CS230 Labs

Lab: Zoo of Animals

Using arrays of Objects

Goals:

1. user-defined classes in java
2. overriding methods (toString(), equals())
3. arrays, and in particular arrays of Objects
4. think carefully about good testing
5. practice with writing javadoc
6. work with command-line arguments
7. using java API and javadoc

PRE-LAB tasks to be completed before coming to lab. Make sure your preLab work is easily accessible at the start of the lab. One easy way to do so, would be to have it uploaded to your gradescope account.

Pre-Lab tasks:

- Task1 The Animal class, new version

PRE-LAB

TASK 1: The ANIMAL class (new version)

Study this newer version of the Animal class. In particular, pay attention to an extra instance variable, named `goodHealth` and its getter and setter. Also, notice a new instance method, named `equals()`. This is an important method for our purposes of maintaining a collection of Animal objects!

- What does it mean for two animal instances to be "equal" according to the way this method is defined?

- What other new elements do you see there? Any questions on any of them? Write them down and bring them to the lab.

TASK 2: The ZOO

Set up

In this task you will define a class named *Zoo*, to create and maintain a Zoo. An instance of the Zoo class has animals, and the user should be able to interact with it. For example, the user should be able to create a Zoo, to add animals to it, to remove animals from it, etc.

In BlueJ, create a new project, name it *lab3Project* and save it on your Desktop. Add a class named *Animal* to the project.

Open this link Animal class definition in a new tab. Then *Select (command+A), Copy (command+C)* and *Paste (command+V)* the whole web page's content in your Animal class.

Compile the code to make sure all looks well so far.

TASK 2-1: THE INSTANCE VARIABLES AND THE CONSTRUCTOR

Next, create a new class named *Zoo*, in the same project.

We will use an array as the container for the animals in the Zoo. Start this array with a certain capacity (let's say, 3). Keep in mind that -in general at any point during execution- the array will not be full, instead it will have available spots to accept more animals. (When it becomes full, it can be expanded so that it can accommodate more animals. More on this later.) **Therefore, in general, the length of the array will not be the same as the number of the animals it contains!** That's why you need another variable -let's call it ***numAnimals***- to hold the actual number of animals the array contains at any point.

Now you should be able to set the instance variables in the Zoo class!

The next step is to define the constructor in the Zoo class to create the array, i.e. allocate the necessary memory space, and initialize the instance variable *numAnimals*.

TASK 2-2: toString() AND main() METHODS

Define the method `toString()`. This method should return a string representation of a Zoo instance, including the number of Animals it contains, as well as a description of each animal in it. The *toString()* method defined already in the Animal class should be very handy here!

Time to add the `main()` method to your class, to start testing your code. Create a Zoo instance, empty at first, and print it. Is everything as expected?

TASK 2-3: ADDING ANIMALS TO THE ZOO

Define the `addAnimal()` method, to do exactly that: add an animal to this zoo. Does this method need an input (typical) parameter? What type? Should this method return anything?

How should this method behave if there is no more available space in the array to hold the new animal?

Any other special case(s) that you might be thinking regarding the behavior of this method?

Make sure you state, in the top-of-the-method javadoc comments, any assumptions you made in the process of defining this (and for that matter, any other) method.

TASK 2-4: DO WE HAVE A VET IN THE HOUSE?

Define a method, `getNumOfAnimals()` to return the number of all animals in the zoo.

Define a method, `getNumOfHealthyAnimals()` to return the number of healthy animals in the zoo.

Define a method, `getNumOfSickAnimals()` to return the number of animals in the zoo that are not healthy.

Finally, define a method, `getHealthyAnimals()` to return an array with all the animals in the zoo whose health is good. This is a trickier method because you need to return an array of animals, not just an integer. How big should this array be? Is there a method that you have already written that can tell that?

TASK 2-5: FINDING ANIMALS IN THE ZOO

Define a method, `findAnimal()`, which should return the index in the array the animal was found at. Does this method need an input? What should it return if the animal was not found in the array? You will need to call the `equals()` method here, defined in the Animal class.

Make sure to test this method thoroughly. Think about what cases you need to cover in your testing. Make a note about it in your notebook.

TASK 2-6: COMMAND-LINE ARGUMENTS

In the `main()` method, arrange for the program to take one command-line argument: the initial capacity of the Zoo. Think, how can you test this setup?

Command-line arguments means that your program will expect input from the command-line at the time the program is executed.

When command-line arguments are provided, they are stored in the `String[] args`, which is the input to the `main()` method:

```
public static void main (String[ ] args)
```

This means the first argument is stored in args[0], the second in args[1], and so on.

A good resource on command-line arguments can be found here:

Java's Command-line Arguments tutorial

Test your code to verify that the array was indeed created with the command-line inputted capacity!

OPTIONAL: TASK 2-7: ADD SOME JAVADOC

Are you totally comfortable with writing javadoc at this point? If not, spend a little more time on it: Add some good javadoc to your program. Take another look at the official javadoc from Oracle

First, look at the Animal.java version you downloaded earlier. Then, in the BlueJ Editor window, from the drop down menu, in the upper right corner, choose "Documentation". What do you see? Then open the folder that contains your project. Do you see anything new there?

SUBMITTING

Submit `Zoo.java` as a group, by uploading to Gradescope, in lab3.

EXTRA CHALLENGES?

TASK 2-7: EXPAND ZOO CAPACITY

Define a method `expandZoo()` to expand the Zoo capacity. Make it so that the capacity of the Zoo doubles after calling this method. When would such a method be called? What kind of tests can you run to make sure this method works as expected?

TASK 2-8: THE REMOVEANIMAL() METHOD

Define the `removeAnimal()` method, to remove an animal from this zoo. Does this method need an input (typical) parameter? Should it return anything? What would be the return type? Can we invoke one of the existing methods in this method?

Recall that all empty slots in an array are expected to be found in the right-most part of it, i.e. an array should contain no"holes"!

SAMPLE SOLUTIONS

- Animal.java
- Zoo.java