PROGRAMMING ASSIGNMENT 2

TAs : Emirhan SELİM
Due Date : 4.05.2024 (23:00:00)
Programming Language: Java 8 (Oracle)

Zoo Manager

1 Introduction

In this assignment, you will make use of OOP principles, mainly Abstraction and Polymorphism, to create a system that simulates the management of a zoo. You will also define and handle custom exceptions.

2 System Definition

2.1 Animals

The animal itself and various details and operations related to them need to be implemented. All animals will have their own names, ages and daily meal sizes. Different Types of animals each have a default meal sizes that can change based on the animals age. All animals can be fed and their habitats can be cleaned. Each type of animal is fed and their habitats are cleaned in different ways. A custom exception with a self explanatory message should be used if there is not enough food for feeding.

2.1.1 Lion

Lions eat meat. A lion at age 5 eats 5kgs of meat per meal, increasing or decreasing by 50 grams per age difference.

2.1.2 Elephant

Elephants eat fruits and hay, classified as plants. An elephant at age 20 eats 10kgs of plant per meal, increasing or decreasing by 15 grams per age difference.

2.1.3 Penguin

A penguin at age 4 eats 3 kgs of fish, increasing or decreasing by 40 grams per age difference

2.1.4 Chimpanzee

A chimpanzee at age 10 eats 3 kgs of Meat and 3 kgs of plants. The total amount it eats increases or decreases by 0.025 per age difference

2.2 People

Everyone at the zoo have their own names and ids. There are never 2 people with same ids. Any of these people may attempt to use the system to either attempt to feed an animal or visit the animal. Throw a custom exception and handle it if the person in question does not actually exist in the system.

2.2.1 Personnel

When an authorized personnel visits an animal, they clean the animal's habitat. They are also authorized for feeding the animal. You should also handle the case in which the animal doesn't exist

2.2.2 Visitor

They do not have the authority to feed an animal, which results in an exception. They can visit animals but not clean their habitats

3 Input and Output

Your inputs will be taken from .txt files. Keep in mind that the filenames given here may change and your program needs to take filenames from command line arguments. Initialization files don't have mistakes but commands.txt does have lines with mistakes in them such as mentioning a nonexisting animal. These should cause exceptions and stop the operations but not stop the program's runtime

3.1 Initialization

These files will be used to initialize your system. and your program needs to take the filenames as command line arguments in the format I will explain later.

3.1.1 Animals

Animals at initialization time will be read from animals.txt file. The format for all animals will be <Type>,<Name>,<Age>

3.1.2 Persons

Animals at initialization time will be read from person.txt file. The format for all animals will be <Type>,<Name>,<ID>

3.1.3 Foods

Food at initialization time will be read from foods.txt file. The format for all animals will be <Type>,<Amount> There are only 3 different types of food: meat,plants and fish.

3.2 commands

These are the various operations that the program executes after initialization.

The command for "Feed Animal" follows the following format:

<operation type>,<Id>,<Animal name>,<number of meals>

The command for "Animal Visitation" follows the following format:

<operation type>,<Id>,<Animal name>

The command for List Food Stock follows:

<operation type>

Here is an example animals file below:

```
Lion,Aslan,10
Elephant,Boncuk,6
Penguin,Leo,8
Chimpanzee,George,4
Lion,Simba,3
Elephant,Dumbo,27
Penguin,Kowalski,2
Chimpanzee,Mojo,16
```

An example persons file:

```
Visitor,Ahmet,124031
Visitor,John,357023
Personnel,Jack,532146
Personnel,Abigail,186534
Visitor,Ayşe,755332
```

An example foods file:

```
Meat,50
Fish,80
Plant,50
```

An example commands file:

```
List Food Stock
Animal Visitation,532146,Aslan
Animal Visitation,755332,Aslan
Feed Animal,755332,Aslan,2
Feed Animal,532146,Aslan,2
Feed Animal,532146,George,5
```

```
Animal Visitation,186534,Kowalski
Animal Visitation,124031,Leo
List Food Stock
Feed Animal,186534,Kowalski,7
Feed Animal,186534,Mojo,5
Feed Animal,532146,George,5
Animal Visitation,532146,George
Animal Visitation,532146,Boncuk
List Food Stock
```

