BUILDING JAVA PROGRAMS CHAPTER 11

Java Collections Framework

In Class Examples.

SETS

Removing from a set

- What is the proper way to remove from a set while iterating through it.

ITERATORS

reading: 11.1; 15.3; 16.5

Examining sets and maps

- elements of Java `Set`s and `Map`s can't be accessed by index
  - must use a "foreach" loop:

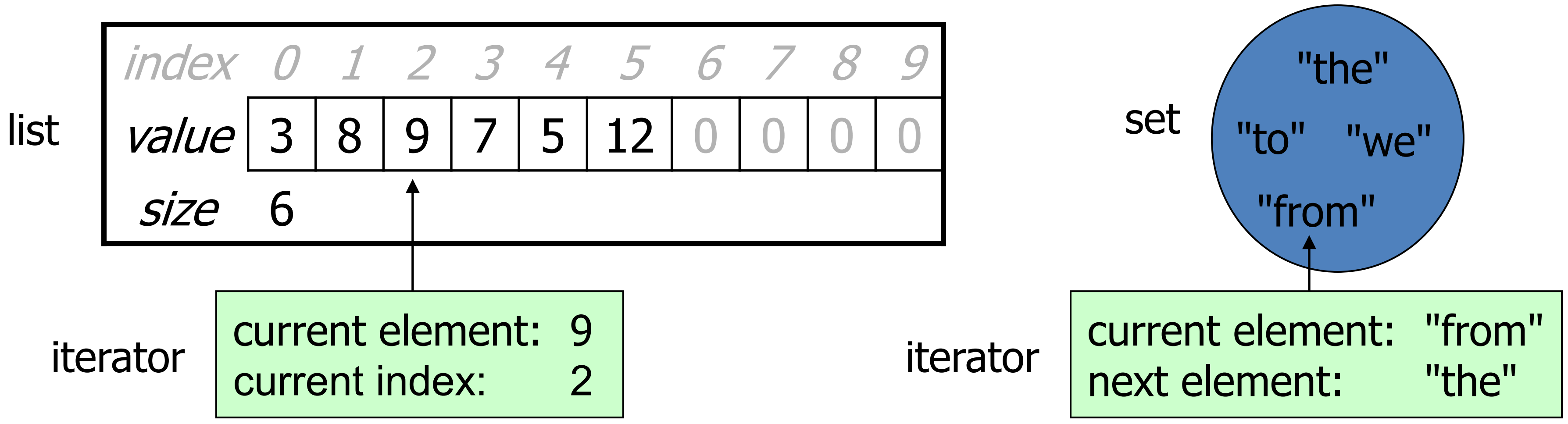
```
Set<Integer> scores = new HashSet<Integer>();
for (int score : scores) {
    System.out.println("The score is " + score);
}
```

  - Problem: foreach is read-only; cannot modify set while looping

```
for (int score : scores) {
    if (score < 60) {
    // throws a ConcurrentModificationException
        scores.remove(score);
    }
}
```

Iterators (11.1)

- **iterator**: An object that allows a client to retrieve the elements of any collection.
  - Remembers a position, and lets you:
    - get the element at that position
    - advance to the next position
    - remove the element at that position



`Iterator` methods

`hasNext()`	returns `true` if there are more elements to examine
`next()`	returns the next element from the collection (throws a `NoSuchElementException` if there are none left to examine)
`remove()`	removes the last value returned by `next()` (throws an `IllegalStateException` if you haven't called `next()` yet)

- `Iterator` interface in `java.util`
  - every collection has an `iterator()` method that returns an iterator over its elements

```
Set<String> set = new HashSet<String>();
...
Iterator<String> itr = set.iterator();

while (itr.hasNext()) {
    /* do something with itr.next() */
```

Iterator example

```
Set<Integer> scores = new TreeSet<Integer>();
scores.add(94);
scores.add(38);    // Kim
scores.add(87);
scores.add(43);    // Marty
scores.add(72);
…

Iterator<Integer> itr = scores.iterator();
while (itr.hasNext()) {
    int score = itr.next();

    System.out.println("The score is " + score);

    // eliminate any failing grades
    if (score < 60) {
        itr.remove();
    }
}
System.out.println(scores);  // [72, 87, 94]
```

Iterator example 2

```
Map<String, Integer> scores = new TreeMap<String, Integer>();
scores.put("Kim", 38);
scores.put("Lisa", 94);
scores.put("Roy", 87);
scores.put("Marty", 43);
scores.put("Marisa", 72);
...

Iterator<String> itr = scores.keySet().iterator();
while (itr.hasNext()) {
    String name = itr.next();
    int score = scores.get(name);
    System.out.println(name + " got " + score);

    // eliminate any failing students
    if (score < 60) {
        itr.remove();      // removes name and score
    }
}
System.out.println(scores);  // {Lisa=94, Marisa=72, Roy=87}
```

WordCount Exercise

- Write a program to count the occurrences of each word in a large text file (e.g. *Moby Dick* or the King James Bible).
  - Allow the user to type a word and report how many times that word appeared in the book
  - Report all words that appeared in the book at least 1000 times
- How will we store the data to solve this problem?

- Write a method removeEvenLength that accepts a set of strings as a parameter and that removes all the strings of even length from the set.

- Write a method called `sortAndRemoveDuplicates` that accepts a list of integers as its parameter and rearranges the list's elements into sorted ascending order, as well as removing all duplicate values from the list. For example, the list [7,4,-9, 4, 15, 8, 27, 7, 11, -5, 32, -9, -9] would become [-9, -5, 4, 7, 8, 11, 15, 27, 32] after a call to your method. Use a set as part of your solution.

MAP ADT

Using the Map ADT - Examples

Example 1

- Write a method `isUnique` that accepts a Map from strings to strings as a parameter and returns true if no two keys map to the same value (and false if any two or more keys do map to the same value).
- For example, calling your method on the following map would return true:
  - {Marty=Stepp, Stuart=Reges, Jessica=Miller, Amanda=Camp, Hal=Perkins}
- Calling it on the following map would return false,
  - {Kendrick=Perkins, Stuart=Reges, Jessica=Miller, Bruce=Reges, Hal=Perkins}
- The empty map is considered to be unique, so your method should return true if passed an empty map.

Example 2

- Write a method that takes a set of strings and then maps them by length.

- Print off the size of each group based upon length.
  - i.e. *"There are ____ words of size ____"*

- Then return the largest set of words back to the original program.