Gustavo Torrealba

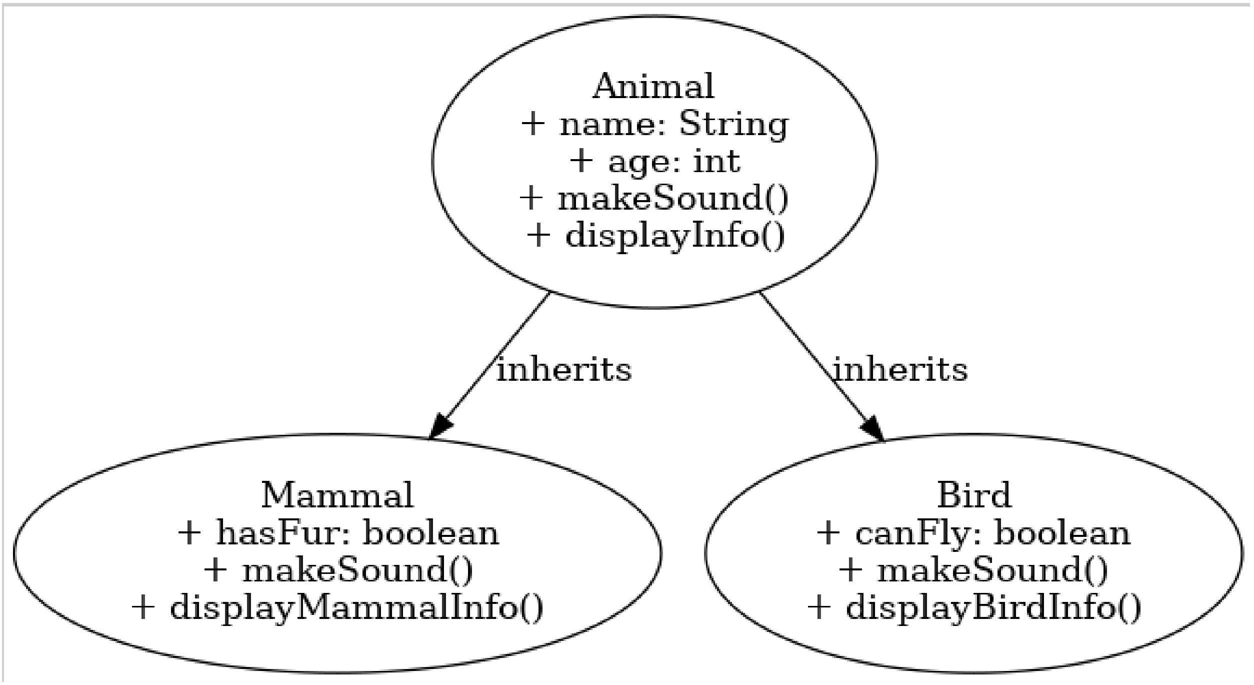
CS6301 Homework #3

Class Diagrams

Code utilized – Animal.java

Class diagram generated using graphviz – AnimalDiagram.py

Resulting Class Diagram:



Code Analysis:

Firstly, definitions – what are encapsulation and modularity?

- Encapsulation: a programming concept that bundles data and methods into a single unit, hiding the internal details of the data. It restricts/protects direct access to some of an object's components, using access modifiers like public/private/protected to hide certain details. It sort of acts like a protective black box.
- Modularity: the concept of breaking down a system into smaller, independent parts which may be re-combined to form a whole. It's a design principle which simplifies a system's operation by breaking it down into smaller components.

Evaluation:

- The code in Animal.java follows good principles of ***<u>Encapsulation</u>***, as it hides away parameters that are preferred to be left immutable such as the animal name and age by using the protected classifier. These attributes are still accessible via the sub-classes Mammal and Bird, but not directly from outside of these. The attributes can be viewed via the displayInfo() method, but not modified from outside of the class, which would be possible if their access modifier was public instead of protected. Similarly, the classes Mammal and Bird have private attributes in hasFur and canFly, respectively, also preventing direct modification from outside the class. Furthermore, all of the behavioral methods follow good encapsulation standards as they allow access to these Animals' properties without exposing the raw data to modification. For example, if you had a Mammal class, you could use the displayInfo() method to verify that it has fur, but you can't modify the hasFur property itself.
- As for ***<u>Modularity</u>***, the code clearly separates the classes of Animal, Mammal, and Bird, while providing with each class with unique attributes and behaviors. Mammal and Bird inherit from Animal, which eliminates the need to declare redundant attributes (name and age) for these sub-classes, and the sub-classes also serve unique purposes, as each type of animal makes a different noise.