Report 4

1. Reflection

In this assessment, I reviewed and practiced a lot of LinkedList data structures and object-oriented programming concepts such as inheritance, interfaces, abstract class, polymorphism, also some basic algorithms such as linklist traverse, append, clone, merge and reverse operations, etc. Also used UML and unittests diagram to form a clear logic structure for the project and test the function accordingly.

2. Advantages are of using a linked list in this application

As learned in 5008, linklist is more efficient in memory since it only needs memory of all nodes objects. But dynamic array (list in java) usually needs more memory and time to resize based on the limit of memory allocated.

Also linklist is more flexible in manipulation such as delete, merge and clone, since each node is separate and usually just need to alter the next pointers of the node to make the change.

linklist can be easily handle different types of nodes, which makes it a flexible data structure that can be adapted to different needs.

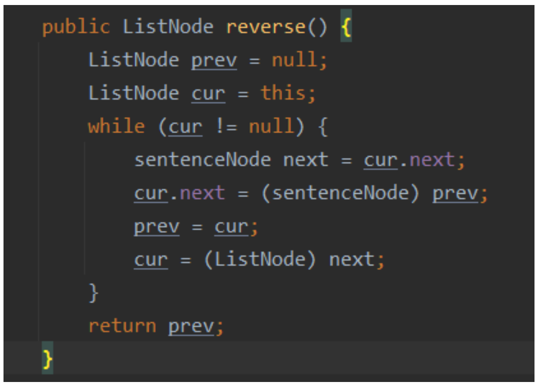
3. Do you think this could have been implemented without a linked list?

I think it is possible to do so with some data structures such as array or arraylist in java, however it might not be the best practice since it is not very flexible enough as mentioned in part2 of the report.

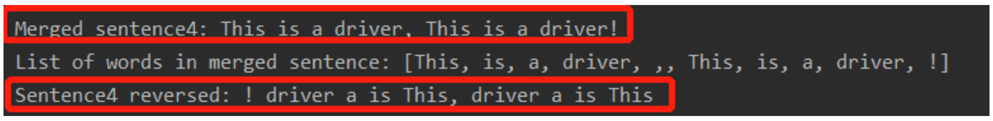
For example, a fixed size array is not efficient enough for appending words since size is limited and cannot hold long sentence, a dynamic array is usually not so memory efficient. linklist allows for efficient operations such as add, merge and reverse of elements. it can grow or shrink as needed. it is more memory efficient.

4. Extensions

- **Addeded reverse() function in Abstract class ListNode**

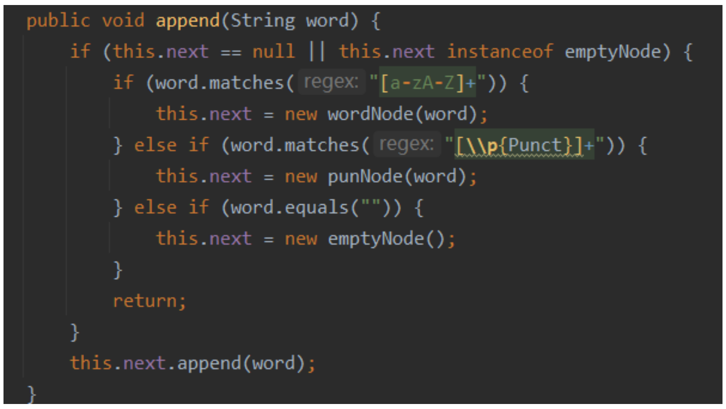


```java
public ListNode reverse() {
    ListNode prev = null;
    ListNode cur = this;
    while (cur != null) {
        sentenceNode next = cur.next;
        cur.next = (sentenceNode) prev;
        prev = cur;
        cur = (ListNode) next;
    }
    return prev;
}
```

