Object-Oriented Programming – Exercise Week 3

This week, we are going to create a zoo full of animals. We will follow a similar menu structure as in week 2. The menu will again be operated via command line.

The learning goal for this exercise

1. Create an object (Zoo) which contains other objects (Animals)
2. Use of ArrayList (a dynamic array for storing data)
3. Use of constructors when creating new Animals
4. User input handling (prevent crashing when faulty selection is given)

Submission

You need similar project structure than in the week 1. In addition to App.java file containing the main method and the menu structure, you need Zoo.java and Animal.java classes.

Through menu, you can create new animals into the zoo. Each animal will have information of the species in question, the name of the animal and age of the animal. All of these values are defined in the constructor.

Once created, the brand new animal will be passed onward to the Zoo object that will store the animals in an ArrayList.

Through the menu, you can also tell zoo objects what to do. For this, addAnimal, listAnimals and runAnimals methods must be included in the zoo object.

The listAnimals method prints out the details of every animal (species, age and name). This can be achieved using for example with the for (each) -loop.

The runAnimals will make every animal in the zoo run based on the number of laps user gives as an input. Therefore, you need to ask the user how many laps the animals shall run. The animals will run in the same order as they are in the ArrayList.

1 point implementation

Feature 1: The program asks for the zoo's name when launching.

Feature 2: Animals can be added to the zoo with addAnimals() command. This command takes the parameters of species, age, and name.

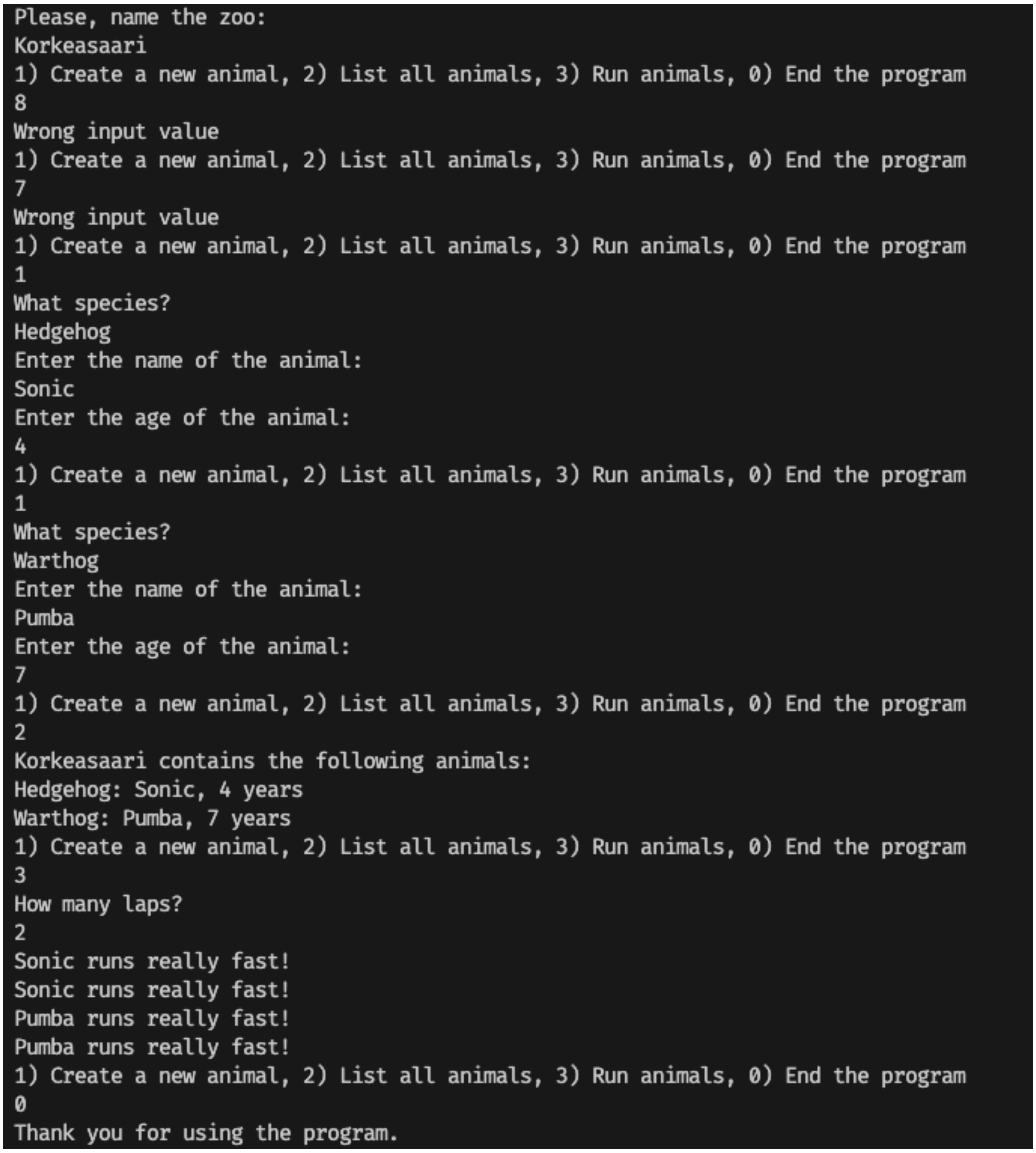
Feature 3: All animals can be listed with the listAnimals() command.

2 point implementation

Feature 4: All animals in the zoo can be successfully ordered to run with the zoo.animalsRun() command.

Feature 5: If the user tries to pick an option that was not included in the menu, the system will not crash but tell the user to pick a valid option. There is no need to handle non-integer values (e.g., user inputing text such as “hello”). Just handle numbers greater than the existing options.

Example console output and input for the program (see next page)



```
Please, name the zoo:
Korkeasaari
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
8
Wrong input value
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
7
Wrong input value
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
1
What species?
Hedgehog
Enter the name of the animal:
Sonic
Enter the age of the animal:
4
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
1
What species?
Warthog
Enter the name of the animal:
Pumba
Enter the age of the animal:
7
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
2
Korkeasaari contains the following animals:
Hedgehog: Sonic, 4 years
Warthog: Pumba, 7 years
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
3
How many laps?
2
Sonic runs really fast!
Sonic runs really fast!
Pumba runs really fast!
Pumba runs really fast!
1) Create a new animal, 2) List all animals, 3) Run animals, 0) End the program
0
Thank you for using the program.
```