**CS 1302 – Test 3 Review Problems**

A screenshot of a computer screen

Description automatically generatedHere are some sample problems that may help you prepare for Test 3. These are based on the material from chapters 4 and 5. I do not have solutions for these problems available at this time. Problems 1-5 are based on the class diagram provided below.

1. Write the *Animal* class header, *equals* method, and *feed* abstract method. The equals method will return true if two *Animals* have the same name. You do not need to write the instance variable nor the other methods not mentioned above.
2. Write the entire *Bear* class.

* The *feed* method should reduce *hunger* by 1, and return the (new) *hunger.* However, *hunger* cannot fall below zero. Thus, if *hunger* is zero, a call to *feed* will not change *hunger* and simply return zero.

1. Write the entire *MommaBear* class.
   1. If number of cubs is not specified in the constructer, it should be set to the default of 2.
   2. The *feed* method should work the same as in the *Bear* class, but the return value should be increased by the number of cubs.
2. Write the entire Zoo class.
   1. You can assume the class has a no-arg constructor which you do not need to write.
   2. The class has a list of *Animals*.
   3. *addAnimal* accepts an *Animal* and adds it to the list of *Animals*.
   4. *getAnimal* returns the *Animal* at *index* if *index* is valid; otherwise, it returns *null*.
   5. *removeAnimal(index:int)* removes the *Animal* at *index* from the list of *Animals* and returns the removed *Animal.* You can assume the index is valid.
   6. *removeAnimal(a:Animal)* – Removes *a* from the list of *Animals*, returning *true* if successful and *false* otherwise. Note: we haven’t written a method like this in class or HW; however, it is simple. Hint: research the *ArrayList* method, *remove(object:Object).*
   7. Add a *getAnimal* method (not shown in diagram) to the *Zoo* class that accepts a *name* and returns the *Animal* with that name, if it exists; and *null* otherwise. Your code should be as efficient as possible – this means that it should not use a loop. Hint: use the “dummy” object approach used in class and HW.
   8. Add a *removeAnimal* method (not shown in diagram) to the *Zoo* class that accepts a *name*. This method should remove the *Animal* with that name and return it, if it exists; and should return *null* otherwise. Your code should be as efficient as possible – this means that it should not use a loop. Hint: use the “dummy” object approach used in class and HW.
   9. The *getMommaBears* method returns an *ArrayList* of *MommaBears* with at least as many cubs as specified in the parameter. For example, *getMommaBears(2)* would return Ursa with 2 cubs and Barda with 3 cubs, but not Sammy with 1 cub.
   10. The feedAll method calls the feed method for all animals at the zoo and returns the sum total of the returned values.
3. Write the following snippets of code:
   1. Make a Bear object and store it in a variable.
   2. Make two MommaBear objects using different constructors, storing both in different variables.
   3. Make a Zoo object and store it in a variable.
   4. Add all three Animals created above to the Zoo.
   5. Assume lots of animals have been added to the zoo. Get the 8th animal from the zoo and store it in a variable
   6. Remove the 5th animal in the zoon and store the result in a variable
   7. Remove the animal with name, “Quola” from the zoon and store the result in a variable.
   8. Call a method that returns a list of the mamma bears from the zoo with at least 3 cubs and be sure to define a variable to hold the return from this method.
   9. Call the feedAll method and catch the result in a variable.
4. Write a *Fighter* interface. The Fighter interface has an *attack* method that accepts a double and doesn’t return anything.
5. Short explanation: Modify the class below so it can be sorted alphabetically by name.

Longer explanation: Implement the *Comparable* interface such that an *ArrayList<Person>* could be sorted on *name*. You are not to write code to sort an ArrayList, you are modifying the *Person* class so that a list could be sorted. Hint: there are 3 separate things you need to do.

**public** **class** Person {

**private** String name;

**private** **int** age;

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **int** getAge() {

**return** age;

}

}