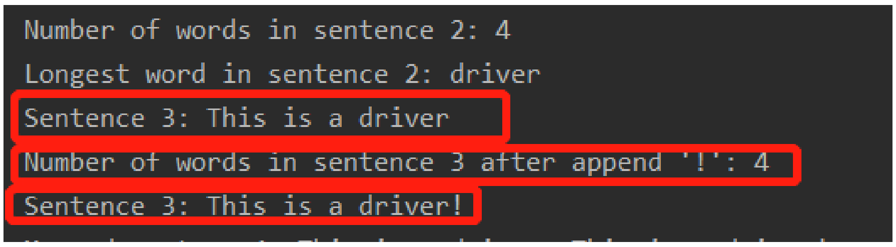


```
Merged sentence4: This is a driver, This is a driver!
List of words in merged sentence: [This, is, a, driver, ,, This, is, a, driver, !]
Sentence4 reversed: ! driver a is This, driver a is This
```

- **Added append(String s) function in Abstract class ListNode, using regular expression to decide the append Node type**

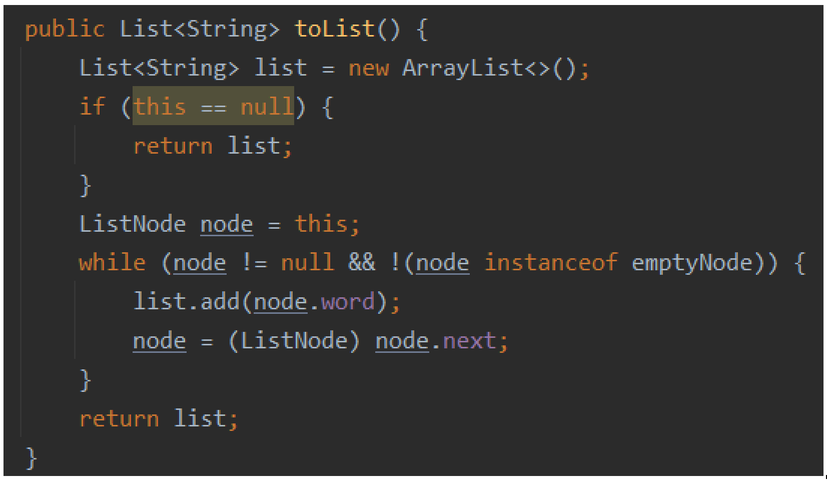


```java
public void append(String word) {
    if (this.next == null || this.next instanceof emptyNode) {
        if (word.matches( regex: "[a-zA-Z]+")) {
            this.next = new wordNode(word);
        } else if (word.matches( regex: "[\\p{Punct}]+")) {
            this.next = new punNode(word);
        } else if (word.equals("")) {
            this.next = new emptyNode();
        }
        return;
    }
    this.next.append(word);
}
```



```
Number of words in sentence 2: 4
Longest word in sentence 2: driver
Sentence 3: This is a driver
Number of words in sentence 3 after append '!': 4
Sentence 3: This is a driver!
```

- **Added toList() function in Abstract class ListNode**

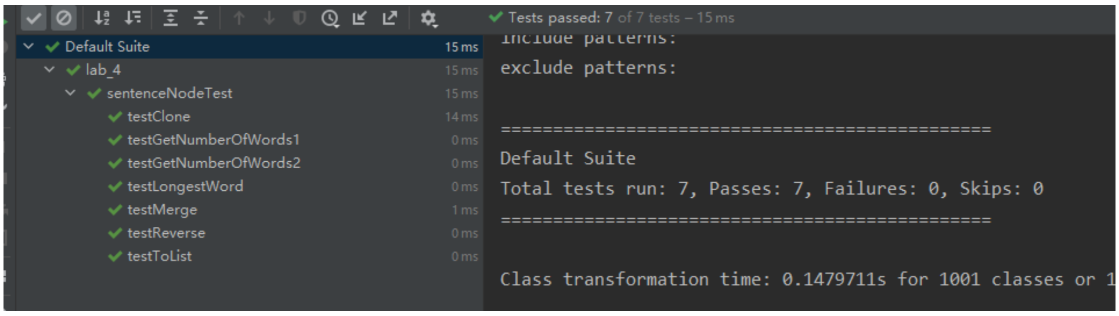


```java
public List<String> toList() {
    List<String> list = new ArrayList<>();
    if (this == null) {
        return list;
    }
    ListNode node = this;
    while (node != null && !(node instanceof emptyNode)) {
        list.add(node.word);
        node = (ListNode) node.next;
    }
    return list;
}
```

.

