|  |  |  |
| --- | --- | --- |
| **Mykyev A. Bakirovich**  FYQWAV  [mykyev.atabek@gmail.com](mailto:mykyev.atabek@gmail.com)  Group 4 | **3. assignment/10. Task** | 1st May 2022 |

**Task**

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

*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 by 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 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 (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.*

**Plan**

|  |
| --- |
| **Animal** |
| # name:  # ehxilarationLevel: |
| + getExhilarationLevel() : {query}  + getName() : {query}  + alive() : {query}  + takeCare(Mood\*) : {virtual}  + increaseExhilarationLevel(Int) : |

|  |
| --- |
| **Tarantula** |
|  |
| + takeCare(m: Mood\*) : {override} |

|  |
| --- |
| return name |

|  |
| --- |
| return exhilarationLevel |



|  |
| --- |
| return exhilarationLevel > 0 |

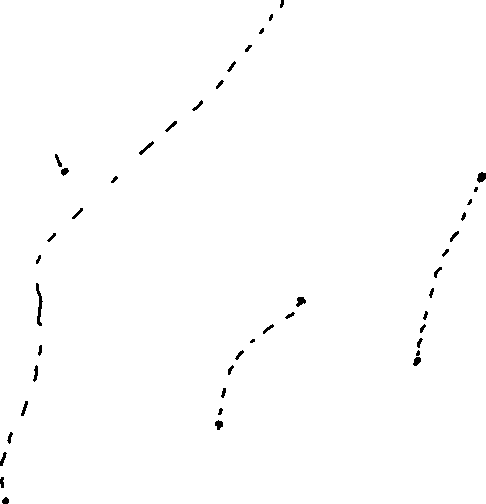


|  |
| --- |
| **Cat** |
|  |
| + takeCare(m: Mood\*) : {override} |



|  |
| --- |
| **Hamster** |
|  |
| + takeCare(m: Mood\*) : {override} |

|  |
| --- |
| m->takeCare(this) |



|  |
| --- |
| m->takeCare(this) |

|  |
| --- |
| m->takeCare(this) |

|  |  |  |
| --- | --- | --- |
| alive() exhilirationLevel + n > 70 | alive() exhilarationLevel + n > 0 | alive() |
| exhilarationLevel := 70 | exhilarationLevel := 0 | exhilarationLevel := exhilarationLevel + n |

Methods takeCare() of the concrete animals expect a mood object as an input parameter as a visitor and calls method which corresponds to the species of the creature.

|  |
| --- |
| **<<**Interface**>>**  **Mood** |
|  |
| + increase() : {virtual}  + takeCare(Tarantula\*) : {virtual}  + takeCare(Hamster\*) : {virtual}  + takeCare(Cat\*) : {virtual} |

|  |
| --- |
| **Joyful** |
|  |
| + increase() : {override}  + takeCare(a: Tarantula\*) : {override}  + takeCare(a :Hamster\*) : {override}  + takeCare(a: Cat\*) : {override} |

|  |
| --- |
| **Blue** |
|  |
| + increase() : {override}  + takeCare(a: Tarantula\*) : {override}  + takeCare(a :Hamster\*) : {override}  + takeCare(a: Cat\*) : {override} |

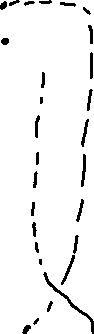


|  |
| --- |
| **Usual** |
|  |
| + increase() : {override}  + takeCare(a: Tarantula\*) : {override}  + takeCare(a :Hamster\*) : {override}  + takeCare(a: Cat\*) : {override} |

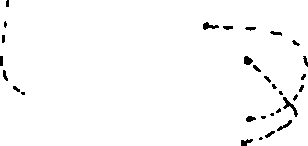
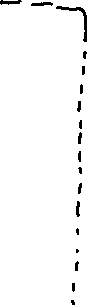


|  |
| --- |
| a->increaseExhilarationLevel(1) |

|  |
| --- |
| a->increaseExhilarationLevel(2) |



|  |
| --- |
| a->increaseExhilarationLevel(3) |



|  |
| --- |
| a->increaseExhilarationLevel(-2) |

|  |
| --- |
| a->increaseExhilarationLevel(-3) |

|  |
| --- |
| a->increaseExhilarationLevel(3) |

|  |
| --- |
| a->increaseExhilarationLevel(-3) |



|  |
| --- |
| a->increaseExhilarationLevel(-5) |



|  |
| --- |
| a->increaseExhilarationLevel(-7) |



Methods increase() return a higher mood which is Joyful in case of Joyful, Joyful in case of Usual and Usual in case of Blue.

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, we have n+1 versions of the mood and m versions of animals. The 0th version is the initial mood. We have to calculate animals’ exhilaration level every day which is denoted by function takeCare: which gives changed animal. After it we have to find maximal exhilaration level. And we should not forget that at the beginning of the each day we increase mood if all animals were alive.

