OOP Cristea Andrei: Documentation, 3’rd assignment.

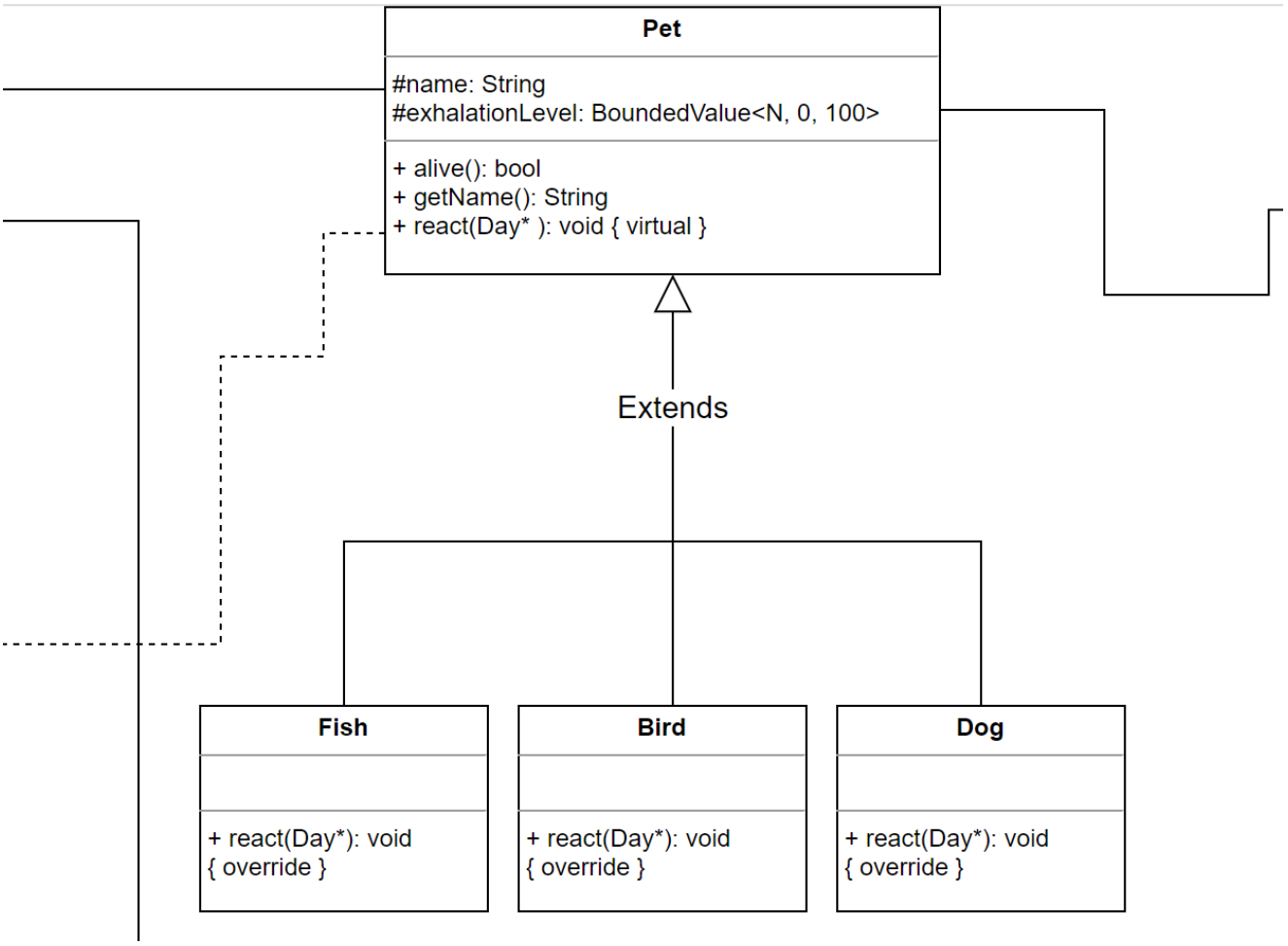
Cristea Andrei NEPTUN CODE	**3 assignment. Task 9** **W61RAB**	05.05.2021
w61rab@inf.elte.hu Group 4		