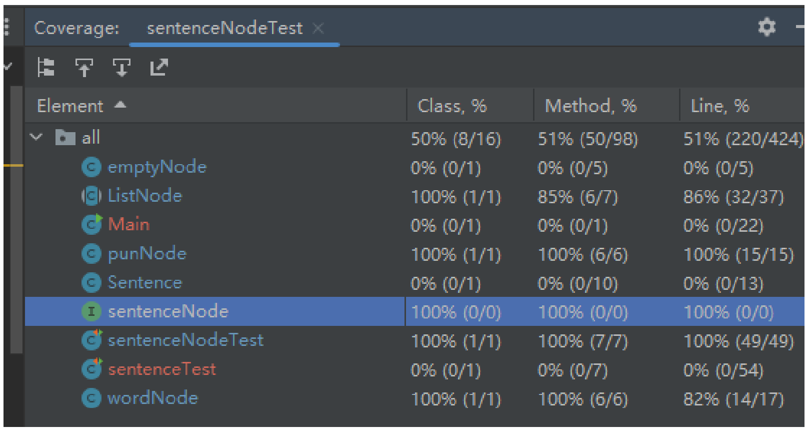
- **A more detailed tests for all functions in Sentence and SentenceNode including extensions, and reach 100% coverage.**

SentenceNode tests



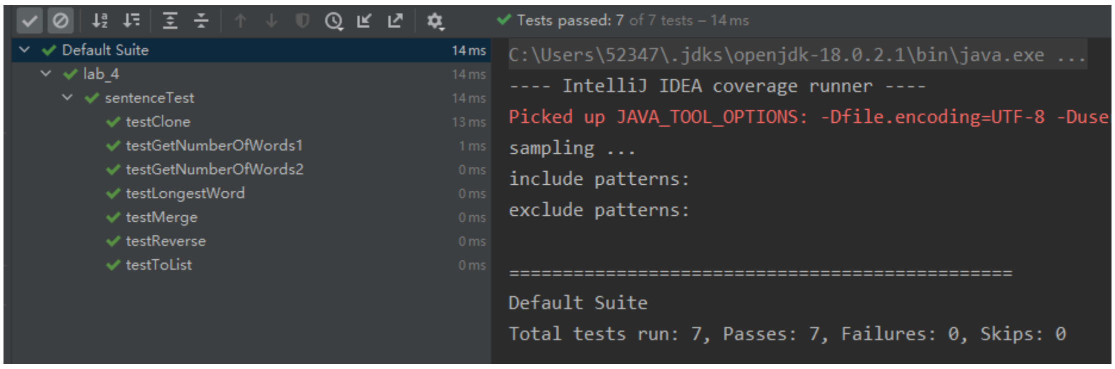
```
✓ Tests passed: 7 of 7 tests – 15 ms

✓ Default Suite                15 ms
  ✓ lab_4                      15 ms
    ✓ sentenceNodeTest         15 ms
      ✓ testClone              14 ms
      ✓ testGetNumberOfWords1   0 ms
      ✓ testGetNumberOfWords2   0 ms
      ✓ testLongestWord         0 ms
      ✓ testMerge               1 ms
      ✓ testReverse             0 ms
      ✓ testToList              0 ms
```

```
include patterns:
exclude patterns:

===============================================
Default Suite
Total tests run: 7, Passes: 7, Failures: 0, Skips: 0
===============================================

Class transformation time: 0.1479711s for 1001 classes or 1
```

