Lab10 - War and Peace

Due: Wed Mar 9, 2015

This lab focuses on:

- ❑ Chapter 18 Collections

First step - Copy all the files from my k: drive, most notably `war_and_peace.txt`.

The Gutenberg Project is cool. They put free ebooks online: www.gutenberg.org. Click, click, click… I found War and Peace by Tolstoy. It's just a text file, so we can play with it.

The `Lab10` class is where you'll start. It contains our `main()` and a `switch` for the 4 parts of Lab 10. Go!

Part 1 - Count words

Question: How many words are in War and Peace?
Answer: Read each token in the file and count 'em.
Here's the EZ pseudo-code:

```
word count = 0
foreach( token in file) {
   increment word count
}
```

Actually, this part is already coded up. Walk through it and run. We'll be using it throughout.

Part 2 - List of all words

Question: What are all the words in War and Peace?
Answer: Instead of just counting each token, let's store it in a list.

Copy-paste the `part1()` method to `part2()` and make your changes.
Here's the pseudo-code:

```
create new LinkedList
foreach( token in file) {
   add token to list
}
write each word in the list to a file
```

For your file name, use the constant: `PART2_WORDS_FILE`.
I recommend the `println()` method in `PrintWriter` to do your writing. This will write one word per line.

Part 3 - Count unique words

Question: How many **different** words are there in War and Peace?
Answer: This sounds like a job for a Set. Remember - sets only store each object once. So, no matter how many times a word appears in War and Peace, it will appear in our set only one time.
Pseudo me!

```
create new TreeSet
foreach( token in file) {
   make token lower case
   add token to the set
}
write each word in the TreeSet to a file
```

Try `toLowerCase()` in the `String` class to make all your chars lower case.
After processing all tokens, the size of your collection is your unique word count

Part 4 - Rank words

Question: What are the **top 100** most used words in War and Peace?
Answer: This one's a little tougher. We need to associate (map!) a current count (or tally) with each word. Let's try using a `HashMap`.
Here's the pseudo-code:

```
create new Map
foreach( token in file) {
   current tally = get tally from Map for this token
   if( tally == null) {
      create a new tally with count 1
      put it in Map
   }
   else {
      increment the tally
   }
}
// our map is complete...now, report the top 100
convert your Map to a List
sort List by tally
report top 100
```

Notes:

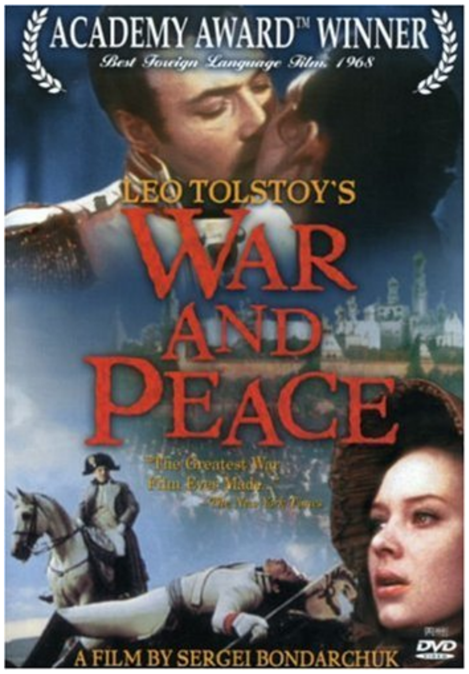
- I have provided a class, `WordTally`, that you can use to store each word's count in the map. It's very simple. Check it out.

- For `HashMap`, the `values()` method returns all the values in the map (`WordTally` objects, in our case) as a `Collection`.
- For printing in that last step, you need to sort your list using a `Comparator`. Create one for `WordTally` objects that sorts on the class variable tally.

Good luck!
At least, I wish you better luck than gloomy, old Tolstoy does.

QOTD

We can know only that we know nothing.
And that is the highest degree of human wisdom.
- L Tolstoy, War and Peace (www.goodreads.com/work/quotes/4912783)



PS - Some sample output for you…

PART1

```
Welcome to Lab 10
Collections!

war_and_peace.txt

**> PART 1 - total words
war_and_peace.txt: word count=???
```

PART2

```
Welcome to Lab 10
Collections!

war_and_peace.txt

PART 2 - save words
Num words in list=???
Words written to file=part2_words.txt
```

PART3

```
Welcome to Lab 10
Collections!

war_and_peace.txt

PART 3 - unique words
Num unique words in tree=???
Unique words written to file=part3_unique_words.txt
```

PART4

```
Welcome to Lab 10
Collections!

war_and_peace.txt

**> PART 4 - rank words
      1. XXX (???)
      2. YYY (???)
      3. ZZZ (???)
      … and so on
```

PART5

```
Welcome to Lab 10
Collections!

war_and_peace.txt
Error: Bad part choice=5
```