***Java Directions****: If you are doing this as a Java exercise, you will need to create a class file for each class and test it in a main method. Sample Java main method follows:*

*Public static void main(String[] args) {*

*}*

**The classes you Use in this exercise**

You should have already the classes listed in the diagram below. We will add a few classes in this exercise in order to practice Java:

* Abstract Classes
* Polymorphing
* Interfaces

*Class names are bold, instance variables are the not bold*

**Important Note:** You should have already created the Animal, Mammal, Canine, and Feline classes!!!

**Existing Classes**

**Animal Class**

* Properties/attributes
  + Gender
* Methods
  + getGender and setGender

***Continued on Reverse***

**Mammal Class *(Subclass of Animal)***

1. This class is a subclass of the Animal class.
   1. Attributes:
      1. age
      2. weight
   2. Methods:
      1. getAge()
      2. setAge()
      3. getWeight()
      4. setWeight()
      5. move() - Just print “*Mammal Moves*” in this method.
      6. makeNoise() - Just print “*Mammal Makes a Noise*” in this method.
2. Constructor:
   1. Accepts the values gender, age, and weight as parameters and sets the appropriate attributes.
   2. Prints the values of gender, age, and weight using the appropriate “get” methods.

**Feline class *(Subclass of Mammal)***

1. This class is a subclass of the Mammal class.
   1. Attributes:
      1. breed
      2. isDangerous
   2. Methods:
      1. Getters and setters for all the attributes above.
2. Constructor:
   1. Accepts the values gender, age, weight, breed, and isDangerous as parameters and sets the appropriate attributes.
   2. Prints the values of age, breed, and isDangerous, using the appropriate “get” methods.
3. The makeNoise() method that:
   1. Prints “Feline says purr”
   2. Calls the makeNoise() method of the superclass.

**Canine Class *(Subclass of Mammal)***

1. This class is a subclass of the Mammal class.
   1. Attributes:
      1. breed
      2. isDangerous
   2. Methods:
      1. Getters and setters for all the attributes above.
2. Constructor:
   1. Accepts the values gender, age, weight, breed, and isDangerous as parameters and sets the appropriate attributes.
   2. Prints the values of age, breed, and isDangerous, using the appropriate “get” methods.
3. The makeNoise() method:
   1. Prints “Canine says howl”
   2. Calls the makeNoise() method of the superclass.

**New Classes To Create**

**The PolyTest() Class**

This class receives no parameters into its constructor and contains no instance variables. We will use it to practice one of the best uses of polymorphism.

We will create instances of the Feline and Dog classes, add them to the same ArrayList() and modify (and print) the animal names. Everything for this example can be done in the main method or placed in an appropriately named method.

1. Create a class named PolyTest().
   1. Properties/attributes
      1. None
   2. Methods
      1. None
   3. Constructor: Prints the following:
      1. "Instance of: Animal created."
2. Define a main method. Inside the main method:
   1. Create two instances of the Feline() class using “Animal” as the type and cat1 and cat2 as the names of the instance variables.

Animal cat1 = new Feline();

* 1. Create two instances of the Canine() class using “Animal” as the type and dog1 and dog2 as the names of the instance variables.
  2. Define an ArrayList() named animalList that uses Animal as the type.
  3. Add your four objects to the list (cat1 cat2 dog1 dog2)
  4. Print the message "\n \*\*\* Polymorphic test \*\*\* "
  5. Print the message " \* Printing all genders BEFORE MODIFICATION\* "
  6. Use a for loop to print the gender of all items in the animalList.
  7. Use a for loop to set the gender of all items in the animalList. To “unknown”.
  8. Print the message " \* Printing all genders AFTER MODIFICATION\* "
  9. Use a for loop to print the gender of all items in the animalList.
  10. **Master Coders**
      1. Move all the print logic into a method named printAnimalGenders(). It will receive an ArrayList of the type Animal as a parameter.
      2. How can you call this method?
      3. Do we need to add a modifier to make this work?

**The InterfaceTest() Class**

**The AbstractTest() Class**