Programming Assignment 1: WordNet | wordnet.zip

Help Center

Submission	
Submission time	Mon-16-Nov 15:49:54
Raw Score	103.12 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  35/35 tests passed
Memory:       4/4 tests passed
Timing:       18/16 tests passed

Aggregate score: 103.12% [Correctness: 65%, Memory: 10%, Timing: 25
%, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 28K
-rw-r--r-- 1 1.2K Nov 16 23:51 Outcast.java
-rw-r--r-- 1 7.9K Nov 16 23:51 SAP.java
-rw-r--r-- 1 6.5K Nov 16 23:51 WordNet.java
-rw-r--r-- 1 4.6K Nov 16 23:51 studentSubmission.zip


*********************************************************************
***********
*  compiling
```

```
**********************************************************************
**********


% javac SAP.java
*-----------------------------------------------------------


================================================================


% javac WordNet.java
*-----------------------------------------------------------


================================================================


% javac Outcast.java
*-----------------------------------------------------------


================================================================



% checkstyle SAP.java WordNet.java Outcast.java
*-----------------------------------------------------------
================================================================


% findbugs *.class
*-----------------------------------------------------------
================================================================


Testing the APIs of your programs.
*-----------------------------------------------------------
SAP:

WordNet:

Outcast:

================================================================


**********************************************************************
**********
```

```
*  correctness
*****************************************************************
**********

Testing methods in SAP
*-----------------------------------------------------------
Running 19 total tests.

Test 1: test length() and ancestor() on fixed digraphs
  *  digraph1.txt
  *  digraph2.txt
  *  digraph3.txt
  *  digraph4.txt
  *  digraph5.txt
  *  digraph6.txt
  *  digraph9.txt
==> passed

Test 2: check length() and ancestor() on WordNet digraph
  *  100 random vertex pairs in digraph-wordnet.txt
==> passed

Test 3: check length() and ancestor() on directed paths
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 4: check length() and ancestor() on directed cycles
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 5: check length() and ancestor() on complete graphs
  *  5
  *  10
  *  20
  *  50
==> passed
```

```
Test 6: check length() and ancestor() on tournament digraphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 7: check length() and ancestor() on complete binary trees
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 8: check length() and ancestor() on random DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 9: check length() and ancestor() on random rooted-in DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 10: check length() and ancestor() on random rooted-out DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 11: check length() and ancestor() on random rooted-in trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
==> passed

Test 12: check length() and ancestor() on random rooted-out trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
```

```
==> passed

Test 13: check length() and ancestor() on random simple digraphs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 14: check whether two SAP objects can be created at the same t
ime
  *  digraph1.txt and digraph2.txt
  *  digraph3.txt and digraph4.txt
  *  digraph5.txt and digraph6.txt
  *  digraph2.txt and digraph1.txt
==> passed

Test 15: check whether SAP is immutable
  *  digraph1.txt
  *  digraph2.txt
  *  digraph3.txt
  *  digraph4.txt
  *  digraph5.txt
  *  digraph6.txt
  *  digraph-ambiguous-ancestor.txt
==> passed

Test 16: test invalid arguments to length() and ancestor() in digra
ph1.txt
  *  v = -1, w = 0
  *  v = 0, w = -1
  *  v = 13, w = 0
  *  v = 0, w = 13
  *  v = -1 2 7 , w = 1 4 6 10 11
  *  v = 2 7 , w = -1 1 4 6 10 11
  *  v = 13 2 7 , w = 1 4 6 10 11
  *  v = 2 7 , w = 13 1 4 6 10 11
==> passed

Test 17: test length() and ancestor() with Iterable arguments
  *  100 random subsets of 1 and 1 vertices in digraph-wordnet.txt
  *  100 random subsets of 1 and 2 vertices in digraph-wordnet.txt
  *  100 random subsets of 2 and 1 vertices in digraph-wordnet.txt
  *  100 random subsets of 2 and 2 vertices in digraph-wordnet.txt
  *  100 random subsets of 3 and 11 vertices in digraph-wordnet.txt
```

```
  *  100 random subsets of 11 and 3 vertices in digraph-wordnet.txt
  *  100 random subsets of 0 and 5 vertices in digraph-wordnet.txt
  *  100 random subsets of 5 and 0 vertices in digraph-wordnet.txt
  *  100 random subsets of 0 and 0 vertices in digraph-wordnet.txt
==> passed

Test 18: Check Iterable version of length() and ancestor() with nul
l arguments
==> passed

Test 19: random calls to length() and ancestor(), with probabilitie
s
         p1 and p2, respectively
  *  random calls in a random rooted DAG (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
  *  random calls in a random digraph (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
==> passed


Total: 19/19 tests passed!

================================================================

*****************************************************************
**********
*  correctness (substituting reference SAP.java)
*****************************************************************
**********

Testing methods in WordNet
*-----------------------------------------------------------
Running 14 total tests.

Test 1: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 2: test distance() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
```

```
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 3: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 4: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 5: test sap() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 6: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
```

```
==> passed

Test 7: check whether WordNet is immutable
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 8: check that constructor throws an IllegalArgumentException w
hen the input is not a rooted DAG
  *  synsets3.txt, hypernyms3InvalidTwoRoots.txt
more than one root
  *  synsets3.txt, hypernyms3InvalidCycle.txt
more than one root
  *  synsets6.txt, hypernyms6InvalidTwoRoots.txt
more than one root
  *  synsets6.txt, hypernyms6InvalidCycle.txt
more than one root
  *  synsets6.txt, hypernyms6InvalidCycle+Path.txt
more than one root
==> passed

Test 9: check that distance() and sap() throw an IllegalArgumentExc
eption
        when either argument is not a WordNet noun
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = invalid
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = b
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = eleventeen
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = INVALID
==> passed

Test 10: check isNoun()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 11: check nouns()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 12: check whether two WordNet objects can be created at the sa
me time
  *  synsets1 = synsets15.txt; hypernyms1 = hypernyms15Tree.txt
```

```
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
  *  synsets1 = synsets.txt; hypernyms1 = hypernyms.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
==> passed

Test 13: call distance(), sap(), and isNoun() with null arguments
  *  synsets15.txt, hypernyms15Path.txt
==> passed

Test 14: random calls to isNoun(), distance(), and sap(), with
         probabilities p1, p2, and p3, respectively
  *  100 random calls (p1 = 0.5, p2 = 0.5, p3 = 0.0)
  *  100 random calls (p1 = 0.5, p2 = 0.0, p3 = 0.5)
  *  100 random calls (p1 = 0.0, p2 = 0.5, p3 = 0.5)
  *  100 random calls (p1 = 0.2, p2 = 0.4, p3 = 0.4)
==> passed


Total: 14/14 tests passed!

================================================================
**********************************************************************
**********
*  correctness (substituting reference SAP.java and WordNet.java)
**********************************************************************
**********

Testing methods in Outcast
*-----------------------------------------------------------
Running 2 total tests.

Test 1: test outcast() on WordNet digraph
        (synsets.txt and hypernyms.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast4.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8a.txt
  *  outcast8b.txt
  *  outcast8c.txt
```

```
  *  outcast9.txt
  *  outcast9a.txt
  *  outcast10.txt
  *  outcast10a.txt
  *  outcast11.txt
  *  outcast12.txt
  *  outcast12a.txt
  *  outcast17.txt
  *  outcast20.txt
  *  outcast29.txt
==> passed

Test 2: test outcast() on WordNet subgraph
        (synsets50000-subgraph.txt and hypernyms50000-subgraph.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast10.txt
  *  outcast11.txt
==> passed


Total: 2/2 tests passed!

================================================================

*****************************************************************
**********
*  memory
*****************************************************************
**********

Computing memory of SAP
*-----------------------------------------------------------
Running 1 total tests.

student     memory    = 9827704 bytes
reference   memory    = 8347712 bytes
```

```
ratio                  = 1.18
maximum allowed ratio = 2.50

vertices = 82192
edges    = 84505

Total: 1/1 tests passed!

================================================================



Computing memory of WordNet
*-----------------------------------------------------------
Running 3 total tests.

