CSC 20 Chapter 11: Collection classes / Iterators

We have seen `ArrayList`. It holds references to objects for us in an ordered list. It is one of many "collection" classes in Java.

There are three basic type of collections.

List. A list is an ordered collection of objects. You access elements of the list by index. Items may repeat in the list.

Set. A set is an *unordered* collection of objects, so no methods take or return an index. You can add, remove, query, and iterate over the items in the set.

Map. A map associates one object with another, sort of like an array associates an integer index with an element.

[Look at javadoc for each of these.]

Note that these are Interfaces. You can declare variables of these types and access all of the listed functionality, but the implementation must be by a class that implements the interface.

`ArrayList` implements `List` using an array (meaning add/remove at the front is O(n) but all access are O(1)).

`LinkedList` implements `List` using a linked structure (meaning add/remove at front is O(1) but worst-case accesses are O(n)).

`TreeMap` and `TreeSet` structure their data in a binary tree structure, using `compareTo` to decide which items are on which side.

`HashMap` uses a mathematical "hash" to map objects to objects. `HashSet` uses `HashMap` to do it's work.

Before choosing which of these to use, know the strengths and weaknesses.

Let's write a program that reads tokens from the keyboard and stops as soon as it sees the same token twice.

```
import java.util.*;

class Repeat {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Set<String> tokens = new TreeSet<String>();
        String tmp = in.next();
        while ( ! tokens.contains(tmp) ) {
            tokens.add(tmp);
            tmp = in.next();
        }
        System.out.println(tmp + " was seen twice");
    }
}
```

If I wanted to stop only when I saw some token 3 times, I'd have to change my strategy. It's no longer enough to know *if* I've seen a token before, but now I need to know *how many* times I've seen the token before.

```
import java.util.*;

class Repeats {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
```

```
        Map<String,Integer> tokens = new TreeMap<String,Integer>();
        String tmp;
        int seen = 0;
        while ( seen != 3 ) {
            tmp = in.next();
            if (tokens.containsKey(tmp)) {
                seen = tokens.get(tmp)+1;
            } else {
                seen=1;
            }
            tokens.put(tmp,seen);
        }
        System.out.println(tmp + " was seen thrice");
    }
}
```

Utilities in Collections class

The collections class has some useful methods for manipulating collections.

`Collections.sort(cxn)` will rearrange the items in `cxn` to be in sorted order.
`Collections.shuffle(cxn)` will rearrange the items in `cxn` to be in random order.

[Show javadoc.]

Iterators

If you want to visit each element in a collection, use an iterator. If `list` is a collection of Strings, the following will access each:

```
Iterator<String> itr = list.iterator();
while (itr.hasNext()) {
    System.out.println(itr.next());
}
```

Here's the last program, outputting the number of times each token was seen.

```
import java.util.*;

class Untitled {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Map<String,Integer> tokens = new TreeMap<String,Integer>();
        String tmp = "";
        int seen = 0;
        while ( seen != 3 ) {
            tmp = in.next();
            if (tokens.containsKey(tmp)) {
                seen = tokens.get(tmp)+1;
            } else {
                seen=1;
            }
            tokens.put(tmp,seen);
        }
        System.out.println(tmp + " was seen thrice");
        Iterator<String> itr = tokens.keySet().iterator();
```

```
        while (itr.hasNext()) {
            tmp = itr.next();
            System.out.println(tmp + " was seen " + tokens.get(tmp) + " time(s).");
        }
    }
}```
```