Linked List Size

Write a recursive method to compute the length of a linked list.

```java
public class ListNode {
    public int data;
    public ListNode next;
}

public class LinkedList {
    private ListNode head;
    public int size() {
        // ???
    }
}
```

Exercise

- Write a program that counts the number of unique words in a large text file (say, *Hamlet* or the King James Bible).
  - Store the words in a collection and report the # of unique words.
  - Once you've created this collection, allow the user to search it to see whether various words appear in the text file.
- What collection is appropriate for this problem?

Sets (11.2)

- **set**: A collection of unique values (no duplicates allowed) that can perform the following operations efficiently:
  - add, remove, search (contains)
  - We don't think of a set as having indexes; we just add things to the set in general and don't worry about order

```java
set.contains("to")  // true
set.contains("be")  // false
```
Set implementation

- in Java, sets are represented by `Set` interface in `java.util`
- `Set` is implemented by `HashSet` and `TreeSet` classes
  - `HashSet`: implemented using a "hash table" array; very fast: \(O(1)\) for all operations; elements are stored in unpredictable order
  - `TreeSet`: implemented using a "binary search tree"; pretty fast: \(O(\log N)\) for all operations; elements are stored in sorted order
  - `LinkedHashSet`: \(O(1)\) but stores in order of insertion

Set methods

```
List<String> list = new ArrayList<String>();
...
Set<Integer> set = new TreeSet<Integer>();    // empty
Set<String> set2 = new HashSet<String>(list);
```
- can construct an empty set, or one based on a given collection

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(\text{value})</td>
<td>adds the given value to the set</td>
</tr>
<tr>
<td>contains(\text{value})</td>
<td>returns \text{true} if the given value is found in this set</td>
</tr>
<tr>
<td>remove(\text{value})</td>
<td>removes the given value from the set</td>
</tr>
<tr>
<td>clear()</td>
<td>removes all elements of the set</td>
</tr>
<tr>
<td>size()</td>
<td>returns the number of elements in list</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>returns \text{true} if the set's size is 0</td>
</tr>
<tr>
<td>toString()</td>
<td>returns a string such as &quot;[3, 42, -7, 15]&quot;</td>
</tr>
</tbody>
</table>

Sets and ordering

- `HashSet`: elements are stored in an unpredictable order
  ```java
  Set<String> names = new HashSet<String>();
  names.add("Twilight Sparkle");
  names.add("Applejack");
  Names.add("Rainbow Dash");
  Names.add("Pinkie Pie");
  names.add("Fluttershy");
  names.add("Rarity");
  System.out.println(names);
  // [Rainbow Dash, Fluttershy, Twilight Sparkle,
  Rarity, Applejack, Pinkie Pie]
  ```

- `TreeSet`: elements are stored in their "natural" sorted order
  ```java
  Set<String> names = new TreeSet<String>();
  ...
  // [Applejack, Fluttershy, Pinkie Pie, Rainbow Dash,
  Rarity, Twilight Sparkle]
  ```

The "for each" loop (7.1)

```
for (\text{type name} : \text{collection}) {
    \text{statements};
}
```
- Provides a clean syntax for looping over the elements of a `Set`, `List`, array, or other collection

```
Set<Double> grades = new HashSet<Double>();
...
for (double grade : grades) {
    System.out.println("Student's grade: "+ grade);
}
```
- needed because sets have no indexes; can't get element i
Exercise

• Write a program to count the number of occurrences of each unique word in a large text file (e.g. *Moby Dick*).
  – 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 500 times, in alphabetical order.

• What collection is appropriate for this problem?

Maps and tallying

• a map can be thought of as a generalization of a tallying array
  – the "index" (key) doesn't have to be an int

• recall LetterInventory example from last week:
  – count letters: "cabbage"
    | letter | a | b | c | d | e | f | g | h |
    | index  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
    | value  | 2 | 2 | 1 | 0 | 1 | 0 | 1 | 0 |

  – count words: "I was like baby, baby, oh Like baby, baby, baby, no"
    ("Baby," Justin Bieber)

Map implementation

• in Java, maps are represented by Map interface in java.util

  Map is implemented by the HashMap and TreeMap classes
  – HashMap: implemented using an array called a "hash table"; extremely fast: $O(1)$; keys are stored in unpredictable order
  – TreeMap: implemented as a linked "binary tree" structure; very fast: $O(\log N)$; keys are stored in sorted order

  – A map requires 2 type parameters: one for keys, one for values.

