Lab 14

ArrayLists

Objectives of this Lab:

1. This lab will explore the ArrayList data structure as well as the following aspects of OO programming in Java:
    a. Constructing objects
    b. Access modifiers (private, public) and accessing/modifying instance variables
    c. Method usage
    d. Client objects
    e. toString()
    f. equals(Object other)

Introduction

The Java `ArrayList` is a dynamic array-like data structure that can grow or shrink in size during the execution of a program as elements are added/deleted. An Array on the other hand, has a fixed size: once we declared it to be a particular size, that size cannot be changed.

To use an `ArrayList`, you first have to import the class:

```
import java.util.ArrayList;
```

You can then create a new ArrayList object:

```
ArrayList<Object> listTest = new ArrayList<Object>();
```

The Java API has a list of all the methods provided by an ArrayList. See: Java ArrayList API.

Lab Assignment

Please complete this lab individually. You are always welcome to bring any lab material and questions to office hours. We will use **Animal.java** and **GuinnessBook.java**. Please download the following files:

- animalList.txt
- Animal.java
- GuinnessBook.java

Previously, you have used arrays to store and manipulate collections of primitives and objects. However, Java arrays can only represent collections of a fixed size. This is a limitation as the programmer often does not know how many items will need to be stored in advance.
Many programming languages, including Java, provide programmers with array-like data structures that are *resizable*. In Java, this data structure is ArrayList. An ArrayList internally maintains an array of data that is resized to accomodate additional elements. ArrayList also allows the programmer to add/remove items at arbitrary indices and even find the index of a particular item.
Complete the implementation of **Animal.java**, by adding the following methods:

- Getters and setters for name and topSpeed. Print an error if topSpeed is over 70 or below 0 and leave the speed alone.
- A constructor that takes in a String for the name and an int for the topSpeed. Use the setters to set the instance variables.
- a toString() method that returns a string with that animal's information. EX: Name: elephant Top Speed: 25
- an equals(Object other) method that returns True if two animals have the same speed within 2mph, and False otherwise (recall that equals takes in an instance of Object rather than Animal, in order to override the default implementation of equals).

When implementing constructors, getters/setters, toString or equals methods you can use Eclipse to help you write those methods - under the "Source" tab of Eclipse there are options for generating stubs for those methods.

In **GuinnessBook.java**: Complete the code for the constructor. You just need to add code that adds the animals to the `landAnimals` ArrayList.

The `toString()` method. This can make use of Animal's `toString()` method. It should return a string representation of all the elements in the arrayList. EX:

```
Name: giraffe Top Speed: 32
Name: pronghorn Top Speed: 61
Name: reindeer Top Speed: 32
```

Testing

Test your code in the main method.

Follow the instructions in main and testGuinnessBook.

Turn In

Show your work to your TA before submitting to GitHub. Submit all of your files with correct headers to GitHub here: https://classroom.github.com/a/6VcaTWrA