Test 1a: test memory of WordNet object
  *  synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-
subgraph.txt
     -  student   memory = 663560 bytes
     -  reference memory = 1198784 bytes
     -  number vertices  = 1000
     -  number of edges  = 1008
     -  student / reference ratio  =  0.6
     -  maximum allowed rato       =  2.0

==> passed

Test 1b: test memory of WordNet object
  *  synsets = synsets5000-subgraph.txt; hypernyms = hypernyms5000-
subgraph.txt
     -  student   memory = 3310760 bytes
     -  reference memory = 5952136 bytes
     -  number vertices  = 5000
     -  number of edges  = 5059
     -  student / reference ratio  =  0.6
     -  maximum allowed rato       =  2.0

==> passed

Test 1c: test memory of WordNet object
  *  synsets = synsets10000-subgraph.txt; hypernyms = hypernyms1000
0-subgraph.txt
     -  student   memory = 7846112 bytes
```

```
     -  reference memory = 13853632 bytes
     -  number vertices  = 10000
     -  number of edges  = 10087
     -  student / reference ratio  =  0.6
     -  maximum allowed rato       =  2.0

==> passed

Total: 3/3 tests passed!

================================================================



*****************************************************************
**********
*  timing
*****************************************************************
**********

Timing SAP
*-----------------------------------------------------------
Running 7 total tests.

Test 1: time SAP constructor
   *  digraph-wordnet.txt
      -  student solution time =  0.04 seconds
      -  maximum allowed  time =  1.00 seconds
==> passed

Test 2a-c: time length() and ancestor() with random pairs of vertic
es
  *  digraph-wordnet.txt
     -  reference solution calls per second:  177525.33
     -  student   solution calls per second:  228399.33
     -  reference / student ratio:                0.78

=> passed    student <= 25000x reference
=> passed    student <=  2500x reference
=> passed    student <=   250x reference
=> BONUS     student <=  10.0x reference

Test 3a-c: time length() and ancestor() with random sets of 5 verti
ces
```

```
  *  digraph-wordnet.txt
     -  reference solution calls per second:   37323.33
     -  student   solution calls per second:   55055.33
     -  reference / student ratio:                0.68

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> BONUS     student <=  10.0x reference


Total: 9/7 tests passed!

================================================================



*****************************************************************
***********
*  timing (substituting reference SAP.java)
*****************************************************************
***********

Timing WordNet
*-----------------------------------------------------------
Running 8 total tests.

Test 1: timing WordNet constructor
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  student constructor time =  1.00 seconds
     -  maximum allowed     time = 10.00 seconds
==> passed

Test 2: check that exactly one SAP object created per WordNet objec
t
==> passed

Test 3a-c: timing sap() and distance() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:   41080.80
     -  student   solution calls per second:   40970.20
     -  reference / student ratio:                1.00

=> passed    student <= 10000x reference
```

```
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> passed    student <=    10x reference
=> passed    student <=     5x reference

Test 4: timing isNoun() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:  258790.00
     -  student   solution calls per second:  249556.00
     -  reference / student ratio:                 1.04
     -  allowed ratio:                             2.00
==> passed

Total: 8/8 tests passed!

================================================================



*****************************************************************
**********
*  timing (with reference SAP.java and WordNet.java)
*****************************************************************
**********

Timing Outcast
*-----------------------------------------------------------
Running 1 total tests.

1.45 seconds to build WordNet

Computing time to find outcasts of various outcast files.
Total time must not exceed 1 seconds.

    filename       N     time
------------------------------
   outcast4.txt    4     0.01
   outcast5.txt    5     0.00
  outcast5a.txt    5     0.00
   outcast5.txt    5     0.00
   outcast7.txt    7     0.00
   outcast8.txt    8     0.00
  outcast8a.txt    8     0.00
  outcast8b.txt    8     0.00
```

```
  outcast8c.txt    8     0.00
   outcast9.txt    9     0.00
  outcast9a.txt    9     0.01
  outcast10.txt   10     0.01
 outcast10a.txt   10     0.01
  outcast11.txt   11     0.01
  outcast12.txt   12     0.01
 outcast12a.txt   12     0.01
  outcast20.txt   20     0.02
  outcast29.txt   29     0.04
=> passed, total elapsed time: 0.15

Total: 1/1 tests passed!