Task

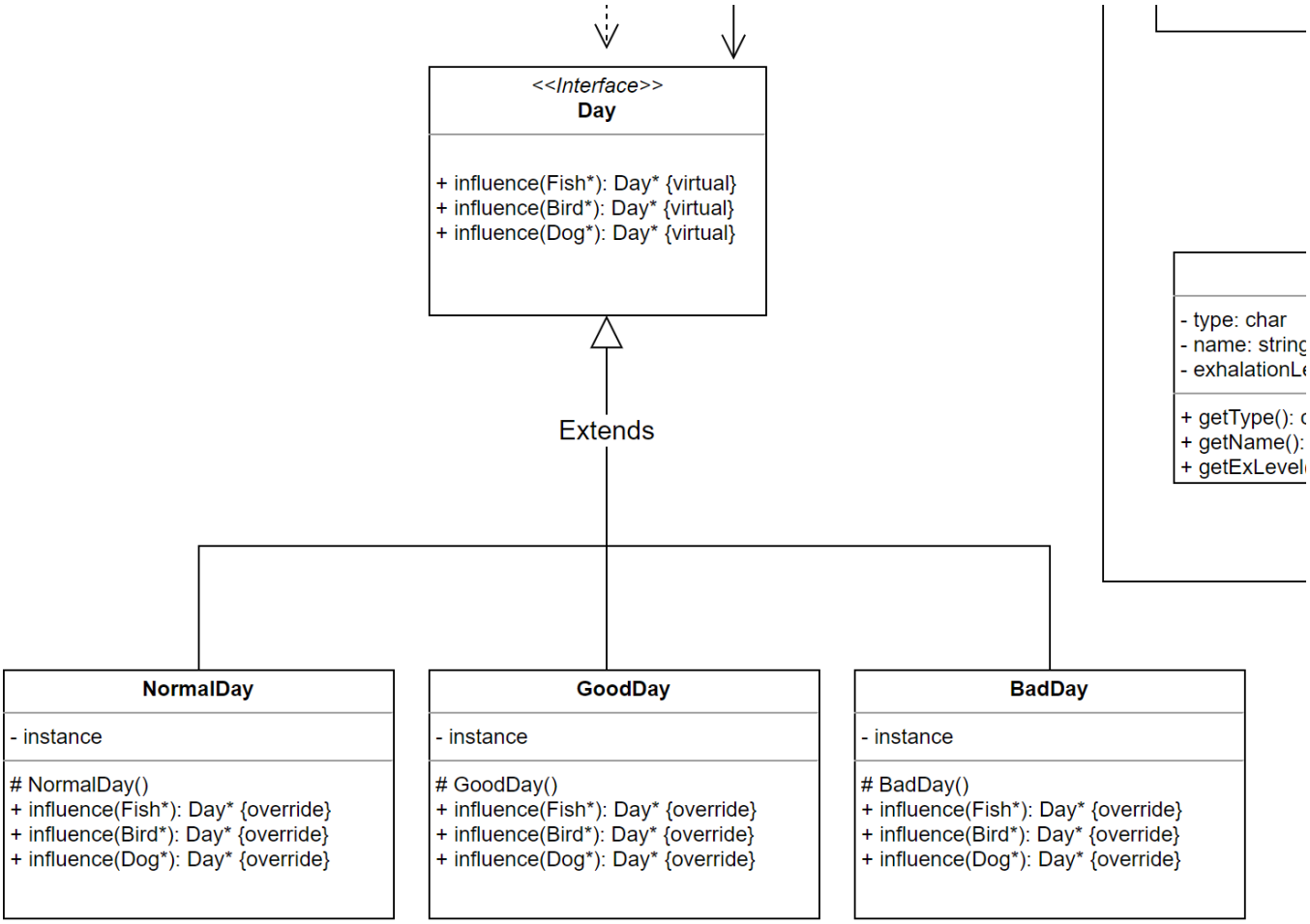
Hobby animals need several things to preserve their exhilaration. Cathy has some hobby animals: fishes, birds, and dogs. Every animal has a name and their exhilaration level is between 0 and 100 (0 means that the animal dies). If their keeper is in a good mood, she takes care of everything to cheer up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

