Programming Assignment: WordNet

✔ Passed · 100/100 points

Deadline Pass this assignment by December 24, 11:59 PM PST

Instructions

My submission

Discussions

Specification

Here is the programming assignment specification that describes the assignment requirements.

Be sure that your code conforms to the prescribed APIs: each program must be in the "default" package (i.e., no **package** statements) and include only the public methods and constructors specified (extra private methods are fine). Note that **algs4.jar** uses a "named" package, so you must use an **import** statement to access a class in **algs4.jar**.

Checklist

The checklist contains frequently asked questions and hints. If you're not sure where to start, see the section at the end of the checklist.

Testing

The file wordnet-testing.zip contains sample data files that you can use to test **SAP.java**, **WordNet.java**, and **Outcast.java**.

Web Submission

Submit a zip file named **wordnet.zip** that contains the source files **SAP.java**, **WordNet.java**, **Outcast.java** along with any other supporting files (excluding **algs4.jar**). You can use one of the following three approaches to create the zip file:

Mac OS X.

1. Select the required files in the Finder.
2. Right-click and select *Compress 3 Items*.
3. Rename the resulting file to **wordnet.zip**.

Windows.

1. Select the required files in Windows Explorer.
2. Right-click and select *Send to -> Compressed (zipped) folder*.
3. Rename the resulting file to **wordnet** (the .zip extension is automatic).

Command line (Linux or Mac OS X).

1. Change to the directory containing the required .java files.
2. Execute the command: **zip wordnet.zip SAP.java WordNet.java Outcast.java**

Assessment Report

Here is some information to help you interpret the assessment report. See the Assessment Guide for more details.

- *Compilation*: we compile your .java files using a Java 8 compiler. Any error or warning messages are displayed and usually signify a major defect in your code.
- *Bugs*: we run Findbugs to check for common bug patterns in Java programs. A warning message strongly suggests a bug in your code but occasionally there are false positives. Here is a summary of bug descriptions, which you can use to help decode warning messages.
- *Style*: we run Checkstyle to automatically checks the style of your Java programs. Here is a list of available Checkstyle checks, which you can use to help decode any warning messages.
- *API*: we check that your code exactly matches the prescribed API (no extra methods and no missing methods). If it does not, no further tests are performed.
- *Correctness*: we perform a battery of unit tests to check that your code meets the specifications.
- *Memory*: we determine the amount of memory according to the 64-bit memory cost model from lecture.
- *Timing*: we measure the running time and count the number of elementary operations.

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submission" tab.