Bluegrass Community and Technical College

Programming Requirements Document

Inheritance

NARRATIVE DESCRIPTION

Attached to this assignment in Blackboard is a **LivingCreature** class. This is a parent class. Look over the code. You will notice it has 4 instance data items:

- **isAlive**: a Boolean variable which indicates whether or not the living creature is alive (true or false)
- **dateOfBirth**: a String variable which indicates the birthdate of the creature (mm/dd/yyyy)
- **dateOfDeath**: a String variable which indicated the date the creature died (if the creature is not alive)
- **description**: a String variable that describes the creature

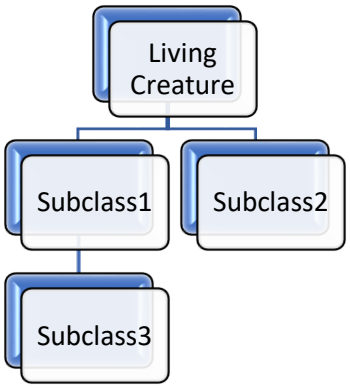
The **LivingCreature** contains two constructors, accessors, mutators, and a toString() method. This code is similar to what you are accustomed to seeing in a class from Chapter 5.

The **LivingCreature** class is an **abstract** class because it has one method, the **superPower()** method, which is not defined. This means any class that is a child class of **LivingCreature** would need to define **superPower( )** (unless it too is an abstract class).

Also attached to this assignment is a sample child class named **Human** and a **driver named CreatureTest.** Compile all three java files and run the driver. Pay attention to how the child class uses the parent class (in the constructors, in the toString method, and by defining the **superPower()** method).

Step 1: Create Child Classes (Subclasses) of LivingCreature

- Create 3 new and different subclasses of **LivingCreature**. Two should be inherited from **LivingCreature**. The third subclass should be inherited from one of the first two subclasses you created.



You are not to use any examples from the Internet or textbooks. Use your imagination and what you have learned about inheritance to complete this assignment. You may choose animals, plants, mythical creatures, etc. NOTE: You should name the classes with descriptive names. Do not use Subclass1, Subclass2, and

Subclass3. Quality if factored into your grades. Students who expend minimal effort on this assignment will not earn full credit.

- For each class you create, include:
  - At least 3 additional pieces of instance data items (notice how Human had 4 additional items). Points are awarded for well-designed classes. Use relevant data items which are appropriate for the creature. Do not mirror the Human class. Make sure your data items are unique. Your child classes should not mirror each other. Minimal effort is awarded minimal points.
  - Constructors for the new classes (don't forget the call to the parent using super() – refer to Human, as needed).
  - Accessors and mutators for the new instance data items.
  - An updated toString( ) method - refer to Human as an example.
  - An overridden superPower( ) method – refer to Human as an example. Points are awarded for well-designed methods. Creativity factors into your grade.
  - Proper use of inheritance (using super( ) to avoid duplication of code). Points are awarded for well-designed methods. The toString() method should be relevant and make sense.
  - Include data validation where appropriate (for parameters values).

Step 2: Testing the new Creature Classes

- Create a new driver program that displays the data in a better format than was provided.
- Instantiate (with data) at least one object from each of the new classes you defined. Document this coding section well.
- Display each of the new objects using the **toString()** methods of each class. Document this coding section well. Include headings in the output indicating the toString() methods are used.
- Display each object using accessors rather than the toString() methods. Make sure you use each of the accessors from each class. This is testing each of your accessors. Document this coding section well. Include headings in the output indicating the accessors are used.
- Update each object by using the mutators for each object. Make sure you use each mutator from each class. Document this coding section well. Include headings in the output indicating the mutators are used.
  - Display the old data and new data for each change.
- Unfortunately, one of your creatures died. Use the proper mutators in the **LivingCreature** class to change the settings for **isAlive** and **dateOfDeath**. Document this coding section well. Include headings in the output indicating the methods from LivingCreature is used.
  - Display the old data and new data for the change.

Step 3: Describe your classes

- In a Word document or other file saved as a PDF file, describe IN DETAIL your subclasses. Describe the data items you included and why. Your grade will reflect how well you planned your creature classes.

Zip all of .java files together, including your driver, and your Word or PDF file. Submit in Blackboard for grading.

Restrictions:

You are not to use arrays for this assignment. This is an assessment of our basic knowledge and understanding of Java inheritance.

Software Requirements

R1: Properly create the 3 subclasses of LivingCreature.
R2: Use the call to the parent (super) where appropriate.
R3: Do not duplicate code from the parent in the child.
R4: Define the abstract superPower() method in each of the child classes.
R5: Create a drive that tests the child classes as described in the narrative.
R6: Document the classes and drivers thoroughly.
R7: Create a separate documentation file (Word or PDF file) to describe your classes in detail.

Security Considerations

Classes should validate method parameters, use private instance data items, and only provide accessors and mutators where needed and necessary. Child classes and parent classes can share data and still include proper security. Pay attention to the access modifier (private, protected, and public) and use them properly.