(1) Analysis

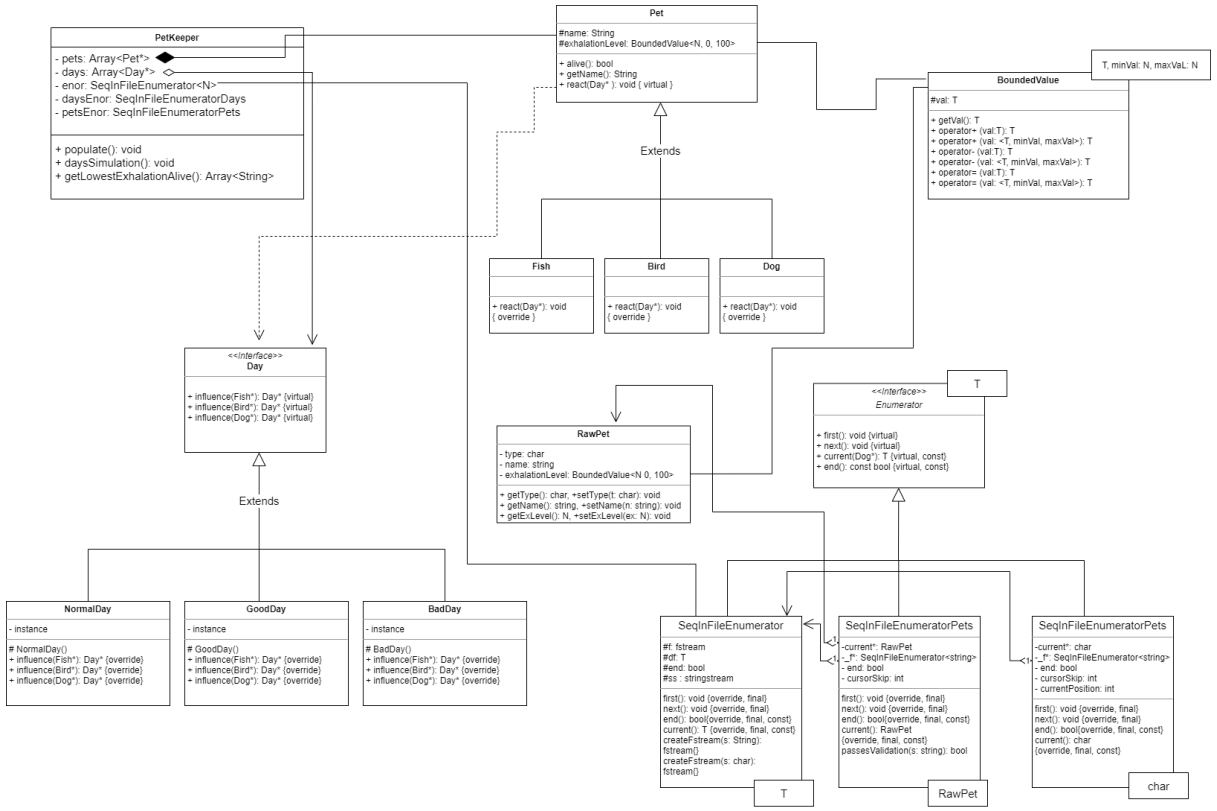
We will have the following most important classes in our program **PetKeeper**, **Pet** and ***Day***. Other classes will be helping handling the file input (via Enumerators) or will provide some supplementary project features (as error handling and integer type restriction to bounded area). PetKeeper class is one one which will contain Pet and Day classes collection, as well as their Enumerators and where major calculations will occur (like populating with days and pets, getting pets with lowest exhalation level and so on). Pet class will be extended by Bird, Dog and Fish classes and will contain almost all important logic, therefore here pattern Template will be used. Also every child class implements *react()* method which takes as a parameter current day.



Also, class Day is working as <<*interface*>> for NormalDay, BadDay, and NormalDay. For optimization purposes these classes will be used as Single Tones, thus instantiated only once by a special static function constructor. Other than that Day extended classes will take role of visitors for Bird, Dog and Fish classes respectively



Full UML Diagram



Better quality can be found here: https://github.com/havenousername/cplusplus-assingments/blob/main/assignment%233/assignmentULM.svg

Specification

In the specification, we would need to use Minimum Search and Summation algorithmic patterns. Firstly, it is needed to simulate days for pets. The crossing of one pet with corresponding day is denoted by function *influence()* : Pet × Day → Pet × Day which gives the changed pet. After simulating days' influence, we can enumerate over pets in searching for smallest exhalation leveled pets. This will be concatenated to the *lowest* string and the smallest number will be stored in temporal integer *exhalation.*

A = $days$: $Days^{m}$, $pets$: Pet^{n}, $lowest$: $String$ *, $exhalation$: N

Pre = $days = days_0 \wedge pets = pets_0$

Post =

$days = days_0 \wedge \forall i = [1..m]: (\forall j = [1..n]: pets_j = react(pets[j]_0, days[i])$

