Write an object-oriented program in Java 8 or later that implements a Zoo full of Animals. Animals that must appear in your Zoo are represented on slide 16 of Lecture 5. Create a class structure like slide 17 of lecture 5 with at least three levels of inheritance (e.g. Animal, Feline, Cat). You may decide how many of each Animal live in your zoo, but there should be at least two instances of each lowest level subclass (like Cat). Individual Animals should have constructors that create Animal instances with individual names that start with the same letter of their subclass (e.g. Charlie and Chloe for Cat).

Add a fourth type of Animal of your choice at the same class level as Feline. There must be at least two unique subclasses of this Animal (like Cat and Lion) also of your choice.

When run, the program will ask for input of a number of days. On each day, a Zookeeper object will arrive at the Zoo and execute a set of tasks in order. Methods for the Zookeeper class include: wake the animals, roll call the animals, feed the animals, exercise the animals, and tell the animals to sleep. Once these tasks are complete for that day, the Zookeeper will leave the Zoo. When the Zookeeper performs any action, a string associated with the action should be displayed as output. Examples: “Zookeeper arrives at Zoo on Day 1.”, “Zookeeper feeds Charlie the Cat.”, etc.

The Zookeeper methods should be defined using an abstract class called ZooEmployee. The Zookeeper class will extend the abstract class ZooEmployee. As you choose, methods that are common to ZooEmployees should be defined in the abstract class. Zookeeper may override these methods or define methods particular to itself.

To respond to the Zookeeper object methods, each Animal instance in the Zoo must have a method that corresponds to the Zookeeper’s tasks: wake up, make noise, eat, roam, and sleep. These methods should also print a string indicating the Animal executing the task, ex: “Charlie the Cat eats.”, “Charlie the Cat sleeps.”, etc. For the strings used by both Zookeepers and Animals, they should be generated based on the executing object’s attributes (such as name, type, or day number), and should not be hardcoded for individual instances.

You should demonstrate the use of overriding inherited methods by having some subclasses override methods at each of the two levels after the Animal superclass. For instance, an Animal may have a default eat method, but you may define alternate definitions for eat at the Feline or Cat level. You will choose which methods will be defined at which level and what is different about them.

Allow for the following special cases. When a Dog is given the exercise command by the Zookeeper, there is a 25% chance the dog will dig instead of roaming. When a Feline is given a sleep command, there is a 30% chance it will roam instead, a 30% chance it will make noise, and a 40% chance it will sleep. When a Pachyderm is given an exercise command, there is a 25% chance it will charge instead of roam. Use a random number function to generate these results.

In your internal program documentation, you should provide comments that clearly identify at least one instance of: abstraction, encapsulation, polymorphism, abstract class, and identity. For demonstrating polymorphism, it should be possible to issue method calls to the collection of all Animal instances in the Zoo regardless of their subclasses.

Remember, in your internal program documentation, you must cite the source (by URL) of any code elements taken from online sources (such as Stack Overflow). You may not directly use or duplicate

code being submitted by other students, although you may (and should) talk with others about solution approaches. You may want to credit fellow students or class staff for elements they helped with.

Run the program with a period of seven days. Collect all output from the Zookeeper actions and the Animals responses. You will likely need to initialize the program by creating instances of Animals and the Zookeeper; you will also need some form of a collection for the Animals in the Zoo.

Capture the text output of the final program in an output text file called "dayatthezoo.out". Your submission must include your Java code, a README (described below), and this output text file.

Grading Rubric:

This programming assignment is worth 50 points for each of the two/three team members.

10 points: Java code functionality (demonstrated in code and in the output text file) as requested, including flow of zoo activities and random elements. (-1 or -2 points for incomplete or missing behavior.)

15 points: Java code structure (the code must be completely OO-based using communicating objects, and should not include significant procedural style code) (-1 to -3 for poor commenting, excessive procedural code, or poor logic; -15 if not an OO solution as requested.)

15 points (3 points each): Internal code documentation (comments) highlighting examples of five OO terms: abstraction, encapsulation, polymorphism, abstract class, and identity. (-1 to -3 for poor, incomplete, or missing terms.)

10 points: README Markdown document including project title, project team names, any comments or assumptions made for development, any issues encountered during development, and any special instructions to run the application. Use Markdown syntax to make the file more readable, including use of section headers. (-1 or -2 for incomplete or missing elements.)

There is no extra credit available for this assignment.

The project will be submitted by providing a GitHub Repository URL that contains: the code, the output text file, and the README Markdown file. All class staff must have access to the repo (if private).

Homework/Project 1 is Due at 12 Noon on Wednesday 9/16.

Assignments will be accepted late as follows. There is no late penalty within 4 hours of the due date/time. In the next 44 hours, the penalty for a late submission is 5%. In the next 48 hours after that the late penalty increases to 15% of the grade. After that point, assignments will not be accepted.

Late penalties will be waived for health or medical related issues (of course).

Use office hours, e-mail, or Piazza to reach the class staff regarding homework/project questions.