================================================================
```

Submission

Submission time	Mon-16-Nov 14:20:25
Raw Score	103.12 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     Potential bugs found.
API:          PASSED

Correctness:  35/35 tests passed
Memory:       4/4 tests passed
Timing:       18/16 tests passed

Aggregate score: 103.12% [Correctness: 65%, Memory: 10%, Timing: 25
%, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 28K
-rw-r--r-- 1 1.2K Nov 16 22:21 Outcast.java
-rw-r--r-- 1 8.0K Nov 16 22:21 SAP.java
-rw-r--r-- 1 6.6K Nov 16 22:21 WordNet.java
-rw-r--r-- 1 4.6K Nov 16 22:21 studentSubmission.zip


************************************************************************
***********
*  compiling
************************************************************************
***********


% javac SAP.java
*------------------------------------------------------------


================================================================


% javac WordNet.java
*------------------------------------------------------------


================================================================


% javac Outcast.java
*------------------------------------------------------------


================================================================



% checkstyle SAP.java WordNet.java Outcast.java
*------------------------------------------------------------
SAP.java:2:8: Unused import statement for 'edu.princeton.cs.algs4.G
raph'.
SAP.java:8:8: Unused import statement for 'java.util.HashSet'.
SAP.java:9:8: Unused import statement for 'java.util.List'.
WordNet.java:9:8: Unused import statement for 'java.util.Stack'.
Checkstyle ends with 4 errors.
```

```
================================================================


% findbugs *.class
*---------------------------------------------------------
M P UPM_UNCALLED_PRIVATE_METHOD UPM: Private method SAP$BFS.bfs(Dig
raph, int) is never called  At SAP.java:[lines 67-70]
Warnings generated: 1
================================================================


Testing the APIs of your programs.
*---------------------------------------------------------
SAP:

WordNet:

Outcast:

================================================================


**************************************************************
***********
*  correctness
**************************************************************
***********

Testing methods in SAP
*---------------------------------------------------------
Running 19 total tests.

Test 1: test length() and ancestor() on fixed digraphs
  *  digraph1.txt
  *  digraph2.txt
  *  digraph3.txt
  *  digraph4.txt
  *  digraph5.txt
  *  digraph6.txt
  *  digraph9.txt
==> passed

Test 2: check length() and ancestor() on WordNet digraph
  *  100 random vertex pairs in digraph-wordnet.txt
```

```
==> passed

Test 3: check length() and ancestor() on directed paths
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 4: check length() and ancestor() on directed cycles
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 5: check length() and ancestor() on complete graphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 6: check length() and ancestor() on tournament digraphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 7: check length() and ancestor() on complete binary trees
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 8: check length() and ancestor() on random DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
```

```
==> passed

Test 9: check length() and ancestor() on random rooted-in DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 10: check length() and ancestor() on random rooted-out DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 11: check length() and ancestor() on random rooted-in trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
==> passed

Test 12: check length() and ancestor() on random rooted-out trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
==> passed

Test 13: check length() and ancestor() on random simple digraphs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 14: check whether two SAP objects can be created at the same t
ime
  *  digraph1.txt and digraph2.txt
  *  digraph3.txt and digraph4.txt
  *  digraph5.txt and digraph6.txt
  *  digraph2.txt and digraph1.txt
==> passed

Test 15: check whether SAP is immutable
  *  digraph1.txt
  *  digraph2.txt
  *  digraph3.txt
```

```
  *  digraph4.txt
  *  digraph5.txt
  *  digraph6.txt
  *  digraph-ambiguous-ancestor.txt
==> passed

Test 16: test invalid arguments to length() and ancestor() in digra
ph1.txt
  *  v = -1, w = 0
  *  v = 0, w = -1
  *  v = 13, w = 0
  *  v = 0, w = 13
  *  v = -1 2 7 , w = 1 4 6 10 11
  *  v = 2 7 , w = -1 1 4 6 10 11
  *  v = 13 2 7 , w = 1 4 6 10 11
  *  v = 2 7 , w = 13 1 4 6 10 11
==> passed

Test 17: test length() and ancestor() with Iterable arguments
  *  100 random subsets of 1 and 1 vertices in digraph-wordnet.txt
  *  100 random subsets of 1 and 2 vertices in digraph-wordnet.txt
  *  100 random subsets of 2 and 1 vertices in digraph-wordnet.txt
  *  100 random subsets of 2 and 2 vertices in digraph-wordnet.txt
  *  100 random subsets of 3 and 11 vertices in digraph-wordnet.txt
  *  100 random subsets of 11 and 3 vertices in digraph-wordnet.txt
  *  100 random subsets of 0 and 5 vertices in digraph-wordnet.txt
  *  100 random subsets of 5 and 0 vertices in digraph-wordnet.txt
  *  100 random subsets of 0 and 0 vertices in digraph-wordnet.txt
==> passed

Test 18: Check Iterable version of length() and ancestor() with nul
l arguments
==> passed

Test 19: random calls to length() and ancestor(), with probabilitie
s
         p1 and p2, respectively
  *  random calls in a random rooted DAG (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
  *  random calls in a random digraph (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
==> passed
```

```
Total: 19/19 tests passed!

================================================================

*****************************************************************
**********
*  correctness (substituting reference SAP.java)
*****************************************************************
**********

Testing methods in WordNet
*-----------------------------------------------------------
Running 14 total tests.

Test 1: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 2: test distance() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 3: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 4: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
```

```
txt
==> passed

Test 5: test sap() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 6: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 7: check whether WordNet is immutable
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 8: check that constructor throws an IllegalArgumentException w
hen the input is not a rooted DAG
  *  synsets3.txt, hypernyms3InvalidTwoRoots.txt
more than one root
  *  synsets3.txt, hypernyms3InvalidCycle.txt
more than one root
  *  synsets6.txt, hypernyms6InvalidTwoRoots.txt
more than one root
  *  synsets6.txt, hypernyms6InvalidCycle.txt
more than one root
  *  synsets6.txt, hypernyms6InvalidCycle+Path.txt
more than one root
==> passed
```

```
Test 9: check that distance() and sap() throw an IllegalArgumentExc
eption
        when either argument is not a WordNet noun
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = invalid
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = b
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = eleventeen
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = INVALID
==> passed

Test 10: check isNoun()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 11: check nouns()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 12: check whether two WordNet objects can be created at the sa
me time
  *  synsets1 = synsets15.txt; hypernyms1 = hypernyms15Tree.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
  *  synsets1 = synsets.txt; hypernyms1 = hypernyms.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
==> passed

Test 13: call distance(), sap(), and isNoun() with null arguments
  *  synsets15.txt, hypernyms15Path.txt
==> passed

Test 14: random calls to isNoun(), distance(), and sap(), with
         probabilities p1, p2, and p3, respectively
  *  100 random calls (p1 = 0.5, p2 = 0.5, p3 = 0.0)
  *  100 random calls (p1 = 0.5, p2 = 0.0, p3 = 0.5)
  *  100 random calls (p1 = 0.0, p2 = 0.5, p3 = 0.5)
  *  100 random calls (p1 = 0.2, p2 = 0.4, p3 = 0.4)
==> passed


Total: 14/14 tests passed!
```

```
================================================================

**********************************************************************
**********
*  correctness (substituting reference SAP.java and WordNet.java)
**********************************************************************
**********

Testing methods in Outcast
*-----------------------------------------------------------
Running 2 total tests.

Test 1: test outcast() on WordNet digraph
        (synsets.txt and hypernyms.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast4.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8a.txt
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast9a.txt
  *  outcast10.txt
  *  outcast10a.txt
  *  outcast11.txt
  *  outcast12.txt
  *  outcast12a.txt
  *  outcast17.txt
  *  outcast20.txt
  *  outcast29.txt
==> passed

Test 2: test outcast() on WordNet subgraph
        (synsets50000-subgraph.txt and hypernyms50000-subgraph.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
```

```
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast10.txt
  *  outcast11.txt
==> passed


Total: 2/2 tests passed!

================================================================


*****************************************************************
**********
*  memory
*****************************************************************
**********

Computing memory of SAP
*-----------------------------------------------------------
Running 1 total tests.

student     memory    = 9827704 bytes
reference   memory    = 8347712 bytes
ratio                 = 1.18
maximum allowed ratio = 2.50

vertices = 82192
edges    = 84505

Total: 1/1 tests passed!

================================================================



Computing memory of WordNet
*-----------------------------------------------------------
Running 3 total tests.

Test 1a: test memory of WordNet object
  *  synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-
subgraph.txt
     -  student   memory = 663560 bytes
```

```
     -  reference memory = 1198784 bytes
     -  number vertices  = 1000
     -  number of edges  = 1008
     -  student / reference ratio  =  0.6
     -  maximum allowed rato       =  2.0

==> passed

Test 1b: test memory of WordNet object
  *  synsets = synsets5000-subgraph.txt; hypernyms = hypernyms5000-
subgraph.txt
     -  student   memory = 3310760 bytes
     -  reference memory = 5952136 bytes
     -  number vertices  = 5000
     -  number of edges  = 5059
     -  student / reference ratio  =  0.6
     -  maximum allowed rato       =  2.0

==> passed

Test 1c: test memory of WordNet object
  *  synsets = synsets10000-subgraph.txt; hypernyms = hypernyms1000
0-subgraph.txt
     -  student   memory = 7846112 bytes
     -  reference memory = 13853632 bytes
     -  number vertices  = 10000
     -  number of edges  = 10087
     -  student / reference ratio  =  0.6
     -  maximum allowed rato       =  2.0

==> passed

Total: 3/3 tests passed!

================================================================



*****************************************************************
***********
*  timing
*****************************************************************
***********
```

```
Timing SAP
*-----------------------------------------------------------
Running 7 total tests.

Test 1: time SAP constructor
   *  digraph-wordnet.txt
      -  student solution time =  0.04 seconds
      -  maximum allowed  time =  1.00 seconds
==> passed

Test 2a-c: time length() and ancestor() with random pairs of vertic
es
  *  digraph-wordnet.txt
     -  reference solution calls per second:  197984.67
     -  student   solution calls per second:  257920.67
     -  reference / student ratio:                 0.77

=> passed    student <= 25000x reference
=> passed    student <=  2500x reference
=> passed    student <=   250x reference
=> BONUS     student <=  10.0x reference

Test 3a-c: time length() and ancestor() with random sets of 5 verti
ces
  *  digraph-wordnet.txt
     -  reference solution calls per second:   42488.00
     -  student   solution calls per second:   64128.00
     -  reference / student ratio:                 0.66

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> BONUS     student <=  10.0x reference


Total: 9/7 tests passed!

===========================================================



*************************************************************
***********
*  timing (substituting reference SAP.java)
```

```
**********************************************************************
**********

Timing WordNet
*-----------------------------------------------------------
Running 8 total tests.

Test 1: timing WordNet constructor
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  student constructor time =  0.92 seconds
     -  maximum allowed     time = 10.00 seconds
==> passed

Test 2: check that exactly one SAP object created per WordNet objec
t
==> passed

Test 3a-c: timing sap() and distance() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:   47370.60
     -  student   solution calls per second:   47737.40
     -  reference / student ratio:                 0.99

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> passed    student <=    10x reference
=> passed    student <=     5x reference

Test 4: timing isNoun() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:  366525.00
     -  student   solution calls per second:  326179.00
     -  reference / student ratio:                1.12
     -  allowed ratio:                            2.00
==> passed

Total: 8/8 tests passed!

================================================================



**********************************************************************
```

```
***********
*  timing (with reference SAP.java and WordNet.java)
*****************************************************************
***********

Timing Outcast
*-----------------------------------------------------------
Running 1 total tests.

1.32 seconds to build WordNet

Computing time to find outcasts of various outcast files.
Total time must not exceed 1 seconds.

    filename       N     time
------------------------------
   outcast4.txt    4     0.00
   outcast5.txt    5     0.00
  outcast5a.txt    5     0.01
   outcast5.txt    5     0.00
   outcast7.txt    7     0.00
   outcast8.txt    8     0.00
  outcast8a.txt    8     0.00
  outcast8b.txt    8     0.00
  outcast8c.txt    8     0.00
   outcast9.txt    9     0.00
  outcast9a.txt    9     0.00
  outcast10.txt   10     0.00
 outcast10a.txt   10     0.00
  outcast11.txt   11     0.01
  outcast12.txt   12     0.01
 outcast12a.txt   12     0.01
  outcast20.txt   20     0.02
  outcast29.txt   29     0.03
=> passed, total elapsed time: 0.12

Total: 1/1 tests passed!

===========================================================
```

