CSE 8B—Intro to CS: Java

Winter, 2006

March 9 (Day 18)
Collections
Iterators
Maps

Instructor: Neil Rhodes

Collection<E>

Interface represents a collection of elements (of type E):

- Required operations:
  - Find whether an element is present in the collection
    - boolean contains(Object o)
  - Return number of elements
    - int size()
  - Convert to array (puts in passed-in array if will fit, otherwise returns new array):
    - <T> T[] toArray(T[] arr)
  - Return an iterator to iterate over the elements in the collection
    - Iterator<E> iterator()

Following operations are collection mutators. All return true if the collection was actually modified.

- boolean add(E o)
- boolean addAll(Collection<? extends E> c)
- boolean remove(Object o)
- boolean removeAll(Collection<?> c)
- boolean retainAll(Collection<? super E> c)
- boolean clear()

Some collections can’t support the above operations (read-only collections, for example)

- The operations will throw a
  UnsupportedOperationException if not supported

Collection Hierarchy
Different collections for different purposes

Sets
- order doesn’t matter
- duplicates are not allowed

Sorted Set
- order does matter
- duplicates are not allowed

Lists
- order does matter
- duplicates are allowed

Sequential list
- quick to add/delete
- may be slow to access a particular element (random-access mode)

Examples uses

HashSet<E>
- Keeping a collection of CSE courses offered in the Spring.

TreeSet<E>
- Keeping that same collection of CSE courses if you want to then print out the results sorted by course number

ArrayList<E>
- Keeping an array of strings in a certain order

LinkedList<E>
- Keeping a list of notifiers (no need to random access, quick add/delete a bonus).

Extra methods provided by specific classes

List<E>
- void add(int index, E newElement)
- E get(int index)
- void set(int index, E newElement)
- E remove(int index)
- int indexOf(Object target)
- int lastIndexOf(Object target)
- List<E> sublist(int fromIndex, int toIndex)
- ListIterator<E> listIterator()
- ListIterator<E> listIterator(int startIndex)

HashSet<E>
- HashSet(int initialCapacity)

ArrayList<E>
- ArrayList(int initialCapacity)

Example: count unique words in input stream

```java
import java.util.*;
import java.io.*;

public class UniqueWords {
  public static void main(String[] args) {
    Scanner in = null;
    try {
      in = new Scanner(new FileInputStream(args[0]));
    } catch (FileNotFoundException e) {
      System.err.println("can't open: " + e.getMessage());
      System.exit(0);
    }
    Set<String> set = new HashSet<String>();
    while (in.hasNext())
      set.add(in.next());
    System.out.println("# unique words: " + set.size());
  }
}
```
Iterators

Iterators iterate over items (of a collection)
Methods:
- boolean hasNext()
- E next()
- void remove();
  - can throw UnsupportedOperationException

Usage

Simpler syntax using for-each loop

Set<String> set = …
...
for (Iterator<String> iter = set.iterator(); iter.hasNext();)
      System.out.println(iter.next());
Set<String> set = …
...
for (String s: set)
  System.out.println(s);

ListIterator

ListIterator can go forward and backward and can modify the collection:
- boolean hasPrevious()
- int nextIndex();
- int previousIndex();
- E previous();
- void add(E o);
  - inserts between next and previous (afterwards, previous is o)
- void remove();
  - removes the last element retrieved from next or previous
- void set(E o);
  - replaces the last element retrieved from next or previous

Map

Used to map from one object (key) to another object (value)
- For example, keep count of how many times a word appeared in a file (mapping from word to number)

Methods
- put(K key, V value)
- boolean containsKey(K key)
- boolean containsValue(V value)
- V get(K key)
- V remove(K key)
- Set<T> keySet()
- Collection<T> values()
- Entry<K, V> entrySet()

Inner class
Entry<K, V> stores a key/value pair
Methods:
  K getKey()
  V getValue()
Map Example
Count occurrences of words in a file
   Scanner in;
   ...
   Map<String, Integer> m = new HashMap<String, Integer>();
   while (in.hasNext()) {
     String s = in.next();
     Integer num = m.get(s);
     if (num == null)
       m.put(s, 1);
     else
       m.put(s, num + 1);
   }
   for (Map.Entry<String, Integer> e: m.entrySet())
     System.out.println(e.getKey() + " : " + e.getValue());

Interface vs. Implementation

<table>
<thead>
<tr>
<th>Collection</th>
<th>Hash table</th>
<th>Resizable array</th>
<th>Balanced Tree</th>
<th>Linked list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>HashSet</td>
<td>TreeSet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SortedSet</td>
<td>HashSet</td>
<td>TreeSet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List</td>
<td>ArrayList</td>
<td>TreeSet</td>
<td></td>
<td>LinkedList</td>
</tr>
<tr>
<td>Map</td>
<td>HashMap</td>
<td>TreeMap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SortedMap</td>
<td>TreeMap</td>
<td>TreeMap</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Combined Map/List example
Finds all anagrams
   For each word, adds to a list keyed by alphabetized version of word
   Map<String, List<String>> m =
     new HashMap<String, List<String>>();
   while (in.hasNext()) {
     String s = in.next();
     String alphabetic = alphabetize(s);
     List<String> list = m.get(alphabetic);
     if (list == null)
       m.put(alphabetic, list = new ArrayList<String>());
     list.add(s);
   }
   for (List<String> l: m.values())
     if (l.size() > 5)
       System.out.println(l.size() + " : " + l);