Anagram Groups Problem

- Given a dictionary of words.
- Find and print all anagram groups of size two or larger.
- Print them in increasing order of size.
- An anagram group is a collection of words that are anagrams of each other.
- Example: \{bard, brad, drab\} is an anagram group of size three.

Java Collections Framework

- The Java collections framework (JCF) is a set of classes and interfaces that implement commonly reusable collection data structures. (Wikipedia)

- A collection — sometimes called a container — is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. (Oracle Java Tutorials)
Examples of JCF Interfaces
(Oracle Java Tutorials)

![Diagram showing Collection, Map, Set, List, Queue, SortedSet]

Examples of JCF Classes

- **ArrayList**: (Similar to our FlexibleArray.)
- **LinkedList**: (Similar to our ListDL.)
- **TreeSet**: (Similar to our BSTree.)
- **HashSet**: (Alternative implementation of set container.)
- **TreeMap**: (Similar to our AssociativeMapBST.)
- **HashMap**: (Alternative implementation of map container.)
Bucket Sort

- Partition a set of objects into subsets, which we will call “buckets”.
- Each bucket is labeled with a key.
- The key represents a property shared by the contents.
  - E.g., Partition students in class by birth month.
  - E.g., Partition words in dictionary by number of characters.

MultiMap: An Interface for Classes Implementing Bucket Sort

```java
public interface MultiMap<K extends Comparable<K>, V> {
    void put(K key, V value);
    List<V> get(K key);
    Set<K> keySet();
}
```
TreeMultiMap

- Implements the `MultiMap<K, V>` interface.
- Uses a JCF `TreeMap<Km, Vm>` to store the data:
  - Key type `Km` is the `MultiMap` key type `K`.
  - Value type `Vm` is a `LinkedList<D>` where `D` is the `MultiMap` value type.

```java
public class TreeMultiMap<K extends Comparable<K>, V> implements MultiMap<K, V> {
    TreeMap<K, LinkedList<V>> theMap;

    public TreeMultiMap() {
        this.theMap = new TreeMap<K, LinkedList<V>>();
    }

    // ... Omitted ...

    public void put(K key, V value) {
        LinkedList<V> values = theMap.get(key);
        if (values==null) {
            values = new LinkedList<V>();
            theMap.put(key, values);
        }
        values.add(value);
    }

    public LinkedList<V> get(K key) {
        return theMap.get(key);
    }

    public Set<K> keySet() {
        return theMap.keySet();
    }
}
```
Anagram Groups (Version 1)

- Sort words into buckets corresponding to anagram groups:
  - Key is sorted character string.
  - Value is word.
- Iterate over key set (sorted character strings) and print the bucket contents (anagram group) for each key.

Anagram Groups (Version 2)

- Sort anagram groups into buckets corresponding to group size:
  - Key is number of words in the anagram group.
  - Value is the anagram group.
- Iterate over key set (sizes of groups) and print the bucket contents (all anagram groups of that size) for each key.