Lab 6

In this lab, we will model Man’s (and Woman’s) Best Friend™ (and practice with String and loops).

Those Tricky Dogs

Open a new BlueJ project and create a class Dog. Let’s start by giving it a single String attribute called name to store the Dog’s name.

Give the Dog class a constructor that takes that dog’s name as an argument.

Then create two methods to access the name. The method getName takes no arguments and returns the name. The method printName takes no arguments, returns nothing, and prints the name on its own line.

Now let us add some tricks for Dog objects to perform. Start by creating a method sit. This should take no arguments and return nothing, but print the name of the Dog followed by “sits”. Thus, asking a dog named Spot to sit would print the following:

Spot sits

Once you have completed sit, add two more tricks of your choosing. (Common tricks like layDown are obvious choices, but consider more interesting tricks like writeJavaProgram; without Dog programmers, where will we get smash hits like “World Sniffing Championships III: This time it really smells”.) Each trick should print a relevant message to the screen. Make the messages appealing to read rather than using the method name. The user of your program does not care that one of your methods is called standOnHeadAndWhistleDixie, but they are interested in seeing a message that this is what the dog is doing.

Now let us add an ability to see what tricks the Dog has performed. Add an attribute called trickList that is an ArrayList of String objects. The constructor should use new to create this object. Then each trick method should add a String describing the trick performed to it. You’ll also need to add the line

import java.util.ArrayList;

to the top of your file (outside the class).

Now we need something to do with the list of tricks. Add a method called printTrickList (no arguments and no return value) to print a list of tricks performed. More specifically, if the list is empty, print a message like “dogName has not performed any tricks yet”, where dogName is the name of the Dog. (Recall that ArrayList has a method size, which you can use to see if it is empty.) Otherwise, print a message like “dogName has performed the following tricks:”, followed by the list of tricks, each indented (just print some spaces before each trick). After this, you should be able to get a sequence of messages like

Spot has not performed any tricks yet
Spot sits
Spot sleeps
Spot reads the paper
Spot has performed the following tricks:
  sit
  sleep
  read paper
Remember that you can do a foreach loop with code like

```java
for(String s : trickList) {
    body
}
```

This will execute the statements in `body` once for each string in `trickList` (with `s` taking the value of the current string).

These dogs are already pretty impressive, but think of how much more they’ll be able to do by working together. Create a new class `Pack` to represent a pack of dogs. This class should have an attribute `members` that is an `ArrayList` of `Dog` objects. Also give it an `int` attribute `leaderIndex` that is the index within `members` of the pack’s leader.

Give this class a constructor that takes a `Dog` object. The constructor should create the `members` attribute, add the leader to `members`, and set `leaderIndex` to zero (since that’s the index at which the leader will be placed).

Next, write a method `getLeaderName` that takes no arguments and returns the name of the pack leader. (The `getName` method of `Dog` will be useful here.) Test your method by creating a `Dog`, making them leader of a `Pack`, and then calling the `getLeaderName` method.

A pack containing only the leader is not much of a pack so we need a way to add other dogs. Add a method `addMember` that takes a `Dog` and adds it to the pack.

Now, so we can tell who is in the pack, write a method `printPack` that prints the members in the following form:

```
The pack contains:
  Spot
  Fido
  Rover
```

With larger packs will inevitably come struggles for leadership. Add a method `newLeader` that takes an integer, interpreted as the index of the desired new leader of the pack. Check that the value given is in range and not the current pack leader. If these conditions are met, change the `leaderIndex` attribute and print a message like

```
Rover deposes Spot as the leader of the pack
```

After getting this message, call `getLeaderName` to make sure that the pack’s leadership really has changed.

The pack will get a chance to work together, however. Add a method corresponding to one of your tricks that will make the entire pack perform that trick. For example, a `allSit` method would make each pack member perform the `sit` trick. Then add a method `allPrintTricks` method that makes all the pack members print the list of tricks they’ve performed. Have the dogs perform their tricks individually and together, periodically printing them to make sure everything is working.

That’s it! (But feel free to continuing training the dogs as individuals and packs.)