base class *Animal* to describe the general properties and 3 children for the concrete kinds: *Tarantula*, *Hamster*, and *Cat*. Regardless the type of the animals, they have several common properties, like the name (*\_name*) and the power (*\_power*), the getter of its name (*getName()*), if it is alive (*isAlive()*) and it can be examined what happens when it passes a day. This latter operation (*swing()*) changes the daily mood, while operation *updatePower*() modifies the power of animal object. Operations is*Alive()* and get*Name()* may be implemented in the base class already, but *updatePower()* just on the level of the concrete classes as its effect depends on the species of the creature. Therefore, the general class *Animal* is going to be abstract, as methods *updatePower() and swing()* are abstract and we do not wish to instantiate such classes.

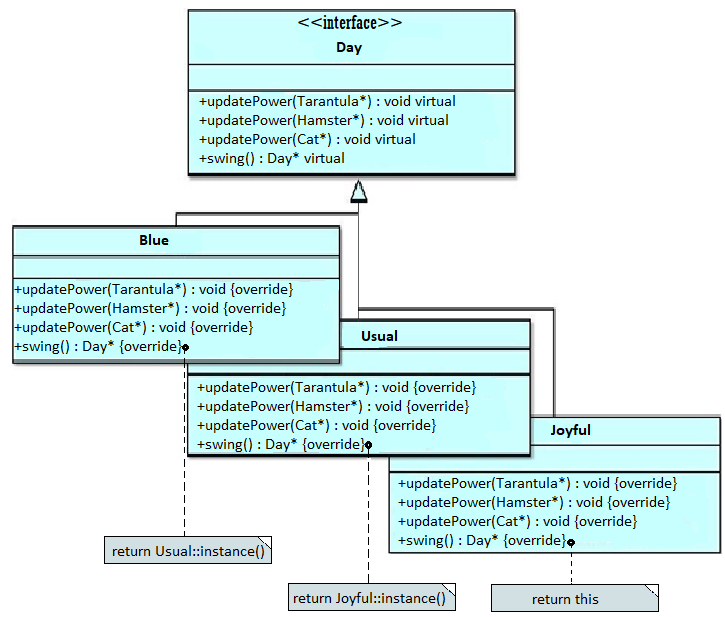
General description of the days is done the base class *Day* from which concrete days are inherited: *Blue*, *Usual*, and *Joyful*. Every concrete day has four methods that show how a Tarantula, a Hamster, or a Cat changes during crossing it and how the ground changes, too. Objects are referred by pointers.

The special animal classes initialize the name and the power through the constructor of the base class and override the operation *updatePower()* in a unique way. Initialization and the override are explained in Section Analysis. According to the tables, in method *swing()*, conditionals have to be used in which the type of the day is examined. Though, the conditionals are not effective if the program might be extended by new day types, as all of the methods *updatePower()* and *swing()* in all of the concrete animal classes have to be modified. To avoid it, design pattern Visitor is applied where the day classes are going to have the role of the visitor.

2 Plain text explanation is not necessary for the student documentations



Methods *updatePower()* of the concrete animals expect a day object as an input parameter as a visitor and calls the methods which corresponds to the species of the animal.

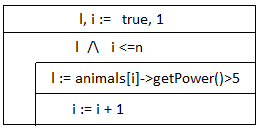


Program allOver5()

A = *animals*: vector<Animal\* *n* >

Pre = *animals = animals0*

Post = Post = (Pre ** l = ∀𝑆𝐸𝐴𝑅𝐶𝐻𝑖=1..𝑛 animals(i)-->getPower()>5)

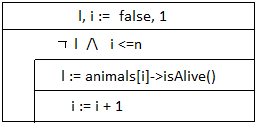


Program ExistAlive()

A = *animals*: vector<Animal\* *n* >

Pre = *animals = animals0*

Post = Post = (Pre ** l = 𝑆𝐸𝐴𝑅𝐶𝐻𝑖=1..𝑛 animals(i)-->isAlive())



Main Program

A = *track*: *Day m*, *animals*: *Animaln*, *highest k* : *int* Pre = *animals = animals0  track = track0*

Post = *track* = *trackn* 

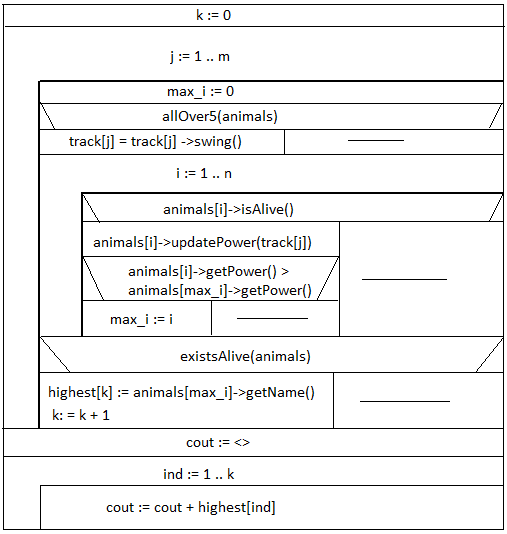
*i*[*1..n*]: *animals*[*i*], track = track -> swing() 

*AllOver5(animals)*

*max\_i* = 𝑀𝐴𝑋𝑖=1..*𝑛 animals(i).power*

*highest* = ⊕𝑖=1 .. 𝑛< *MAX* >

*existsAlive(animals)*



# Testing

Grey box test cases:

1. *Examination of functions updatePower() and swing()*

Different test cases depending on the animal and the mood

*2)Inner loop(length of the interval)*

length-based:

* + zero animal
  + one animal
  + more animals
  + no animal survives

*3)Outer Loop*

*- check auxiliary function allOver15*

*- check auxiliary function existsAlive*

*- if there exist at least one maximum before the period and after*

4)Maximum Check

1. One animal becomes healthier than the other next day
2. No maximum animal exists
3. Last animal is more durable at the end of period