Submission	
Submission time	Mon-16-Nov 14:15:55
Raw Score	0.00 / 100.00

Feedback	Compilation: FAILED
	
	WordNet.java failed to compile, javac reports:
	
	WordNet.java:107: error: illegal start of expression
	if (!isValid()) throws java.lang.IllegalArgumentExcept
	ion();
	^
	1 error
	
	Outcast.java failed to compile, javac reports:
	
	./WordNet.java:107: error: illegal start of expression
	if (!isValid()) throws java.lang.IllegalArgumentExcept
	ion();
	^
	./WordNet.java:107: error: cannot find symbol
	if (!isValid()) throws java.lang.IllegalArgumentExcept
	ion();
	^
	symbol: class lang
	location: package java
	2 errors

Submission	
Submission time	Mon-16-Nov 14:09:57
Raw Score	0.00 / 100.00
Feedback	Compilation: PASSED API: FAILED

```
WordNet:


The following methods should be removed or made private:
  *  public boolean isValid()
```

Submission	
Submission time	Mon-16-Nov 11:31:34
Raw Score	97.55 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     Potential bugs found.
API:          PASSED

Correctness:  32/35 tests passed
Memory:       4/4 tests passed
Timing:       18/16 tests passed

Aggregate score: 97.55% [Correctness: 65%, Memory: 10%, Timing: 25%
, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 28K
-rw-r--r-- 1 1.2K Nov 16 19:33 Outcast.java
-rw-r--r-- 1 7.8K Nov 16 19:33 SAP.java
-rw-r--r-- 1 4.9K Nov 16 19:33 WordNet.java
-rw-r--r-- 1 4.1K Nov 16 19:33 studentSubmission.zip
```

```
**********************************************************************
**********
*  compiling
**********************************************************************
**********


% javac SAP.java
*-----------------------------------------------------------


================================================================


% javac WordNet.java
*-----------------------------------------------------------


================================================================


% javac Outcast.java
*-----------------------------------------------------------


================================================================



% checkstyle SAP.java WordNet.java Outcast.java
*-----------------------------------------------------------
SAP.java:2:8: Unused import statement for 'edu.princeton.cs.algs4.G
raph'.
SAP.java:8:8: Unused import statement for 'java.util.HashSet'.
SAP.java:9:8: Unused import statement for 'java.util.List'.
Checkstyle ends with 3 errors.
================================================================


% findbugs *.class
*-----------------------------------------------------------
M P UPM_UNCALLED_PRIVATE_METHOD UPM: Private method SAP$BFS.bfs(Dig
raph, int) is never called  At SAP.java:[lines 64-67]
Warnings generated: 1
================================================================
```

```
Testing the APIs of your programs.
*-----------------------------------------------------------
SAP:

WordNet:

Outcast:

================================================================


*****************************************************************
**********
*  correctness
*****************************************************************
**********

Testing methods in SAP
*-----------------------------------------------------------
Running 19 total tests.

Test 1: test length() and ancestor() on fixed digraphs
  *  digraph1.txt
  *  digraph2.txt
  *  digraph3.txt
  *  digraph4.txt
  *  digraph5.txt
  *  digraph6.txt
  *  digraph9.txt
==> passed

Test 2: check length() and ancestor() on WordNet digraph
  *  100 random vertex pairs in digraph-wordnet.txt
==> passed

Test 3: check length() and ancestor() on directed paths
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 4: check length() and ancestor() on directed cycles
```

```
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 5: check length() and ancestor() on complete graphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 6: check length() and ancestor() on tournament digraphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 7: check length() and ancestor() on complete binary trees
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 8: check length() and ancestor() on random DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 9: check length() and ancestor() on random rooted-in DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 10: check length() and ancestor() on random rooted-out DAGs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
```

```
  *  20 vertices, 100 edges
==> passed

Test 11: check length() and ancestor() on random rooted-in trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
==> passed

Test 12: check length() and ancestor() on random rooted-out trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
==> passed

Test 13: check length() and ancestor() on random simple digraphs
  *  5 vertices, 8 edges
  *  10 vertices, 40 edges
  *  20 vertices, 100 edges
==> passed

Test 14: check whether two SAP objects can be created at the same t
ime
  *  digraph1.txt and digraph2.txt
  *  digraph3.txt and digraph4.txt
  *  digraph5.txt and digraph6.txt
  *  digraph2.txt and digraph1.txt
==> passed

Test 15: check whether SAP is immutable
  *  digraph1.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 3
     -  student length before = 2
     -  student length after  = 1
  *  digraph2.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 2
     -  student length before = 4
     -  student length after  = 1
  *  digraph3.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 1
     -  student length before = -1
```

```
   -  student length after  = 1
  *  digraph4.txt
   -  after adding edges (v, 0) to G
   -  v = 0, w = 1
   -  student length before = 3
   -  student length after  = 1
  *  digraph5.txt
   -  after adding edges (v, 0) to G
   -  v = 0, w = 1
   -  student length before = -1
   -  student length after  = 1
  *  digraph6.txt
   -  after adding edges (v, 0) to G
   -  v = 0, w = 2
   -  student length before = 2
   -  student length after  = 1
  *  digraph-ambiguous-ancestor.txt
   -  after adding edges (v, 0) to G
   -  v = 0, w = 2
   -  student length before = 2
   -  student length after  = 1
==> FAILED

Test 16: test invalid arguments to length() and ancestor() in digra
ph1.txt
  *  v = -1, w = 0
  *  v = 0, w = -1
  *  v = 13, w = 0
  *  v = 0, w = 13
  *  v = -1 2 7 , w = 1 4 6 10 11
  *  v = 2 7 , w = -1 1 4 6 10 11
  *  v = 13 2 7 , w = 1 4 6 10 11
  *  v = 2 7 , w = 13 1 4 6 10 11
==> passed

Test 17: test length() and ancestor() with Iterable arguments
  *  100 random subsets of 1 and 1 vertices in digraph-wordnet.txt
  *  100 random subsets of 1 and 2 vertices in digraph-wordnet.txt
  *  100 random subsets of 2 and 1 vertices in digraph-wordnet.txt
  *  100 random subsets of 2 and 2 vertices in digraph-wordnet.txt
  *  100 random subsets of 3 and 11 vertices in digraph-wordnet.txt
  *  100 random subsets of 11 and 3 vertices in digraph-wordnet.txt
  *  100 random subsets of 0 and 5 vertices in digraph-wordnet.txt
  *  100 random subsets of 5 and 0 vertices in digraph-wordnet.txt
```

```
  *  100 random subsets of 0 and 0 vertices in digraph-wordnet.txt
==> passed

Test 18: Check Iterable version of length() and ancestor() with nul
l arguments
==> passed

Test 19: random calls to length() and ancestor(), with probabilitie
s
         p1 and p2, respectively
  *  random calls in a random rooted DAG (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
  *  random calls in a random digraph (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
==> passed


Total: 18/19 tests passed!

================================================================

****************************************************************
***********
*  correctness (substituting reference SAP.java)
****************************************************************
***********

Testing methods in WordNet
*-----------------------------------------------------------
Running 14 total tests.

