Lab 12: Sequential-Search Symbol Tables

In this lab you will create a `SequentialSearchSymbolTable`. You will use it to implement a very basic restaurant program.

The file `menu.tsv` contains the menu of a certain restaurant. Each line represents one item on the menu. A line contains the item’s name, followed by a tab, and then the cost of the item. (The cost is in cents. This is to avoid the rounding errors inherent in using doubles.)

Remember that an SSST is able to automatically resize itself, in order to hold the necessary key-value pairs. It is your choice how to handle this. You could simply make an `ArrayList` of each one, or you could handle the growing and shrinking yourself. (In this case, you’ll have two arrays. When the size is about to exceed the capacity, increase them by 10 or so. When the size becomes 20 less than the capacity, decrease them by 10. If you take this route, you should have a final variable called `STEP_SIZE`. Remember to avoid hard coding!)

The first object you make must be the `SequentialSearchSymbolTable` object. This will use generics: `K` representing the key, and `V` representing the value. Your class should be declared like this:

```java
class SequentialSearchSymbolTable<K,V> {
```

It should have nothing in it to suggest that it was made for a specific application (i.e. no restaurant-specific stuff, just SSST stuff). Your public methods must be as follows:

- `SequentialSearchSymbolTable()`, the constructor.
- `void clear()`, which deletes all key-value pairs from the SSST.
- `boolean delete(K key)`, which removes a key and its corresponding value. If the key isn’t in the SSST, do nothing and return `false`. If it is, delete it and return `true`.
- `V get(K key)`, which returns the value to which the key maps. If the key is not contained in the SSST, return `null`.
- `int getSize()`, which returns the number of key-value pairs there are in the SSST.
- `void put(K key, V value)`, which puts the key-value pair into the object. If key already mapped to something else, the old value should be replaced with the new one.
- A unit-testing `main()` method to test all the above methods, in a variety of situations.

You may have other private methods if you wish.

Then, when the object is created, make your restaurant program. It will load the file in, and store it into an SSST that maps a menu item to its cost. Enter a loop in which you get queries from the user:

- If the user enters a food item, it will add that item’s cost to the user’s subtotal, and display the subtotal.
If the user hits enter, it will calculate tax (10%) and tip (15%) rounded to the nearest cent, print them, add them to the subtotal, print the total, and exit. All money amounts should be formatted correctly (e.g. print $4.00, not 400).

If the user enters anything else, it will apologize and say that it didn’t understand.

Hint: If you’re having trouble displaying prices properly, look up the printf() function in the Java PrintWriter specs. PrintWriter is the kind of object that System.out and System.err are. You will see the familiar print() and println() functions, as well as printf(). The arguments to printf are a String containing special formatting symbols (that begin with %), followed by arguments that are the numbers to print out. For example, this:

```java
System.out.printf("$%.2d", dollars, cents);
```

Will print out dollars and cents as you expect—so long as you’ve defined two ints named dollars and cents. Here %.2d means print out some integer, and %.02d means print out an integer padded with 0s so that it’s 2 digits long.

The two classes you turn in should be called Restaurant and SequentialSearchSymbolTable.

<table>
<thead>
<tr>
<th>keys</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;apple&quot;</td>
<td>&quot;red&quot;</td>
</tr>
<tr>
<td>&quot;grape&quot;</td>
<td>&quot;purple&quot;</td>
</tr>
<tr>
<td>&quot;orange&quot;</td>
<td>&quot;orange&quot;</td>
</tr>
<tr>
<td>&quot;banana&quot;</td>
<td>&quot;yellow&quot;</td>
</tr>
<tr>
<td>&quot;lemon&quot;</td>
<td>&quot;yellow&quot;</td>
</tr>
<tr>
<td>&quot;berry&quot;</td>
<td>&quot;red&quot;</td>
</tr>
</tbody>
</table>