Algorithms and Data Structures in an Object Orientated Framework

ECS510U

Mini-Project: Implementing a String Collection Class

Sokolenko Maksym

16048215

Contents:

Design

Implementation – Array and Count

Implementation I used for this mini project is array and count implementation. The idea is that it uses an array to store objects which have word and the frequency with which it occurs. Array is initialised by a number that user puts it. This will create an array with its elements set to null and variable currentLen will be set to 0.

Count method will search the array in order to find the word and return its frequency. If it cannot find the word it will return 0.

Upon calling Add method, helper method position which will search the array in order to find the word and return its index. If the word couldn't be found it will return -1. If the word does occur then its frequency is incremented or a new element with the word will be created. Every time that it happens, currentLen is incremented and once it reaches the store length the helper method is called (increaseSize) which will double the length of the array.

When the remove method is called, the helper method position will search the array in order to find the word and return its index. If the word couldn't be found it will return -1. If the word doesn't occur, remove method won't do anything, otherwise remove method will proceed to reduce word`s frequency.

Testing the code

To test whether my code was actually working I used provided code WordTest0.java, WordTest1.java and an additional code that I wrote. During each test I decided to use seed 2017.

Count

Since both add and remove depend on count I decided to test it first. In order to test Count, I generated 300 words with WordTest0.java. In those 300 words I used filter to see if there were any words that occurred more than once:

Word **Con** occurred twice.

Word **Sor** occurred twice.

Word **Ror** occurred three times.

Word **Scur** occurred once.

Word **Hello** did not occur.

Using WordTest1.java and by putting initial amount of words 300 and the same seed the test gave me the same answers as filter in WordTest0.java. I used different amount of words to test whether the numbers would change and as predicted they did meaning that count method did work.

Add

In order to test add method I used WordTest0.java and WordTest1.java. Originally, I generated 20 words which gave me words:

```
pluyos
con
sor
wam
pexy
grolt
yitun
sahurvo
drezzewe
ronthe
biss
frorkin
pat
rul
jat
bayetroox
ath
tinne
```

daskeins
ad

I then used WordTest1.java with the amount of words generated 18. By testing words **Daskeins** and ad the result had to be: NOT generated. This suggests that add method does work.

```
Problems  @ Javadoc  Declaration  Console
WordTest1 [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (10 Dec 2017, 22:32:58
Enter a seed: 2017
Enter the number of words you wish to generate: 18
Enter words to test, empty line to exit
ad
"ad" NOT generated
```

```
Problems  @ Javadoc  Declaration  Console
WordTest1 [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (10 De
Enter a seed: 2017
Enter the number of words you wish to generate: 18
Enter words to test, empty line to exit
daskeins
"daskeins" NOT generated
```

Remove:

In order to test my remove method, I had an additional class RemoveTest.java which would show whether the code works. With seed 2017 and generating 300 words I already know that I will have word **Daskeins** occur once. I tested it by generating 300 words again removing **Daskeins** once that would mean that there are no more words left which should give me NOT generated answer. I also used the word Hello in order to have one test for the program to give me NOT generated answer for the word that did not occur in the first place.

```
Problems  @ Javadoc  Declaration  Console
RemoveTest [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (10 Dec 2017, 23:00:21)
Enter a seed: 2017
Enter the number of words you wish to generate initially: 300
Enter words to test, empty line to exit
daskeins
"daskeins" generated once
hello
"hello" NOT generated

Enter a word you wish to remove: daskeins
Enter a word you wish to remove:

Enter words to test, empty line to exit
daskeins
"daskeins" NOT generated
```

Testing Efficiency

Add method

In order to test how efficient my method is I generated 100, 1000, 10000, 100000, 1000000 words. I then added each number to it with the amount added being the same while initial amount being changed.The results are shown in the AddMethod table.

Remove method

To test the Remove method, I decided to take initial amount of words 100, 1000, 10000, 100000, 1000000. I then removed 100, 1000, 10000, 100000, 1000000 words from every single one of them. The results are shown in the RemoveMethod table.

Tables

Add method table:

Amount of words					
Words Added		Initial Amount		Time Taken	
		100		0 ms	
		1000		1 ms	
100		10000		3 ms	
		100000		13 ms	
		1000000		87 ms	
		100		8 ms	
		1000		4 ms	
1000		10000		25 ms	
		100000		264 ms	
		1000000		794 ms	
		100		246 ms	
		1000		168 ms	
10000		10000		308 ms	
		100000		2844 ms	
		1000000		9421 ms	
		100		10245 ms	
		1000		11647 ms	
100000		10000		12040 ms	
		100000		38968 ms	
		1000000		94210 ms	
		100		546813 ms	
		1000		560254 ms	
1000000		10000		632261 ms	
		100000		1183561 ms	
		1000000		3151655 ms	

Remove method table:

Amount of words					
Words Removed		Initial Amount		Time Taken	
		100		0 ms	
		1000		0 ms	
100		10000		5 ms	
		100000		47 ms	
		1000000		191 ms	
		100		1 ms	
		1000		0 ms	
1000		10000		30 ms	
		100000		191 ms	
		1000000		274 ms	
		100		13 ms	
		1000		28 ms	
10000		10000		1437 ms	
		100000		3596 ms	
		1000000		20741 ms	
		100		33 ms	
		1000		84 ms	
100000		10000		196 ms	
		100000		56834 ms	
		1000000		1438761 ms	
		100		140 ms	
		1000		436 ms	
1000000		10000		6721 ms	
		100000		938719 ms	
		1000000		1131758 ms	

Green colour represents tests that calculated the answer less than in 100 ms. In both cases answers that got less than 100ms had initial 100 words and 1000 words.

