Lab 2

Objectives:

- Javadoc
- ArrayList
- File I/O
- UML diagrams

This is the follow up of Zoo software you were working on. You have some sample data files provided. You need to modify your software so that the zones and animals will automatically be loaded into the system from the given files. Also, more information can be monitored.

Getting Started

Create a new Java project in Eclipse, named according to the lab guidelines (mandatory guidelines of laboratories - available on the blackboard). For this lab, you may reuse your code from Lab 1, but you should correct any mistakes. If you copy the files over, ensure that you choose "copy" if prompted, rather than "link", as the latter will not move the file into this project directory.

Your project should now contain **Zoo.java**, **Zone.java**, and **Animal.java**. All classes in this lab will be in the default package of your project.

Your application will read in data from text files placed in an **animalData** directory. Sample files (animals.csv and zones.csv) can be downloaded from the blackboard. Unzip it and place the folder with files at the top of your project in Eclipse. *Note: the top of your project is within the folder for the project, not within the src folder.*

To get you started, there is a test class provided, **Lab2.java**. Your final submission must include this class exactly as it appears, and the data files given. The only permitted modification to this code is to add Javadoc comments as needed. Once your application is completed, running Lab2.java with the given data files will result in the exact output shown in the attached PDF file.

Carefully inspect the output and modify your toString() methods accordingly.

** UML diagram can be incorporated with your Eclipse project or can be submitted as a separate file.

Animal.java

Animal objects will be as previously defined in Lab 1, additionally they will have:

- A zone code (i.e. T for the Tiger zone)

All class variables must have getters and setters.

Zone.java

With addition to previous definition of Zone objects they will have:

- An **ArrayList** of Animal objects (you can discard previous Array!)
- A zone code (i.e. T for the Tiger zone) in addition to the existing zone name.
- A required safety rating (i.e. low, medium, high, critical)

This class should have a method **addAnimal** which takes in an Animal and adds them to that respective Zone.
It should also have a method **removeAnimal** which takes in an Animal and removes them from that Zone.

All class variables must have getters and setters.

Zoo.java

The Zoo class will be as previously defined; however it will also contain an **ArrayList** of Zone objects, rather than an array.
In order to work with the code provided in Lab2.java, this class will need two new methods: **loadZones(..)** and **loadAnimals(..)**. Both methods will take in a file name and throw an IOException. These object methods will load the data in the given file into the appropriate objects (zones are added to the zoo, and animals are added to respective zones).

Create an object method **relocate(..)** which takes as parameters the name of an animal and the zone code to which they should be relocated. This method should remove the animal from their currently assigned zone, and add them to the zone given by the zone code parameter.

Create an object method **save()** which takes no parameters and saves all current information of the zoo back into the data files provided. *This way, if any animals have been relocated, their new zone information will appear with their name in the zoo files.* Be sure to *overwrite* the data in the file, rather than append to it. The data must be saved in the same format as provided in the CSV files.

All class variables must have getters and setters.

Rubric:

- (30pts) Zoo.java
- (30pts) Animal.java and Zone.java
- (20pts) Correctness - Your program will be tested using different data files, in the same format as given.
- (10pts) Javadoc comments on all classes, including getters and setters. Each class must have your name and abc123 at the top, in addition to the description of the class.
- (10pts) UML diagram

Submissions which do not compile will receive a maximum of 10 points in total.