Test 1: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 2: test distance() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
```

```
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 3: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 4: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 5: test sap() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 6: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 7: check whether WordNet is immutable
```

```
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 8: check that constructor throws an IllegalArgumentException w
hen the input is not a rooted DAG
  *  synsets3.txt, hypernyms3InvalidTwoRoots.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets3.txt, hypernyms3InvalidCycle.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets6.txt, hypernyms6InvalidTwoRoots.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets6.txt, hypernyms6InvalidCycle.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets6.txt, hypernyms6InvalidCycle+Path.txt
     -  failed to throw a java.lang.IllegalArgumentException
==> FAILED

Test 9: check that distance() and sap() throw an IllegalArgumentExc
eption
        when either argument is not a WordNet noun
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = invalid
     java.lang.NullPointerException
     SAP.length(SAP.java:90)
     WordNet.distance(WordNet.java:121)
     TestWordNet.testInvalidNoun(TestWordNet.java:372)
     TestWordNet.test9(TestWordNet.java:394)
     TestWordNet.main(TestWordNet.java:752)

     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = b
     java.lang.NullPointerException
     SAP.length(SAP.java:90)
     WordNet.distance(WordNet.java:121)
     TestWordNet.testInvalidNoun(TestWordNet.java:372)
     TestWordNet.test9(TestWordNet.java:395)
     TestWordNet.main(TestWordNet.java:752)

     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = eleventeen
     java.lang.NullPointerException
     SAP.length(SAP.java:90)
     WordNet.distance(WordNet.java:121)
     TestWordNet.testInvalidNoun(TestWordNet.java:372)
     TestWordNet.test9(TestWordNet.java:396)
```

```
     TestWordNet.main(TestWordNet.java:752)

     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = INVALID
     java.lang.NullPointerException
     SAP.length(SAP.java:90)
     WordNet.distance(WordNet.java:121)
     TestWordNet.testInvalidNoun(TestWordNet.java:372)
     TestWordNet.test9(TestWordNet.java:397)
     TestWordNet.main(TestWordNet.java:752)

     -  failed to throw a java.lang.IllegalArgumentException
==> FAILED

Test 10: check isNoun()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 11: check nouns()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 12: check whether two WordNet objects can be created at the sa
me time
  *  synsets1 = synsets15.txt; hypernyms1 = hypernyms15Tree.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
  *  synsets1 = synsets.txt; hypernyms1 = hypernyms.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
==> passed

Test 13: call distance(), sap(), and isNoun() with null arguments
  *  synsets15.txt, hypernyms15Path.txt
==> passed

Test 14: random calls to isNoun(), distance(), and sap(), with
         probabilities p1, p2, and p3, respectively
  *  100 random calls (p1 = 0.5, p2 = 0.5, p3 = 0.0)
  *  100 random calls (p1 = 0.5, p2 = 0.0, p3 = 0.5)
  *  100 random calls (p1 = 0.0, p2 = 0.5, p3 = 0.5)
  *  100 random calls (p1 = 0.2, p2 = 0.4, p3 = 0.4)
```

```
==> passed


Total: 12/14 tests passed!

================================================================

*****************************************************************
**********
*  correctness (substituting reference SAP.java and WordNet.java)
*****************************************************************
**********

Testing methods in Outcast
*-----------------------------------------------------------
Running 2 total tests.

Test 1: test outcast() on WordNet digraph
        (synsets.txt and hypernyms.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast4.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8a.txt
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast9a.txt
  *  outcast10.txt
  *  outcast10a.txt
  *  outcast11.txt
  *  outcast12.txt
  *  outcast12a.txt
  *  outcast17.txt
  *  outcast20.txt
  *  outcast29.txt
==> passed

Test 2: test outcast() on WordNet subgraph
        (synsets50000-subgraph.txt and hypernyms50000-subgraph.txt)
  *  outcast2.txt
```

```
  *  outcast3.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast10.txt
  *  outcast11.txt
==> passed


Total: 2/2 tests passed!

================================================================

****************************************************************
**********
*  memory
****************************************************************
**********

Computing memory of SAP
*-----------------------------------------------------------
Running 1 total tests.

student     memory    = 9827704 bytes
reference   memory    = 8347712 bytes
ratio                 = 1.18
maximum allowed ratio = 2.50

vertices = 82192
edges    = 84505

Total: 1/1 tests passed!

================================================================



Computing memory of WordNet
*-----------------------------------------------------------
Running 3 total tests.
```

```
Test 1a: test memory of WordNet object
  *  synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-
subgraph.txt
     -  student   memory = 567272 bytes
     -  reference memory = 1198784 bytes
     -  number vertices  = 1000
     -  number of edges  = 1008
     -  student / reference ratio  =  0.5
     -  maximum allowed rato       =  2.0

==> passed

Test 1b: test memory of WordNet object
  *  synsets = synsets5000-subgraph.txt; hypernyms = hypernyms5000-
subgraph.txt
     -  student   memory = 2807520 bytes
     -  reference memory = 5951304 bytes
     -  number vertices  = 5000
     -  number of edges  = 5059
     -  student / reference ratio  =  0.5
     -  maximum allowed rato       =  2.0

==> passed

Test 1c: test memory of WordNet object
  *  synsets = synsets10000-subgraph.txt; hypernyms = hypernyms1000
0-subgraph.txt
     -  student   memory = 6841216 bytes
     -  reference memory = 13853000 bytes
     -  number vertices  = 10000
     -  number of edges  = 10087
     -  student / reference ratio  =  0.5
     -  maximum allowed rato       =  2.0

==> passed

Total: 3/3 tests passed!

================================================================



**********************************************************************
```

```
**********
*  timing
*********************************************************************
**********

Timing SAP
*------------------------------------------------------------
Running 7 total tests.

Test 1: time SAP constructor
   *  digraph-wordnet.txt
      -  student solution time =  0.01 seconds
      -  maximum allowed  time =  1.00 seconds
==> passed

Test 2a-c: time length() and ancestor() with random pairs of vertic
es
  *  digraph-wordnet.txt
     -  reference solution calls per second:  194536.67
     -  student   solution calls per second:  267412.67
     -  reference / student ratio:                0.73

=> passed    student <= 25000x reference
=> passed    student <=  2500x reference
=> passed    student <=   250x reference
=> BONUS     student <=  10.0x reference

Test 3a-c: time length() and ancestor() with random sets of 5 verti
ces
  *  digraph-wordnet.txt
     -  reference solution calls per second:   41506.00
     -  student   solution calls per second:   64284.00
     -  reference / student ratio:                0.65

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> BONUS     student <=  10.0x reference


Total: 9/7 tests passed!

===========================================================
```

```
*****************************************************************
**********
*  timing (substituting reference SAP.java)
*****************************************************************
**********

Timing WordNet
*-----------------------------------------------------------
Running 8 total tests.

Test 1: timing WordNet constructor
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  student constructor time =  0.82 seconds
     -  maximum allowed     time = 10.00 seconds
==> passed

Test 2: check that exactly one SAP object created per WordNet objec
t
==> passed

Test 3a-c: timing sap() and distance() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:   46321.20
     -  student   solution calls per second:   49151.20
     -  reference / student ratio:                0.94

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> passed    student <=   100x reference
=> passed    student <=    10x reference
=> passed    student <=     5x reference

Test 4: timing isNoun() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:  380027.00
     -  student   solution calls per second:  332793.00
     -  reference / student ratio:                1.14
     -  allowed ratio:                            2.00
==> passed

Total: 8/8 tests passed!
```

```
==========================================================================



**************************************************************************
***********
*  timing (with reference SAP.java and WordNet.java)
**************************************************************************
***********

Timing Outcast
*----------------------------------------------------------
Running 1 total tests.

1.37 seconds to build WordNet

Computing time to find outcasts of various outcast files.
Total time must not exceed 1 seconds.

    filename       N     time
-----------------------------
   outcast4.txt    4     0.00
   outcast5.txt    5     0.01
  outcast5a.txt    5     0.00
   outcast5.txt    5     0.00
   outcast7.txt    7     0.00
   outcast8.txt    8     0.00
  outcast8a.txt    8     0.00
  outcast8b.txt    8     0.00
  outcast8c.txt    8     0.00
   outcast9.txt    9     0.00
  outcast9a.txt    9     0.00
  outcast10.txt   10     0.00
 outcast10a.txt   10     0.00
  outcast11.txt   11     0.01
  outcast12.txt   12     0.01
 outcast12a.txt   12     0.01
  outcast20.txt   20     0.02
  outcast29.txt   29     0.03
=> passed, total elapsed time: 0.11

Total: 1/1 tests passed!