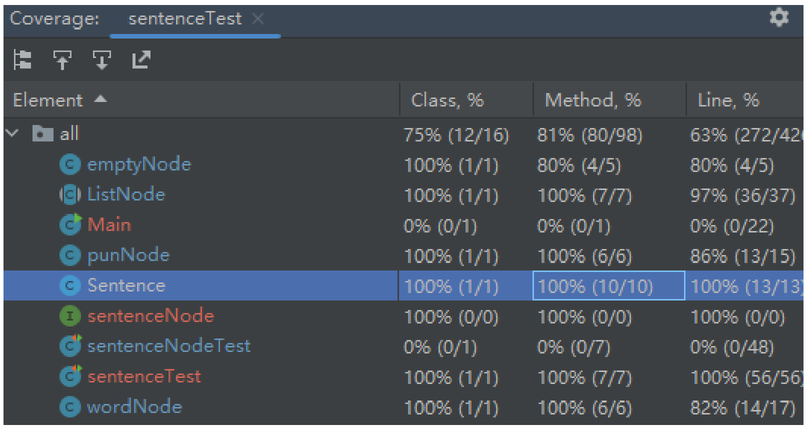

Coverage: sentenceNodeTest

Element	Class, %	Method, %	Line, %
all	50% (8/16)	51% (50/98)	51% (220/424)
emptyNode	0% (0/1)	0% (0/5)	0% (0/5)
ListNode	100% (1/1)	85% (6/7)	86% (32/37)
Main	0% (0/1)	0% (0/1)	0% (0/22)
punNode	100% (1/1)	100% (6/6)	100% (15/15)
Sentence	0% (0/1)	0% (0/10)	0% (0/13)
sentenceNode	100% (0/0)	100% (0/0)	100% (0/0)
sentenceNodeTest	100% (1/1)	100% (7/7)	100% (49/49)
sentenceTest	0% (0/1)	0% (0/7)	0% (0/54)
wordNode	100% (1/1)	100% (6/6)	82% (14/17)

Sentence tests


```
Tests passed: 7 of 7 tests – 14 ms

Default Suite                 14 ms
  lab_4                       14 ms
    sentenceTest              14 ms
      testClone               13 ms
      testGetNumberOfWords1    1 ms
      testGetNumberOfWords2    0 ms
      testLongestWord          0 ms
      testMerge                0 ms
      testReverse              0 ms
      testToList               0 ms
```

```
C:\Users\52347\.jdks\openjdk-18.0.2.1\bin\java.exe ...
---- IntelliJ IDEA coverage runner ----
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8 -Duse
sampling ...
include patterns:
exclude patterns:

===============================================
Default Suite
Total tests run: 7, Passes: 7, Failures: 0, Skips: 0
```


Coverage: sentenceTest

Element	Class, %	Method, %	Line, %
all	75% (12/16)	81% (80/98)	63% (272/42
emptyNode	100% (1/1)	80% (4/5)	80% (4/5)
ListNode	100% (1/1)	100% (7/7)	97% (36/37)
Main	0% (0/1)	0% (0/1)	0% (0/22)
punNode	100% (1/1)	100% (6/6)	86% (13/15)
Sentence	100% (1/1)	100% (10/10)	100% (13/13
sentenceNode	100% (0/0)	100% (0/0)	100% (0/0)
sentenceNodeTest	0% (0/1)	0% (0/7)	0% (0/48)
sentenceTest	100% (1/1)	100% (7/7)	100% (56/56
wordNode	100% (1/1)	100% (6/6)	82% (14/17)

- **Stylish and complete JavaDoc**
  I had added comments for all the classes, tests, functions to increase the readability and be close to a more standardized format.

5. Grading Statement

The project fulfills all basic requirement of the lab instructions, there is 90 points on it.
Extensions: 3~5 extensions are implemented for 10 points.
For the above reasons, I think this work should be graded as 100.