Instructions:

Interface:

Create an interface called ITrainable which contains the following:

Dictionary<string, string> Behaviors{ get; set; }

string Perform(string signal);

string Train(string signal, string behavior);

Here is an explanation of each of the signatures above; (Note: The implementations are described below but should be in the class that implements the interface, not the interface itself):

Dictionary<string, string> Behaviors{ get; set; }

A dictionary property named Behaviors to be implemented in classes that implement ITrainable

The key (a string) is the signal the trainer/user will use to ask the animal to perform a behavior.

The value (a string) is the behavior the animal will perform when it sees the signal.

string Perform(String signal);

A method named Perform to be implemented in classes that implement ITrainable.

Needs a string parameter

Fetch the behavior from the Behaviors dictionary based on the signal.

Should return a string describing how the animal performed the behavior in response to the signal.

string Train(String signal, string behavior);

A method named Train to be implemented in classes that implement ITrainable.

Needs a string parameter that holds the signal sent by the user.

Will add the behavior to the Behaviors dictionary using the signal as the key

Should return a string describing how the animal has been trained and will respond to the specified signal.

Base Abstract Class:

Create a base abstract class named Animal which contains the following properties and fields:

A property named Species. This will hold the animal’s name (i.e. dolphin)

A field named \_foodConsumed to keep track of how much food the animal has consumed.

A field named \_treat used to store the name of the food this animal likes to eat as a treat. (like fish for dolphins, rats for snakes). This will need to be set to protected access since it will be used in subclasses to set the value unique to each animal.

The following methods:

An Eat method

Tracks how much food has been consumed.

If the animal eats over 4 times it should trigger the Poop() method and reset the \_foodConsumed field.

It should return a string describing how the animal ate the food that looks like this: "The dolphin ate a herring”

A MakeNoise method

This will be overridden later and should be virtual.

This should return a string. This will be overriding later but should return a default string. (If your code is correctly created you should not see the default in your working application.)

A Poop method

This should not return anything.

This should write out to the console that the animal has pooped. For ex: "The dolphin pooped!”

Animal Classes:

You will need to implement at least 6 animal classes all should inherit from the Animal base class.

The constructors will need to initialize the Species and Trainable properties as well as the \_treat field.

An overriding method named MakeNoise that returns a string that describes the animal making noise. Ex: “A gargled roar comes from the alligator.”

Some animals are trainable (dogs, dolphins, even cats) and some aren’t (snakes, fish etc) and your application should reflect that. At least half of the animals should implement the ITrainable interface. This will mean they will be required to implement the properties and methods as described above in the Interface section.

The Perform method should return a string similar to: "After you signaled clapping hands the otter will now perform the 'jump through the hoop' behavior."

The Train method should collect the signal and behavior strings and input them into the Behavior dictionary and return a string such as: "The dolphin learned to slap it’s tail when you point to the sky."

Contain a Behaviors property that will hold the signals and behaviors the given animal can perform based on what the user enters.

Zookeeper Application:

Your main program application class should have the following:

A List object which will contain all the animals you create.

A method which welcomes the user and adds instances of each animal into the List object.

Your application should list the animals you’ve initialized, specify which are trainable, and give the user the option of selecting one.

Upon selecting an animal it will announce which animal was selected and then provide a menu for activities the user can do with the selected animal:

Training:

If the user selects Training, the program should ask the user what behavior they would like to teach the animal and what signal they will associate with the behavior. The program then stores this information and reprints the menu:

Feeding the Animal:

For feeding an animal a treat (this is for all animals, not just the trainable ones) it produces the following result, printing the string produced by the Eat() method in the Animal base class:

Get the animal to perform a behavior:

When signaling the animal, if the animal is trainable, then the program should ask the user for the relevant signal. If the animal is not trainable a message should show to the user: “The animal you selected is not a trainable animal. Please select a different activity or exit to select a different animal.”

If the signal is in the dictionary it should respond. (If it is not a message should be shown “The dolphin does not know this signal. Train it to recognize this signal and associate it with a behavior.” )

Make noise:

When this option is selected the animal should make a noise:

Or the user can select a different animal.