==========================================================================
```

Submission	
Submission time	Sat-14-Nov 08:22:45
Raw Score	63.98 / 100.00
Feedback	See the Assessment Guide for information on how to interpret this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  22/35 tests passed
Memory:       3/4 tests passed
Timing:       10/16 tests passed

Aggregate score: 63.98% [Correctness: 65%, Memory: 10%, Timing: 25%
, Style: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 24K
-rw-r--r-- 1 1.2K Nov 14 16:24 Outcast.java
-rw-r--r-- 1 6.6K Nov 14 16:24 SAP.java
-rw-r--r-- 1 4.9K Nov 14 16:24 WordNet.java
-rw-r--r-- 1 3.8K Nov 14 16:24 studentSubmission.zip


*****************************************************************
***********
*  compiling
*****************************************************************
```

```
**********


% javac SAP.java
*---------------------------------------------------------

================================================================

% javac WordNet.java
*---------------------------------------------------------

================================================================

% javac Outcast.java
*---------------------------------------------------------

================================================================



% checkstyle SAP.java WordNet.java Outcast.java
*---------------------------------------------------------
SAP.java:7:8: Unused import statement for 'java.util.Arrays'.
SAP.java:9:8: Unused import statement for 'java.util.List'.
Checkstyle ends with 2 errors.
================================================================


% findbugs *.class
*---------------------------------------------------------
================================================================


Testing the APIs of your programs.
*---------------------------------------------------------
SAP:

WordNet:

Outcast:

================================================================
```

```
**********************************************************************
**********
*  correctness
**********************************************************************
**********

Testing methods in SAP
*-----------------------------------------------------------
Running 19 total tests.

Test 1: test length() and ancestor() on fixed digraphs
  *  digraph1.txt
  *  digraph2.txt
     -  failed on trial 30 of 36
     -  v = 2, w = 0
     -  student   length() = 2
     -  reference length() = 4
  *  digraph3.txt
  *  digraph4.txt
     -  failed on trial 2 of 100
     -  v = 2, w = 7
     -  student   length() = 2
     -  reference length() = 6
  *  digraph5.txt
     -  failed on trial 1 of 484
     -  v = 19, w = 10
     -  student   length() = 3
     -  reference length() = 5
  *  digraph6.txt
     -  failed on trial 10 of 64
     -  v = 6, w = 4
     -  student   length() = 2
     -  reference length() = 3
  *  digraph9.txt
     -  failed on trial 3 of 81
     -  v = 4, w = 8
     -  student   length() = 2
     -  reference length() = -1
==> FAILED

Test 2: check length() and ancestor() on WordNet digraph
  *  100 random vertex pairs in digraph-wordnet.txt
     -  failed on trial 1 of 100
     -  v = 18894, w = 23566
```

```
      -  student   length() = 12
      -  reference length() = 15
==> FAILED

Test 3: check length() and ancestor() on directed paths
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 4: check length() and ancestor() on directed cycles
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 5: check length() and ancestor() on complete graphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 6: check length() and ancestor() on tournament digraphs
  *  5
  *  10
  *  20
  *  50
==> passed

Test 7: check length() and ancestor() on complete binary trees
  *  5
  *  10
  *  20
  *  50
  *  100
==> passed

Test 8: check length() and ancestor() on random DAGs
  *  5 vertices, 8 edges
```

```
   -  failed on trial 4 of 25
   -  v = 2, w = 3
   -  student   length() = 2
   -  reference length() = -1
   -  failed on trial 1 of 100
*  10 vertices, 40 edges
   ancestor() inconsistent with length()
   -  failed on trial 11 of 100
   -  v = 2, w = 5
   -  student   length   = 2
   -  distance from 2 to 5 = 2147483647
   -  distance from 5 to 5 = 0
   -  student   ancestor = 5
   -  reference length   = 2
   -  reference ancestor = 0
   -  failed on trial 2 of 100
*  20 vertices, 100 edges
   ancestor() inconsistent with length()
   -  failed on trial 1 of 400
   -  v = 9, w = 3
   -  student   length   = 2
   -  distance from 9 to 3 = 2147483647
   -  distance from 3 to 3 = 0
   -  student   ancestor = 3
   -  reference length   = 2
   -  reference ancestor = 18
   -  failed on trial 1 of 100
==> FAILED

Test 9: check length() and ancestor() on random rooted-in DAGs
  *  5 vertices, 8 edges
     ancestor() inconsistent with length()
     -  failed on trial 12 of 25
     -  v = 1, w = 0
     -  student   length   = 2
     -  distance from 1 to 0 = 2147483647
     -  distance from 0 to 0 = 0
     -  student   ancestor = 0
     -  reference length   = 2
     -  reference ancestor = 1
     -  failed on trial 3 of 100
  *  10 vertices, 40 edges
     ancestor() inconsistent with length()
     -  failed on trial 6 of 100
```

```
     -  v = 0, w = 7
     -  student   length   = 2
     -  distance from 0 to 7 = 2147483647
     -  distance from 7 to 7 = 0
     -  student   ancestor = 7
     -  reference length   = 2
     -  reference ancestor = 3
     -  failed on trial 1 of 100
  *  20 vertices, 100 edges
     ancestor() inconsistent with length()
     -  failed on trial 4 of 400
     -  v = 10, w = 14
     -  student   length   = 2
     -  distance from 10 to 14 = 2147483647
     -  distance from 14 to 14 = 0
     -  student   ancestor = 14
     -  reference length   = 2
     -  reference ancestor = 10
     -  failed on trial 1 of 100
==> FAILED

Test 10: check length() and ancestor() on random rooted-out DAGs
  *  5 vertices, 8 edges
     ancestor() inconsistent with length()
     -  failed on trial 9 of 25
     -  v = 2, w = 0
     -  student   length   = 2
     -  distance from 2 to 0 = 2147483647
     -  distance from 0 to 0 = 0
     -  student   ancestor = 0
     -  reference length   = 2
     -  reference ancestor = 3
     -  failed on trial 1 of 100
  *  10 vertices, 40 edges
     ancestor() inconsistent with length()
     -  failed on trial 83 of 100
     -  v = 1, w = 7
     -  student   length   = 2
     -  distance from 1 to 7 = 2147483647
     -  distance from 7 to 7 = 0
     -  student   ancestor = 7
     -  reference length   = 2
     -  reference ancestor = 1
     -  failed on trial 2 of 100
```

```
  *  20 vertices, 100 edges
     ancestor() inconsistent with length()
     -  failed on trial 6 of 400
     -  v = 11, w = 3
     -  student   length   = 2
     -  distance from 11 to 3 = 2147483647
     -  distance from 3 to 3 = 0
     -  student   ancestor = 3
     -  reference length   = 2
     -  reference ancestor = 2
     -  failed on trial 1 of 100
==> FAILED

Test 11: check length() and ancestor() on random rooted-in trees
  *  5 vertices
  *  10 vertices
  *  20 vertices
==> passed

Test 12: check length() and ancestor() on random rooted-out trees
  *  5 vertices
     -  failed on trial 1 of 25
     -  v = 2, w = 4
     -  student   length() = 2
     -  reference length() = -1
     -  failed on trial 1 of 100
  *  10 vertices
     -  failed on trial 2 of 100
     -  v = 2, w = 3
     -  student   length() = 3
     -  reference length() = -1
     -  failed on trial 1 of 100
  *  20 vertices
     -  failed on trial 3 of 400
     -  v = 12, w = 17
     -  student   length() = 6
     -  reference length() = -1
     -  failed on trial 1 of 100
==> FAILED

Test 13: check length() and ancestor() on random simple digraphs
  *  5 vertices, 8 edges
     -  failed on trial 2 of 25
     -  v = 3, w = 2
```

```
   -  student   length() = 2
   -  reference length() = 3
   -  failed on trial 3 of 100
*  10 vertices, 40 edges
   ancestor() inconsistent with length()
   -  failed on trial 34 of 100
   -  v = 5, w = 6
   -  student   length   = 2
   -  distance from 5 to 6 = 3
   -  distance from 6 to 6 = 0
   -  student   ancestor = 6
   -  reference length   = 2
   -  reference ancestor = 4
   -  failed on trial 3 of 100
*  20 vertices, 100 edges
   ancestor() inconsistent with length()
   -  failed on trial 27 of 400
   -  v = 3, w = 18
   -  student   length   = 2
   -  distance from 3 to 18 = 3
   -  distance from 18 to 18 = 0
   -  student   ancestor = 18
