**LAB ASSIGNMENT**

**OldMacDonald[[1]](#footnote-1)**

**Background:**

In this lab, we will use the well-known song 'Old MacDonald Had a Farm' to learn about Inheritance and Polymorphism. Old MacDonald had a farm and several types of animals. Every animal shared certain characteristics. They each had a type (such as cow, chick or pig) and each made a sound (moo, cluck or oink).

**Notes:**

For those unfamiliar with it, a version of the *Old MacDonald* song is found at <http://www.scoutsongs.com/lyrics/oldmacdonald.html>.

**Create a new project called Polymorphism. In this project, add the Animal, Cow, Farm and OldMacDonaldFarm.java files.**

**Assignment:**

1. Once we know what it takes to be an Animal, we can define new classes for the cow, chick and pig that extend the Animal abstract class. The Cow class shows meeting the minimum requirements to be an Animal.
2. Using the Cow class as an example, create classes for the chick and the pig. Also complete the OldMacDonaldFarm class to verify your work so far.
3. Complete the Farm class to test all your animals. You will need to change your *OldMacDonald.java* code to create an object of type Farm and then to invoke its animalSounds method.
4. As it turns out, the chick seems a little confused. Sometimes it makes one sound, when it is feeling childish, and another when it is feeling more grown up. Its two sounds are "cheep" and "cluck". Modify the *Chick.java* code to add a second constructor that sets a boolean, called hasTwoSounds, for the chick to indicate whether the chick makes one or two sounds. The getSound() method returns either sound, with equal probability, if there are two sounds available. You will also have to modify your *Farm.java* code to construct the Chick with two possible sounds.
5. Finally, it also came to pass that the cows received personal names, such as Elsie. Create a new class, NamedCow, which extends the Cow class, adding a constructor, a field for the Cow's name, and a new method: getName. Shown below is the final *Farm.java* code to exercise all your modifications:

**public class** Farm

{

private ArrayList <Animal> myFarm;

public Farm() {

myFarm = new ArrayList <Animal>();

myFarm.add(new Cow());

myFarm.add(new Chick());

myFarm.add(new Pig());

myFarm.add(new NamedCow(“Elsie”));

}

public void animalSounds(){

Animal temp;

for(int i = 0; i < myFarm.size(); i++){

temp = myFarm.get(i);

System.out.println(temp.getType() + " goes " + temp.getSound());

}

NamedCow named = (NamedCow)myFarm.get(3);

System.out.println(named.getName());

}

Make sure that you understand what you've just accomplished. Having an ArrayList of Animal objects and then having the getSound() method dynamically decide what sound to make is known as *polymorphism*. This is also known as *late binding* because it wasn't known until run-time that a.get(1), for example, really was a Chick object.

You started with an *abstract class* for an Animal and then used the keyword **extends** in making the three types of animals. Then you created a specialized version of the Cow, a NamedCow, using the keyword **extends**. This illustrates the concept of *inheritance*. The NamedCow had all the attributes and methods of the Cow and then added some: a new field and a new method to access the cow's name.

**Instructions:**

Your lab assignment should consist of the following 7 files:

*Animal.java* – abstract class

*Chick.java*, *Cow.java*, *Pig.java* – extensions of the Animal abstract class

*NamedCow.java* – subclass of the Cow class

*Farm.java* – collection of Animal objects

*OldMacDonaldFarm.java* – testing class

Turn in

*Chick.java*, *Pig.java* – extensions of the Animal abstract class

*NamedCow.java* – subclass of the Cow class

*OldMacDonaldFarm.java* – testing class

1. This lab was adapted, with gratitude, from a lab developed by Roger Frank of Ponderosa HS, in Parker, CO. [↑](#footnote-ref-1)