And the output file expected from those inputs:

```
**********************************
***Initializing Animal information***
Added new Lion with name Aslan aged 10.
Added new Elephant with name Boncuk aged 6.
Added new Penguin with name Leo aged 8.
Added new Chimpanzee with name George aged 4.
Added new Lion with name Simba aged 3.
Added new Elephant with name Dumbo aged 27.
Added new Penguin with name Kowalski aged 2.
Added new Chimpanzee with name Mojo aged 16.
**********************************
***Initializing Visitor and Personnel information***
Added new Visitor with id 124031 and name Ahmet.
Added new Visitor with id 357023 and name John.
Added new Personnel with id 532146 and name Jack.
Added new Personnel with id 186534 and name Abigail.
Added new Visitor with id 755332 and name Ayşe.
**********************************
***Initializing Food Stock***
There are 50.0 kg of Meat in stock
There are 80.0 kg of Fish in stock
There are 50.0 kg of Plant in stock
**********************************
***Processing new Command***
Listing available Food Stock:
Plant: 50.0 kgs
Fish: 80.0 kgs
Meat: 50.0 kgs
**********************************
***Processing new Command***
Jack attempts to clean Aslan's habitat.
Jackstarted cleaning Aslan's habitat.
Cleaning Aslan's habitat: Removing bones and refreshing sand.
**********************************
***Processing new Command***
Ayşe tried  to register for a visit to Aslan.
```

```
Ayşe successfully visited Aslan.
**********************************
***Processing new Command***
Ayşe tried to feed Aslan
Error: Visitors do not have the authority to feed animals.
**********************************
***Processing new Command***
Jack attempts to feed Aslan.
Aslan has been given 12.5 kgs of meat
**********************************
***Processing new Command***
Jack attempts to feed George.
George has been given 12.75 kgs of meat and 12.75 kgs of leaves
**********************************
***Processing new Command***
Abigail attempts to clean Kowalski's habitat.
Abigailstarted cleaning Kowalski's habitat.
Cleaning Kowalski's habitat: Replenishing ice and scrubbing walls.
**********************************
***Processing new Command***
Ahmet tried  to register for a visit to Leo.
Ahmet successfully visited Leo.
**********************************
***Processing new Command***
Listing available Food Stock:
Plant: 37.25 kgs
Fish: 80.0 kgs
Meat: 24.75 kgs
**********************************
***Processing new Command***
Abigail attempts to feed Kowalski.
Kowalski has been given 19.32 kgs of various kinds of fish
**********************************
***Processing new Command***
Abigail attempts to feed Mojo.
Mojo has been given 17.25 kgs of meat and 17.25 kgs of leaves
**********************************
***Processing new Command***
Jack attempts to feed George.
Error: Not enough Meat
**********************************
***Processing new Command***
Jack attempts to clean George's habitat.
Jackstarted cleaning George's habitat.
Cleaning George's habitat: Sweeping the enclosure and replacing branches.
**********************************
***Processing new Command***
```

```
Jack attempts to clean Boncuk's habitat.
Jackstarted cleaning Boncuk's habitat.
Cleaning Boncuk's habitat: Washing the water area.
**********************************
***Processing new Command***
Listing available Food Stock:
Plant: 20.0 kgs
Fish: 60.68 kgs
Meat: 7.5 kgs
```

4 Test and Execution

Your code must be compiled and executed under Java 8 and dev.cs.hacettepe.edu.tr. If your code does not compile and execute under developer server, then you will be graded as 0 for code part even if it works on your own machine. Your program must take input files, and name of the output file. Sample run command is as follows:

```
javac Main.java
java Main animals.txt persons.txt animals.txt commands.txt output.txt
```

5 Grading Policy

Task	Grade
Class Implementations using OOP Principles	40
Exception Handling	20
Correct Output	30*
Comments in JavaDoc Style	10**
Total	100

***Even though producing correct output and commenting seems like enough to get full credit, you must obey to the given rules in the PDF (for example using concept of OOP and four pillars of OOP etc.) otherwise you may face with some point deductions which may result with a grade that is as low as zero. There will be two overall multipliers about quality of your OOP and clean code separately which will vary in between 0 and 1! Note that there may be any other multipliers or point deductions in case of violation of the rules.**

**** The score of the clean code comment part will be multiplied by your overall score (excluding clean code comment part) and divided by the maximum score that can be taken from these parts. Say that you got 45 from all parts excluding clean code comment part and 10 from clean code comment part, your score for clean code comment part is going to be 10*(45/90) which is 5 and your overall score will be 45+5=50.**

6 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system).

```
-b<studentID>.zip
  -Main.java, or *.java
```

Late Policy

You have two days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You must submit your solution in at the most two days later than submission date, otherwise it will not be evaluated. Please do not e-mail to me even if you miss the deadline for a few seconds due to your own fault as it would be unfair for your friends, e-mail submissions will not be considered if you do not have a valid issue.

Notes and Restrictions

- Your code must be able to execute on our department's developer server (dev.cs.hacettepe.edu.tr).
- You must use JavaDoc commenting style for this project, and you must give brief information about the challenging parts of your code, do not over comment as it is against clean code approach. Design your comments so that if someone wants to read your code they should be able to easily understand what is going on. You can check here to access Oracle's own guide about JavaDoc Sytle.
- **Use abstraction and polymorphism where it makes logical and practical sense. Otherwise, you risk losing points for not properly applying OOP principles.**
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- Do not miss the submission deadline.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as "clean code".
- Use UNDERSTANDABLE names for your variables, classes, and functions regardless of the length. The names of classes, attributes and methods must obey to the Java naming convention. This expectation will be graded as "coding standards".
- You can ask your questions through course's Piazza group, and you are supposed to be aware of everything discussed in the Piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.

- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.