   -  reference length   = 2
   -  reference ancestor = 3
   -  failed on trial 1 of 100
==> FAILED

Test 14: check whether two SAP objects can be created at the same t
ime
  *  digraph1.txt and digraph2.txt
     -  v = 0, w = 2
     -  (digraph2.txt) student   length() = 2
     -  (digraph2.txt) reference length() = 4
  *  digraph3.txt and digraph4.txt
     -  v = 0, w = 2
     -  (digraph4.txt) student   length() = 4
     -  (digraph4.txt) reference length() = 6
  *  digraph5.txt and digraph6.txt
     -  v = 0, w = 14
     -  (digraph5.txt) student   length() = 3
     -  (digraph5.txt) reference length() = 9
  *  digraph2.txt and digraph1.txt
     -  v = 0, w = 2
     -  (digraph2.txt) student   length() = 2
```

```
     -  (digraph2.txt) reference length() = 4
==> FAILED

Test 15: check whether SAP is immutable
  *  digraph1.txt
  *  digraph2.txt
  *  digraph3.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 9
     -  student ancestor before = 11
     -  student ancestor after  = 0
  *  digraph4.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 1
     -  student ancestor before = 8
     -  student ancestor after  = 0
  *  digraph5.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 7
     -  student ancestor before = 9
     -  student ancestor after  = 0
  *  digraph6.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 1
     -  student ancestor before = 1
     -  student ancestor after  = 0
  *  digraph-ambiguous-ancestor.txt
     -  after adding edges (v, 0) to G
     -  v = 0, w = 1
     -  student ancestor before = 1
     -  student ancestor after  = 0
==> FAILED

Test 16: test invalid arguments to length() and ancestor() in digra
ph1.txt
  *  v = -1, w = 0
  *  v = 0, w = -1
  *  v = 13, w = 0
  *  v = 0, w = 13
  *  v = -1 1 9 11 12 , w = 0 3 4 5 6 7 8
  *  v = 1 9 11 12 , w = -1 0 3 4 5 6 7 8
  *  v = 13 1 9 11 12 , w = 0 3 4 5 6 7 8
  *  v = 1 9 11 12 , w = 13 0 3 4 5 6 7 8
==> passed
```

```
Test 17: test length() and ancestor() with Iterable arguments
  *  100 random subsets of 1 and 1 vertices in digraph-wordnet.txt
     -  failed on trial 4 of 100
     -  v = 49432
     -  w = 44655
     -  student   length() = 15
     -  reference length() = 17
  *  100 random subsets of 1 and 2 vertices in digraph-wordnet.txt
     -  failed on trial 1 of 100
     -  v = 63313
     -  w = 1241 13003
     -  student   length() = 13
     -  reference length() = 14
  *  100 random subsets of 2 and 1 vertices in digraph-wordnet.txt
     -  failed on trial 5 of 100
     -  v = 2198 57742
     -  w = 28197
     -  student   length() = 12
     -  reference length() = 13
  *  100 random subsets of 2 and 2 vertices in digraph-wordnet.txt
     ancestor() inconsistent with length()
     -  failed on trial 2 of 100
     -  v = 17024 26640
     -  w = 21669 80088
     -  student   length   = 13
     -  distance from v to 44437 = 2147483647
     -  distance from w to 44437 = 4
     -  student   ancestor = 44437
     -  reference length   = 13
     -  reference ancestor = 38003
  *  100 random subsets of 3 and 11 vertices in digraph-wordnet.txt
     -  failed on trial 1 of 100
     -  v = 45526 46217 76391
     -  w size = 11
     -  student   length() = 8
     -  reference length() = 9
  *  100 random subsets of 11 and 3 vertices in digraph-wordnet.txt
     ancestor() inconsistent with length()
     -  failed on trial 13 of 100
     -  v size = 11
     -  w = 5905 53875 68875
     -  student   length   = 7
     -  distance from v to 19231 = 2147483647
```

```
     -  distance from w to 19231 = 1
     -  student   ancestor = 19231
     -  reference length   = 7
     -  reference ancestor = 60216
  *  100 random subsets of 0 and 5 vertices in digraph-wordnet.txt
  *  100 random subsets of 5 and 0 vertices in digraph-wordnet.txt
  *  100 random subsets of 0 and 0 vertices in digraph-wordnet.txt
==> FAILED

Test 18: Check Iterable version of length() and ancestor() with nul
l arguments
==> passed

Test 19: random calls to length() and ancestor(), with probabilitie
s
         p1 and p2, respectively
  *  random calls in a random rooted DAG (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
     ancestor() is not ancestor on shortest ancestral path
     -  failed on call 23 to ancestor()
     -  v = 14, w = 6
     -  student   ancestor = 6
     -  distance from 14 to 6 = 2147483647
     -  distance from 6 to 6 = 0
     -  reference ancestor = 17
     -  reference length   = 2
  *  random calls in a random digraph (20 vertices, 100 edges)
     (p1 = 0.5, p2 = 0.5)
     ancestor() is not ancestor on shortest ancestral path
     -  failed on call 53 to ancestor()
     -  v = 13, w = 18
     -  student   ancestor = 18
     -  distance from 13 to 18 = 3
     -  distance from 18 to 18 = 0
     -  reference ancestor = 13
     -  reference length   = 2
==> FAILED


Total: 8/19 tests passed!

================================================================

****************************************************************
```

```
**********
*  correctness (substituting reference SAP.java)
*********************************************************************
**********

Testing methods in WordNet
*-----------------------------------------------------------
Running 14 total tests.

Test 1: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 2: test distance() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 3: test distance() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 4: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets.txt; hypernyms = hypernyms.
txt
==> passed

Test 5: test sap() of all noun pairs
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
```

```
  *  synsets = synsets15.txt; hypernyms = hypernyms15Tree.txt
  *  synsets = synsets6.txt; hypernyms = hypernyms6TwoAncestors.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11AmbiguousAnces
tor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8WrongBFS.txt
  *  synsets = synsets11.txt; hypernyms = hypernyms11ManyPathsOneAn
cestor.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ManyAncestors.tx
t
==> passed

Test 6: test sap() of random noun pairs
  *  1000 pairs using synsets = synsets100-subgraph.txt; hypernyms
= hypernyms100-subgraph.txt
  *  1000 pairs using synsets = synsets500-subgraph.txt; hypernyms
= hypernyms500-subgraph.txt
  *  1000 pairs using synsets = synsets1000-subgraph.txt; hypernyms
= hypernyms1000-subgraph.txt
==> passed

Test 7: check whether WordNet is immutable
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
==> passed

Test 8: check that constructor throws an IllegalArgumentException w
hen the input is not a rooted DAG
  *  synsets3.txt, hypernyms3InvalidTwoRoots.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets3.txt, hypernyms3InvalidCycle.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets6.txt, hypernyms6InvalidTwoRoots.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets6.txt, hypernyms6InvalidCycle.txt
     -  failed to throw a java.lang.IllegalArgumentException
  *  synsets6.txt, hypernyms6InvalidCycle+Path.txt
     -  failed to throw a java.lang.IllegalArgumentException
==> FAILED

Test 9: check that distance() and sap() throw an IllegalArgumentExc
eption
        when either argument is not a WordNet noun
  *  synsets15.txt, hypernyms15Tree.txt, invalid noun = invalid
     java.lang.NullPointerException
```

```
   SAP.length(SAP.java:90)
   WordNet.distance(WordNet.java:120)
   TestWordNet.testInvalidNoun(TestWordNet.java:372)
   TestWordNet.test9(TestWordNet.java:394)
   TestWordNet.main(TestWordNet.java:752)

   -  failed to throw a java.lang.IllegalArgumentException
*  synsets15.txt, hypernyms15Tree.txt, invalid noun = b
   java.lang.NullPointerException
   SAP.length(SAP.java:90)
   WordNet.distance(WordNet.java:120)
   TestWordNet.testInvalidNoun(TestWordNet.java:372)
   TestWordNet.test9(TestWordNet.java:395)
   TestWordNet.main(TestWordNet.java:752)

   -  failed to throw a java.lang.IllegalArgumentException
*  synsets15.txt, hypernyms15Tree.txt, invalid noun = eleventeen
   java.lang.NullPointerException
   SAP.length(SAP.java:90)
   WordNet.distance(WordNet.java:120)
   TestWordNet.testInvalidNoun(TestWordNet.java:372)
   TestWordNet.test9(TestWordNet.java:396)
   TestWordNet.main(TestWordNet.java:752)

   -  failed to throw a java.lang.IllegalArgumentException
*  synsets15.txt, hypernyms15Tree.txt, invalid noun = INVALID
   java.lang.NullPointerException
   SAP.length(SAP.java:90)
   WordNet.distance(WordNet.java:120)