Evaluation

Nearly every test did not take long to get the answer, however, the 100000 and 1000000 generated words did take longer. This suggests that the best way of using this implementation is when the amount of words less than 100000 since the amount of words higher than that will take more time to produce answer. Higher amount of words generated, added or removed also uses a lot of cpu power so for bigger calculations it will require a powerful computer. To conclude, it is only efficient to use array and count implementation if the calculation itself is not a big number.

Source Code:

WordStoreImp.java

```
class WordStoreImp implements WordStore {
	WordFrequenc[] store;
	int numStrHeld;
	int currentLen = 0;

	public WordStoreImp (int num) {
		numStrHeld = num;
		store = new WordFrequenc[numStrHeld];
	}

	// returns position of String
	// if String doesn't occur - return -1
	private int position(String word) {
		for (int i=0; i<currentLen; i++) {
			if (store[i].getWord().equals(word)){
				return i;
			}
		}
		return -1;
	}

	// returns an int for the amount of times the String is stored
	public int count(String word) {
		for (int i=0; i<currentLen; i++) {
			if (store[i].getWord().equals(word)){
				return store[i].getFrequenc();
			}
		}
		return 0;
	}

	// takes a String and adds it to the collection, if word already in the
collection it will increment its frequency
	public void add(String word) {
		int posfWord = position(word);
		if (posfWord != -1){
			store[posfWord].setFrequenc(store[posfWord].getFrequenc()+1);
		}
		else if (currentLen < store.length) {
			store[currentLen] = new WordFrequenc(word, 1);
			currentLen++;
		}
		else {
			increaseSize();
			store[currentLen] = new WordFrequenc(word, 1);
			currentLen++;
		}
		return;
	}
```

```
	//double the size
	private void increaseSize() {
		WordFrequenc[] newStore = new WordFrequenc[currentLen*2];
		for (int i=0; i<currentLen; i++) {
			newStore[i] = store[i];
		}
		store = newStore;
		return;
	}

	// if String occurs, remove one occurrence of it
		// if String doesn't occur - do nothing
		public void remove(String word) {
			int posfWord = position(word);
			if (posfWord != -1) {
				store[posfWord] = store[currentLen-1];
				currentLen -= 1;
			}
			return;
		}
}//END class
```

RemoveTest.java

```
import java.util.Scanner;

class RemoveTest {

	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		WordGen.initialise(input);
		System.out.print("Enter the number of words you wish to generate
initially: ");
		int n = input.nextInt();
		WordStore words = new WordStoreImp(n);
		for (int i = 0; i < n; i++) {
			words.add(WordGen.make());
		}
		String line = input.nextLine();
		System.out.println("Enter words to test, empty line to exit");
		line = input.nextLine();
		while (!line.equals("")) {
			String[] wordlist = line.split(" ");
			for (int i = 0; i < wordlist.length; i++) {
				int count2 = words.count(wordlist[i]);
				System.out.print("\"" + wordlist[i] + "\" ");
				if (count2 == 0) {
					System.out.println("NOT generated");
				} else if (count2 == 1) {
```

```
                                System.out.println("generated once");
                        } else {
                                System.out.println("generated " + count2 + "
times ");
                        }
                }
                line = input.nextLine();
        }

        String toRemove;
        do {
                System.out.print("Enter a word you wish to remove: ");
                toRemove = input.nextLine();
                words.remove(toRemove);
        } while (!toRemove.equals(""));

        line = input.nextLine();
        System.out.println("Enter words to test, empty line to exit");
        line = input.nextLine();
        while (!line.equals("")) {
                String[] wordlist = line.split(" ");
                for (int i = 0; i < wordlist.length; i++) {
                        int count1 = words.count(wordlist[i]);
                        System.out.print("\"" + wordlist[i] + "\" ");
                        if (count1 == 0) {
                                System.out.println("NOT generated");
                        } else if (count1 == 1) {
                                System.out.println("generated once");
                        } else {
                                System.out.println("generated " + count1 + "
times ");
                        }
                }
                line = input.nextLine();
        }

        System.out.println("Enter words to test, empty line to exit");
        line = input.nextLine();
        while (!line.equals("")) {
                String[] wordlist = line.split(" ");
                for (int i = 0; i < wordlist.length; i++) {
                        int count2 = words.count(wordlist[i]);
                        System.out.print("\"" + wordlist[i] + "\" ");
                        if (count2 == 0) {
                                System.out.println("NOT generated");
                        } else if (count2 == 1) {
                                System.out.println("generated once");
                        } else {
                                System.out.println("generated " + count2 + "
times ");
                        }
                }
                line = input.nextLine();
        }
    }
}
```