  // maps from String keys to Integer values
  Map<String, Integer> votes = new HashMap<String, Integer>();
Map methods

- `put(key, value)`: adds a mapping from the given key to the given value; if the key already exists, replaces its value with the given one
- `get(key)`: returns the value mapped to the given key (`null` if not found)
- `containsKey(key)`: returns `true` if the map contains a mapping for the given key
- `remove(key)`: removes any existing mapping for the given key
- `clear()`: removes all key/value pairs from the map
- `size()`: returns the number of key/value pairs in the map
- `isEmpty()`: returns `true` if the map's size is 0
- `toString()`: returns a string such as `{a=90, d=60, c=70}`
- `keySet()`: returns a set of all keys in the map
- `values()`: returns a collection of all values in the map
- `putAll(map)`: adds all key/value pairs from the given map to this map
- `equals(map)`: returns `true` if given map has the same mappings as this one

Using maps

- A map allows you to get from one half of a pair to the other.
  - Remembers one piece of information about every index (key).
    - // key value
    - `put("Daniel", "541-346-4154")`
  - Later, we can supply only the key and get back the related value:
    - Allows us to ask: What is Daniel's phone number?
    - `Map`
    - `get("Daniel")`
    - "541-346-4154"

Maps vs. sets

- A set is like a map from elements to `boolean` values.
  - Set: Is Daniel found in the set? (true/false)
    - "Daniel" `true` `false`
  - Map: What is Daniel's phone number?
    - "Daniel" "541-346-4154"

Example: The Oscars!

Winners of the 84th Annual Academy Awards:
- Best Picture: The Artist
- Best Actor: Jean Dujardin, The Artist
- Best Actress: Meryl Streep, The Iron Lady...
- Best Visual Effects: Hugo
- Best Art Direction: Hugo
- Best Cinematography: Hugo

```java
Map<String, String> oscars = new TreeMap<String, String>();
oscars.put("Cinematography", "Hugo");
oscars.put("Best Visual Effects", "Hugo");
oscars.put("Best Picture", "The Artist");
```
keySet and values

- `keySet` method returns a `Set` of all keys in the map
  - can loop over the keys in a `foreach` loop
  - can get each key’s associated value by calling `get` on the map

```java
for (String award : oscars.keySet() { 
  String movie = oscars.get(award); 
  System.out.println(award + " : " + movie); 
}
```

- `values` method returns a collection of all values in the map
  - can loop over the values in a `foreach` loop
  - no easy way to get from a value to its associated key(s)

Problem: opposite mapping

- Suppose we want to ask which movies got which awards.
  - How would we structure a map for that?

```java
Map<String, String> oscarsWon = new
    HashMap<String, String>();
oscarsWon.put("Hugo", "Best Cinematography");
oscarsWon.put("Hugo", "Best Visual Effects");
oscarsWon.put("The Artist", "Best Picture");

System.out.println("Hugo won: " +
    oscarsWon.get("Hugo")); // ???
```

- Problem: A movie, such as "Hugo," can win multiple Oscars.

Proper Map Reversal

- Map each movie to a `collection` of awards:

```java
Map<String, Set<String>> oscarsWon =
    new HashMap<String, Set<String>>();
oscarsWon.put("Hugo", new HashSet<String>());
oscarsWon.get("Hugo").add("Best Cinematography");
oscarsWon.get("Hugo").add("Best Visual Effects");
oscarsWon.put("The Artist", new HashSet<String>());
oscarsWon.get("The Artist").add("Best Picture");

System.out.println("Hugo won: " +
    oscarsWon.get("Hugo")); // [Best Cinematography, Best Visual Effects]
```

(Must be careful to initialize the set for a given movie before adding.)

Perl and Python

Maps/Dictionaries are a built-in feature of many scripting languages!

Python:

```python
wordCount = {}
...
if wordCount.has_key(word):
    wordCount[word] += 1
else:
    wordCount[word] = 1
```

Perl:

```perl
my %wordcount = ();
...
if (defined($wordCount{$word})) {
    $wordCount{$word}++;
} else {
    $wordCount{$word} = 1;

$wordCount{$word}++;
```
Exercise solution

// read file into a map of [word --> number of occurrences]
Map<String, Integer> wordCount = new HashMap<String, Integer>();
Scanner input = new Scanner(new File("mobydick.txt"));
while (input.hasNext()) {
    String word = input.next();
    if (wordCount.containsKey(word)) {
        // seen this word before; increase count by 1
        int count = wordCount.get(word);
        wordCount.put(word, count + 1);
    } else {
        // never seen this word before
        wordCount.put(word, 1);
    }
}
Scanner console = new Scanner(System.in);
System.out.print("Word to search for? ");
String word = console.next();
System.out.println("appears " + wordCount.get(word) + " times.");