   TestWordNet.testInvalidNoun(TestWordNet.java:372)
   TestWordNet.test9(TestWordNet.java:397)
   TestWordNet.main(TestWordNet.java:752)

   -  failed to throw a java.lang.IllegalArgumentException
==> FAILED

Test 10: check isNoun()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 11: check nouns()
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
```

```
  *  synsets = synsets15.txt; hypernyms = hypernyms15Path.txt
  *  synsets = synsets8.txt; hypernyms = hypernyms8ModTree.txt
==> passed

Test 12: check whether two WordNet objects can be created at the sa
me time
  *  synsets1 = synsets15.txt; hypernyms1 = hypernyms15Tree.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
  *  synsets1 = synsets.txt; hypernyms1 = hypernyms.txt
     synsets2 = synsets15.txt; hypernyms2 = hypernyms15Path.txt
==> passed

Test 13: call distance(), sap(), and isNoun() with null arguments
  *  synsets15.txt, hypernyms15Path.txt
==> passed

Test 14: random calls to isNoun(), distance(), and sap(), with
         probabilities p1, p2, and p3, respectively
  *  100 random calls (p1 = 0.5, p2 = 0.5, p3 = 0.0)
  *  100 random calls (p1 = 0.5, p2 = 0.0, p3 = 0.5)
  *  100 random calls (p1 = 0.0, p2 = 0.5, p3 = 0.5)
  *  100 random calls (p1 = 0.2, p2 = 0.4, p3 = 0.4)
==> passed


Total: 12/14 tests passed!

================================================================

*****************************************************************
**********
*  correctness (substituting reference SAP.java and WordNet.java)
*****************************************************************
**********

Testing methods in Outcast
*-----------------------------------------------------------
Running 2 total tests.

Test 1: test outcast() on WordNet digraph
        (synsets.txt and hypernyms.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast4.txt
```

```
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8a.txt
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast9a.txt
  *  outcast10.txt
  *  outcast10a.txt
  *  outcast11.txt
  *  outcast12.txt
  *  outcast12a.txt
  *  outcast17.txt
  *  outcast20.txt
  *  outcast29.txt
==> passed

Test 2: test outcast() on WordNet subgraph
        (synsets50000-subgraph.txt and hypernyms50000-subgraph.txt)
  *  outcast2.txt
  *  outcast3.txt
  *  outcast5.txt
  *  outcast5a.txt
  *  outcast7.txt
  *  outcast8.txt
  *  outcast8b.txt
  *  outcast8c.txt
  *  outcast9.txt
  *  outcast10.txt
  *  outcast11.txt
==> passed


Total: 2/2 tests passed!

================================================================

*****************************************************************
**********
*  memory
*****************************************************************
**********
```

```
Computing memory of SAP
*-----------------------------------------------------------
Running 1 total tests.

student     memory    = 21485144 bytes
reference   memory    = 8347488 bytes
ratio                 = 2.57
maximum allowed ratio = 2.50

vertices = 82192
edges    = 84505

Total: 0/1 tests passed!

================================================================



Computing memory of WordNet
*-----------------------------------------------------------
Running 3 total tests.

Test 1a: test memory of WordNet object
  *  synsets = synsets1000-subgraph.txt; hypernyms = hypernyms1000-
subgraph.txt
     -  student   memory = 548872 bytes
     -  reference memory = 1198784 bytes
     -  number vertices  = 1000
     -  number of edges  = 1008
     -  student / reference ratio  =  0.5
     -  maximum allowed rato       =  2.0

==> passed

Test 1b: test memory of WordNet object
  *  synsets = synsets5000-subgraph.txt; hypernyms = hypernyms5000-
subgraph.txt
     -  student   memory = 2717136 bytes
     -  reference memory = 5951912 bytes
     -  number vertices  = 5000
     -  number of edges  = 5059
     -  student / reference ratio  =  0.5
     -  maximum allowed rato       =  2.0
```

```
==> passed

Test 1c: test memory of WordNet object
  *  synsets = synsets10000-subgraph.txt; hypernyms = hypernyms1000
0-subgraph.txt
     -  student   memory = 6661024 bytes
     -  reference memory = 13853264 bytes
     -  number vertices  = 10000
     -  number of edges  = 10087
     -  student / reference ratio  =  0.5
     -  maximum allowed rato       =  2.0

==> passed

Total: 3/3 tests passed!

================================================================



*****************************************************************
***********
*  timing
*****************************************************************
***********

Timing SAP
*-----------------------------------------------------------
Running 7 total tests.

Test 1: time SAP constructor
   *  digraph-wordnet.txt
      -  student solution time =  0.06 seconds
      -  maximum allowed  time =  1.00 seconds
==> passed

Test 2a-c: time length() and ancestor() with random pairs of vertic
es
  *  digraph-wordnet.txt
     -  reference solution calls per second:  178802.00
     -  student   solution calls per second:     108.67
     -  reference / student ratio:              1645.42
```

```
=> passed    student <= 25000x reference
=> passed    student <=  2500x reference
=> FAILED    student <=   250x reference

Test 3a-c: time length() and ancestor() with random sets of 5 verti
ces
  *  digraph-wordnet.txt
     -  reference solution calls per second:   38701.33
     -  student   solution calls per second:     300.67
     -  reference / student ratio:               128.72

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> FAILED    student <=   100x reference


Total: 5/7 tests passed!

================================================================
```

```
*****************************************************************
**********
*  timing (substituting reference SAP.java)
*****************************************************************
**********

Timing WordNet
*-----------------------------------------------------------
Running 8 total tests.

Test 1: timing WordNet constructor
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  student constructor time =  0.86 seconds
     -  maximum allowed     time = 10.00 seconds
==> passed

Test 2: check that exactly one SAP object created per WordNet objec
t
==> FAILED

Test 3a-c: timing sap() and distance() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
```

```
     -  reference solution calls per second:   41637.40
     -  student   solution calls per second:      99.00
     -  reference / student ratio:               420.58

=> passed    student <= 10000x reference
=> passed    student <=  1000x reference
=> FAILED    student <=   100x reference
=> FAILED    student <=    10x reference
=> FAILED    student <=     5x reference

Test 4: timing isNoun() with random nouns
  *  synsets = synsets.txt; hypernyms = hypernyms.txt
     -  reference solution calls per second:  271056.00
     -  student   solution calls per second:  248778.00
     -  reference / student ratio:                 1.09
     -  allowed ratio:                             2.00
==> passed

Total: 4/8 tests passed!

================================================================


****************************************************************
***********
*  timing (with reference SAP.java and WordNet.java)
****************************************************************
***********

Timing Outcast
*-----------------------------------------------------------
Running 1 total tests.

1.45 seconds to build WordNet

Computing time to find outcasts of various outcast files.
Total time must not exceed 1 seconds.

    filename       N     time
------------------------------
   outcast4.txt    4     0.00
   outcast5.txt    5     0.00
  outcast5a.txt    5     0.00
```

```
   outcast5.txt    5     0.00
   outcast7.txt    7     0.00
   outcast8.txt    8     0.00
  outcast8a.txt    8     0.00
  outcast8b.txt    8     0.00
  outcast8c.txt    8     0.00
   outcast9.txt    9     0.00
  outcast9a.txt    9     0.00
  outcast10.txt   10     0.00
 outcast10a.txt   10     0.00
  outcast11.txt   11     0.01
  outcast12.txt   12     0.01
 outcast12a.txt   12     0.01
  outcast20.txt   20     0.02
  outcast29.txt   29     0.03
=> passed, total elapsed time: 0.12

Total: 1/1 tests passed!

================================================================
```

Submission	
Submission time	Sat-14-Nov 08:08:01
Raw Score	0.00 / 100.00
Feedback	Compilation: PASSED API: FAILED SAP: The following methods should be removed or made private: * public boolean isDirectAncestor(int,int)