RadixSort.java

Objective: To make a program that uses “buckets” to sort data.

Background:
We learned several techniques for sorting data back in Chapter 13. Radix Sort is a sorting method that is not based on comparing keys but rather on applying the lookup or hashing idea to them.

Suppose we have a large list of integers with values of 0 to 99. We can create 100 buckets, corresponding to the 100 possible values. In one pass through the list we add each value to the appropriate bucket. Then we scan through 100 buckets in ascending order and collect all of the values together. The result will be a list sorted in ascending order.

For this assignment, instead of integers, we will sort words in lexicographic order (dictionary order). We have to pad (logically, not literally) all the words with “spaces” to the maximum length and start sorting from the rightmost character position. Assume all the words are made up of lowercase letters ‘a’ through ‘z’.

Assignment:
Download the file **RadixSort.zip** from Mr Greenstein’s web site. Unzip the file and it will create the directory **RadixSort**. There will be two files in the directory, **RadixSort.java** and **randomWords.txt**. Add your **FileUtils** class to help open a file to read. **RadixSort** should do the following.

1. Read all the words from **randomWords.txt** into **words**.
2. Sort the **words** using Radix Sort.
3. Print out all the **words** in sorted order.

For part 2, you will implement the following sorting method in **RadixSort.java**:

```
public LinkedList<String> sortWords(LinkedList<String> words)
```

This method inputs a list of words in random order and outputs a list in lexicographic order. Use an **ArrayList** of **List<String>** to hold the temporary buckets, and each bucket is a **LinkedList** of words. Use **Character.getNumericValue(ch)** to get integer values of characters to put them in the right bucket. Remember to “pad” (logically) shorter words with spaces. Use 27 buckets, one for each letter of the alphabet plus the first one reserved for a space.

A sample run:

```
% head -20 randomWords.txt
precept
altra
auxquels
stoel
vena
whipt
vagaries
welled
parisienne
tonnage
turning
ladder
fu
stables
chateaux
receives
strengthens
stato
egoist
```

```
toned
% java RadixSort > output.txt
% head -20 output.txt
a
aabnede
aacute
aamulla
aamuna
aan
aandacht
aangenaam
aangezicht
aankomst
aanleiding
aanstonds
aantal
aanval
aanzien
aard
aarde
aardig
ab
aback
% tail -20 output.txt
zwart
zwarte
zwarten
zwei
zweier
zweifeln
zweimal
zweite
zweiten
zweitens
zweiter
zwingen
zwischen
zwoelf
zy
zyn
zynde
zyne
zynen
zzzzz
```