$s = MIN_{i = 1..n,\ pets->getExhalationLevel() < exhalation}\ pets -> getExhalationLevel()$

$exhalation = s \wedge$

$lowest = \bigoplus_{i = 1..n,\ pets->getExhalationLevel() = exhalation} < pets -> getName() >$

Analogy

1. Pets Simulation

enor(E)	i = 1..m, j = 1..n
f(e)	$react(pets[j]_0, days[i])$)
s	pets
H, +, 0	Pet*, $\bigoplus$, <>

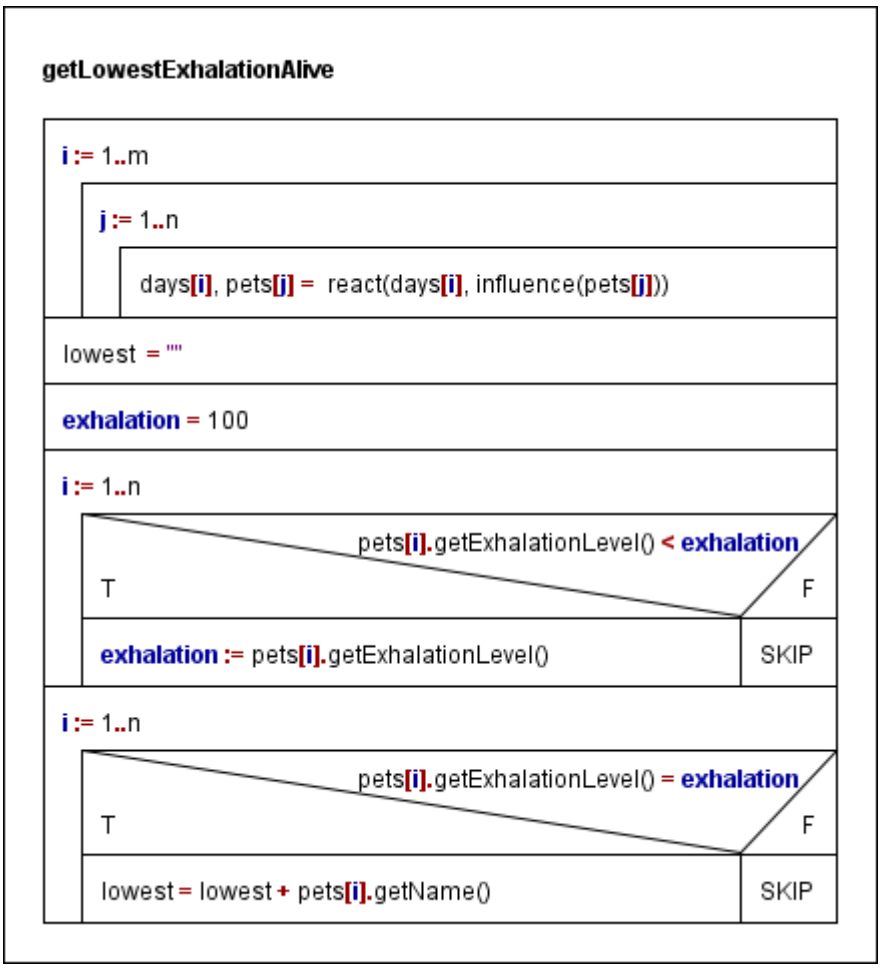
2. Minimum Search

enor(E)	i = 1..n
f(e)	$pets -> getExhalationLevel()$
s	s
cond	$pets -> getExhalationLevel() < exhalation$

3. Summation

enor(E)	i = 1..n
f(e)	$pets -> getName()$
s	$lowest$
cond	$pets -> getExhalationLevel() = exhalation$
H, +, 0	String, $\bigoplus$, <>

Structogram



Testing

Grey box test cases

Outer loop

1. length-based
    - no days
    - one day only
    - two or three days
2. algorithm based
    - same day
    - different days

Inner loop

1. length-based
    - no pets
    - one pet only
    - two or more
2. first and last (and multiple)
    - only first pet will have the smallest exhalation after simulation
    - only last pet will have the smallest exhalation after simulation
    - multiple pets have smallest exhalation
3. algorithm based
    - all pets will die
    - all pets will live
    - all pets are Dogs, or Fish, or Bird

Examination of function *react()*

Nine different cases depending on pet and day