Lab – Categories and Protocols

OVERVIEW

In this lab you'll declare a custom protocol to implement the **Delegate** pattern. You'll also use a category to add a method the **NSArray** class.

PART 1

1. In the **Objective-C Labs** project you created in the previous lab exercise, add a subclass of **NSObject** named **Dog** with the following features:

    1.1. An instance variable of type **NSString** * named **_name**.

    1.2. A custom initializer with the following signature:

    ```
    - initWithName:(NSString *)name
    ```

    1.3. A **name** accessor method.

    1.4. Three methods that take no arguments and return **void**, each of which uses **printf** to print a message on the console. The message should consist of the dog's name followed by a colon and a space, and then the string described below, followed by a newline character.

    - A **growl** method that prints **Grrrrrr!**.
    - A **bark** method that prints **Woof! Woof! Woof!**.
    - A **wagTail** method that prints **[Wags tail.]**.

    1.5. An overridden **description** method that returns the dog's name.

    1.6. A method named **doorbellDidRing** that calls the **growl**, **bark**, and **wagTail** methods.

2. Add a subclass of **Person** named **DogOwner** with the following features:

2.1. An instance variables of type **NSMutableArray *** named **_dogs**.

2.2. A private method named **mutableDogs** that lazily initializes and returns the **_dogs** instance variable.

2.3. A public method named **dogs** that returns the value returned by **mutableDogs**.

2.4. A public method named **addDogs:** that takes an argument of type **NSArray ***, and adds its content to the mutable dogs array.

2.5. An overridden **description** method that prints the owner's full name, followed by the a description of each of the owner's dogs.

3. Add a test case named **DogOwnerTests.m** that includes the following:

3.1. An instance variable of type **DogOwner *** named **_owner**.

3.2. A **setUp** method that initializes **_owner** with an instance of **DogOwner**, and then adds three instances of **Dog** named **Bowser**, **Woofsie**, and **Spot** to the owner's dogs array.

3.3. A **testPart01** method that uses **NSLog** to print the owner's description, and then sends **doorbellDidRing** to each of the owner's dogs.

3.4. Build and run to make sure that the descriptions print as expected, and that each dog prints a growl, bark, and wag tail message.

PART 2

1. Add the following features to the **Dog** class:

    1.1. A protocol named **DogDelegate** that declares three required methods, all of which take an instance of **Dog** as their only argument. The first method, **dogDidHearDoorbell:** should return **void**; the other two, **dogShouldBark:** and **dogShouldWagTail:**, should return **BOOL**.

    1.2. An **_delegate** instance variable of type **id<DogDelegate>**, and a corresponding pair of accessor methods.

    1.3. A **sit** method that prints a message similar to **wagTail**, only with **[Sits.]** as its text instead of **[Wags tail.]**.

    1.4. Modify **doorbellDidRing** as follows:

    - After calling **growl**, send a **dogDidHearDoorbell:** message to the dog's delegate.
    - Add logic to call **bark** only if the dog's delegate is **nil**, *or* if the delegate's **dogShouldBark:** method returns **YES**. Add similar logic before the call to **wagTail**.

2. Add the following features to the **DogOwner** class:

    2.1. Make **DogOwner** conform to the **DogDelegate** protocol.

    2.2. Implement **dogShouldBark:** and **dogShouldWagTail:** to return **NO**.

    2.3. Implement **dogDidHearDoorbell:** to send a **sit** message to **Bowser** and **Woofsie**.

3. In **DogOwnerTests**, write a test method named **testPart02** that does as follows:

    3.1. Make the owner object **Bowser** and **Woofsie**'s delegate.

    3.2. Send **doorbellDidRing** to each dog.

    3.3. Build and run. Make sure that **testPart01** still works as it did previously, and that the output of **testPart02** verfies that **Bowser** and **Woofsie** growl and sit (but don't bark), while **Spot**'s behavior is unchanged.

PART 3

1. Add a **LABAdditions** category to the **NSArray** class. The category should declare and implement a method named **LAB_fancyDescription** with the following behavior:

    1.1. Print the name of the receiving class, followed by a count of its elements.

    1.2. For each element, print the object's class, followed by its description.

2. Write a unit test method named **testPart03** that sends a **LAB_fancyDescription** to the owner's array of dogs.

3. Build and run, and verify that the output is as expected.

