# LAB EXERCISE

## OldMacDonald

**Background:**

Old MacDonald had a farm and several types of animals. Every animal shared certain characteristics: they had a type (such as cow, chick or pig) and each made a sound (moo, cluck or oink). An Interface defines those things required to be an animal on the farm.

```
public interface Animal
{
  public String getSound();
  public String getType();
}
```

In this lab, we use Old MacDonald's Farm to learn about Inheritance and Polymorphism.

**Notes:**
This lab is adapted with gratitude from a lab developed by Roger Frank of Ponderosa HS, Parker CO.

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

**Assignment:**

1. Once we know what it takes to be an `Animal`, we can define new classes for the cow, chick and pig that implement the `Animal` interface. Here is a `Cow` class meeting the minimum requirements to be an `Animal`.

```
class Cow implements Animal
{
  private String myType;
  private String mySound;

  Cow(String type, String sound)
  {
    myType = type;
    mySound = sound;
  }

  public String getSound() { return mySound; }
  public String getType() { return myType; }
}
```

2. Implement classes for the chick and the pig. Also complete the test program below to verify your work so far:

```
class OldMacDonald
{
  public static void main(String[] args)
  {
    Cow c = new Cow("cow", "moo");
    System.out.println( c.getType() + " goes " + c.getSound() );

      // < your code here >
  }
}
```

3. Create a complete farm to test all your animals. Here is the *Farm.java* source code.

```
class Farm
{
  private Animal[] a = new Animal[3];
  Farm()
  {
    a[0] = new Cow("cow","moo");
    a[1] = new Chick("chick","cluck");
    a[2] = new Pig("pig","oink");
  }

  public void animalSounds()
  {
    for (int i = 0; i < a.length; i++)
    {
      System.out.println(a[i].getType() + " goes " + a[i].getSound());
    }
  }
}
```

   You will need to change your *OldMacDonald.java* code to create an object of type `Farm` and then to invoke its `animalSounds` method.

4. Turns out, the chick is a little confused. Sometimes it makes one sound, when she is feeling childish, and another when she is feeling more grown up. Her two sounds are "cheep" and "cluck". Modify the *Chick.java* code to allow a **second constructor** allowing two possible sounds and the `getSound()` method to return 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 get a personal name, like Elsie. Create a new class, `NamedCow`, that extends the `Cow` class, adding a constructor, a field for the `Cow`'s name, and a new method: `getName`. The final *Farm.java* code to exercise all your modifications is shown here:

```
class Farm
{
  private Animal[] a = new Animal[3];
  Farm()
  {
    a[0] = new NamedCow("cow","Elsie","moo");
    a[1] = new Chick("chick","cheep","cluck");
    a[2] = new Pig("pig","oink");
  }

  public void animalSounds()
  {
    for (int i = 0; i < a.length; i++)
    {
      System.out.println(a[i].getType() + " goes " + a[i].getSound());
    }
    System.out.println("The cow is known as " +
                       ((NamedCow)a[0]).getName());
  }
}
```

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

   You started with an *Interface* for an `Animal` and then used the keyword **`implements`** 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:**

1. Develop and test the Old MacDonald Farm classes a described in the Assignment section above.

2. Your lab assignment should consist of the following 7 files:

   *Animal.java* – interface
   *Chick.java*, *Cow.java*, *Pig.java* – implementations of the Animal interface
   *NamedCow.java* – subclass of the Cow class
   *Farm.java* – collection of Animal objects
   *OldMacDonald.java* – testing class

3. Display your source code and run output. Call your instructor to your workspace for scoring.