Week 2 Lab Exercise

You may begin working on the lab exercise as soon as it is released to the class – you can start working at home and you don't have to wait until your scheduled lab time. We recommend that you read all of the instructions until the end before writing any code.

You can come in during scheduled lab times for CSCI 235 to ask questions and get help from the lab instructors and teaching assistants. You may discuss the tasks with your classmates and friends, but you are NOT allowed to share code, or even show your code to one another.

When you are done and ready to submit your lab to Moodle (on the lecture page, not the lab page), find your **src** folder within the project directory, zip it up, and submit the resulting archive file. Any other submissions, such as code copied and passed into text files will not be accepted.

Submission deadline is Friday, August 26, at 11:00 pm.

The maximum grade for this lab is **2 points**. To get the maximum, all of the following tasks should be completed, and your code should compile and run properly. Late submissions will receive 0 points.

Instructions

For this lab, you should implement the following program which simulates a zoo that contains hungry animals. You need to use the proper fields, constructors, and methods in your classes, and you need to use proper component-based design principles as outlined in the lessons. The specifications are as follows:

1. You should first create an **Animal** class. Every Animal should have a:
   a) Name
   b) Weight
   c) Kind of food they eat (represented as a String)
   d) Quantity of this food that they eat in a day
   e) Kind of environment they live in (also represented as a String)
   f) A number of square meters their living environment should take up

   Be sure to create a proper constructor that takes parameter arguments for all of these values so they can be immediately set. Also create a toString method which shows all of these values as a single String, formatted in an easy-to-read way.
2. Create a **Zoo** class that keeps track of at least 100 Animals. You may use an array for this, or you can make use of an ArrayList<Animal>, though you might have to look up how to use them. This class should also provide the following methods:
   a) addAnimal(Animal) -- allows you to add new animals to your zoo
   b) numOfAnimals( ) -- returns the current number of animals in your zoo
   c) toString( ) – returns a string which lists all of the animals in your zoo (you should use the toString method on your animal objects to help in this!)

3. Create three new different classes, each of which is a subclass of Animal. Here, we also provide the specific daily food needs and environmental requirements for all animals of that particular type:
   a) **Giraffe** – always eats 300 units of leaves a day; needs 1500 m^2 of grassland
   b) **Tiger** – always eats 200 units of meat a day; needs 1000 m^2 of grassland
   c) **Penguin** – always eats 15 units of fish a day; needs 20 m^2 of water

   The constructors for these three classes should take the name and weight of the specific animal as parameter arguments, while also setting the common food and environment requirements for each.
4. Create a **ZooTest** class that contains the main method for your program. In its body, it should create a new Zoo object, and add several Giraffes, Tigers, and Penguins to your zoo. Test out the different methods to make sure everything is working fine before going on to the next step.
5. Now, you should add the method **totalFoodNeeded(String)** to your Zoo class, which returns the amount of a specific type of food needed for that day. For example, if your zoo contains 3 Penguins, calling totalFoodNeeded("fish") should return the value 45, or if you call totalFoodNeeded("pizza") it should return 0.
6. Similarly, you should add the method **totalSqMetersNeeded(String)** to Zoo, which returns the total number of square meters needed for the given environment type. For example, if you zoo contains 2 Giraffes and 1 Tiger, the call totalSqMetersNeeded("grassland") should return 800.
7. Add some additional test cases to **ZooTest** to test out these two new methods.
8. For some more fun, create some additional Animal types with different requirements, and test them out!

Important note about Java Strings: If you want to check if two Strings in Java are equivalent, you should always use the equals method, e.g., str1.equals(str2), instead of the == operator. That is because equals checks for value equivalence, whereas == checks for reference equivalence.