Lab 3: A Simple Lookup Table

Due: 11:59PM 9/12/12

Overview

For this assignment, we want to design an object that provides a simple mapping from `String` keys to `String` values – in other words, a *lookup table*.

For example, if I wanted to associate people with their favorite foods, I might do something like this:

```
LookupTable lt = new LookupTable();
lt.set("Craig", "hamburger");
lt.set("Gary","escargot");
lt.set("Frances", "horse livers");

// .....

System.out.println("Frances likes to eat " + lt.get("Frances"));
```

We are going to implement this `LookupTable` in a very specific way. You may already be familiar with containers in the standard library that provide mappings like this. **Do not use them.**

Approach

Most of your `LookupTable`'s public interface is demonstrated in the sample code above. For debugging purposes, you should also provide a `void printAll()` method that simply iterates over all key,value pairs and prints them out to `System.out`. Note, this violates our design directive of *minimizing coupling*, but it's the easiest way to enable this testing behavior for now.

In the example above, a call to `lt.printAll()` could cause the following to appear in the console:

```
Craig -> hamburger
Gary -> escargot
Frances -> horse livers
```

To summarize, the public interface of `LookupTable` should look like this:

- `void set(String key, String value)`

- `String get(String key)`
- `void printAll()`

To implement this, we will introduce a second new class that represents a **single** key, value pair. One possible good name for this class is `StringPair`. This class will need accessors and mutators for its key and value instance variables Make sure you follow good object-oriented design practices regarding encapsulation, even with this small helper class.

Back in your `LookupTable`, you will simply have a private `ArrayList<StringPair>` instance variable. Your `get` and `set` methods work exclusively on this `ArrayList`. Your `printAll` should iterate over the entire list and print out the contents of each `StringPair`.

`get()` should iterate over all the contents of your list and if a list element's `key` equals the desired `key`, then return the corresponding `value`.

`set()` should not allow duplicates! One simple approach is this:

1. iterate through the list of `StringPair`'s, and if you find one that matches the desired `key`, call a method that sets a new value for it and returns.
2. If you get to the end of the list without finding the `key`, add a new `StringPair` to the end of the list.

Driver Code

I am not interested in your driver code as much as your `LookupTable`. You do, however, need to demonstrate that your table works correctly, so provide a driver class with a `main` method that simply associates various keys with various values (think of a theme if you'd like, such as people and their favorite foods, as above), and make sure to demonstrate that `get` and `printAll` work as expected.

Documentation

Make sure that your lookup table has appropriate javadoc comments, and include rendered documentation with your source code.

Turn In

Create a folder named `YourName_1110_Lab3` containing all of your source code and rendered documentation. Zip this up and submit it to Blackboard by the due date.