Concatenation of the animals (whose exhilaration level is highest) after taking care of it is a summation, after it we have a maximum search which gives us value of the highest exhilaration level among the animals and at the end we have an assortment of such animals which is also a summation algorithmic pattern.

|  |  |
| --- | --- |
| enor(e) |  |
| f(e) |  |
| s |  |
| H, +, 0 |  |

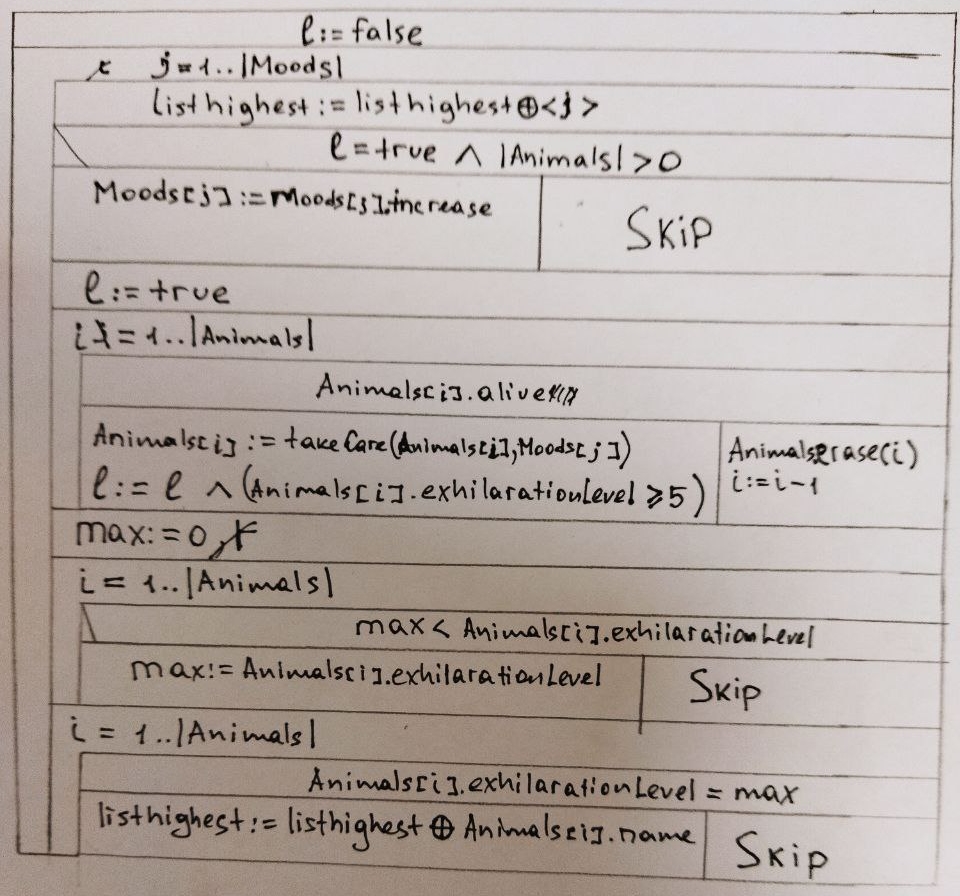
|  |  |
| --- | --- |
| enor(e) |  |
| f(e) |  |
| s |  |
| H, +, 0 |  |

|  |  |
| --- | --- |
| enor(e) |  |
| f(e) | Animals[i] |
| H |  |

Also, increasing of the mood of the keeper in the beginning of the day is included, which is optimistic linear search.

|  |  |
| --- | --- |
| enor(e) |  |
| cond(e) |  |
| t |  |

By merging the above to the same loop we get:



As we can see, there are some modifications.

We initialize *l* to *false* at the beginning, before moods loop – because we don’t know yet, if all the animals are alive, we did not take care of them and we can’t know, whether they all are going to be alive.

We add the day to the *listhighest* to know, on which day we gather animals.

We *takecare* of animal only if he is alive – we do not need to takecare of the dead animal, instead, we say, that if he is not alive, then we *erase* him from animals arbitrary set.

In maximum search, instead of initializing *max* to the first animals’ exhilaration level, we initialize it to the lowest possible value of it, which is 0. We do it in case if all the animals are dead, and there’s no animal in the animals set.

**Testing**

Gray box test cases:

*Inner loop*

1. Length-based:

* Zero animal
* One animal
* More animals

1. First and last:

* First animal survives or not after all days
* Last animal survives or not after all days

*Outer loop*

1. Length-based:

* One animal after 0 days
* One animal after 1 day is taken care
* One animal after more days (dies or survives)

1. First and last

* First mood stays the same after
* Second mood changes or stays the same
* Last mood changes or stays the same

*Examination of function takeCare()*

1. *There is tarantula or hamster or cat with exhilaration level 0 and the joyful day or usual day or blue*
2. *There is tarantula or hamster or cat with exhilaration level 1 and the joyful day or usual day or blue*
3. *There is tarantula or hamster or cat with exhilaration level 70 and the joyful day or usual day or blue*