```
#############################################

Student Name = Jitong Ding

#############################################




CSE017 Grading sheet for Jitong Ding

Homework Assignment Jumble

Total points                          maximum: 100

Completeness: All class/methods included (40)   [  40  ]

Compilation: Program compiles (20)              [  20  ]

Execution: Program executes properly (30)       [  21  ]

Style: Program obeys style rules (10)           [  10  ]

Subtotal                                           91
                                                _____

Late Penalty                                    [      ]

Total Points                                       91
                                                _____


#############################################
#############################################
```

-9 total execution

```
    /*
    CSE 17
    Jitong Ding
    jid221
5   Program #4 DEADLINE: November 16, 2018
    Program Description: Word Jumble Solver
    */

    import java.util.ArrayList;
10
    public class Dictionary{

      /** Private data filed */
      private int maxWordLength; /** An int of max word length */
15    private ArrayList<String>[] wordList;/** A generic ArrayList Array */

      /** Construct a new Dictionary initializing the maxWordLength and wordList */
      public Dictionary(int maxWordLength){
        this.maxWordLength = maxWordLength;
20      wordList = (ArrayList<String>[])new ArrayList[maxWordLength];
        for(int i = 0;i<maxWordLength;++i){
          wordList[i] = new ArrayList<String>();
        }
    }
25    /** Construct a new Dictionary seeting maxWordLength to 10 call the Dictionary
    (maxWordLength) construct*/
      public Dictionary(){
        this(10);
      }

30    /** A method to add the word intp the dictionary*/
      public void addEntry(String word){
        wordList[word.length()-1].add(word);
    }
      /** A method to return the boolean*/
35    public boolean lookup(String s){
        if(s.equals("a")){
          return true;
        }
        return lookup(s,0,(wordList[s.length()-1].size())-1);
40    }

      /** An recursive helper method to overload the method*/
      public boolean lookup(String s, int low, int high){
        if(high<low){    /** not found*/
45        return false;
        }
        int mid = (low+high) / 2;
        if(s.compareToIgnoreCase(wordList[s.length()-1].get(mid))<0){ /** search i
    n first half of array*/
          return lookup(s,low,mid-1);
50    }
        else if(s.equalsIgnoreCase(wordList[s.length()-1].get(mid))){ /** find it*
    /
          return true;
        }
        else{    /** search in second half of array*/
55        return lookup(s,mid+1,high);
    }
    }
    }
```

```java
/*
CSE 17
Jitong Ding
jid221
Program #4 DEADLINE: November 16, 2018
Program Description: Word Jumble Solver
*/

import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

public class Jumble{

    /** A method to return the String array of all s's permutations*/
    public static String[] findPermutations(String s){
        ArrayList<String> permutation = new ArrayList<>();
        findPermutations(s,"",permutation);
        String[] list = new String[permutation.size()];
        for(int i = 0; i<list.length;++i){
            list[i] = permutation.get(i);
        }
        return list;
    }

    /** A recursive helper method to permute the String s1*/
    private static void findPermutations(String s1,String s2, ArrayList<String> pe
rmutations){
        if(s2.length()==s1.length()){
            String s3="";
            for(int i=0;i<s2.length();i++){
                s3+=s1.charAt(Integer.parseInt(""+s2.charAt(i)));
            }
            for(int i=0;i<permutations.size();i++){
                if(permutations.get(i).equals(s3)){
                    return;
                }
            }
            permutations.add(s3);
            return;
        }
        ArrayList<String> notUsed =new ArrayList<String>();
        for(int i=0;i<s1.length();i++){
            if(s2.length()==0){
                notUsed.add(Integer.toString(i));
            }
            for(int j=0;j<s2.length();j++){
                if(i==Integer.parseInt(""+s2.charAt(j))){
                    break;
                }
                if(j==s2.length()-1){
                    notUsed.add(Integer.toString(i));
                }
            }
        }
        for(int i = 0; i<notUsed.size();++i){
            String s = notUsed.get(i);
            findPermutations(s1,s2+s,permutations);
        }
    }


    /** A main method */
    public static void main(String[] args) throws FileNotFoundException{
        Dictionary dictionary;
        File dictionaryFile;
        int wordNums = 0; /** Number of words in dictionary */
        int wordLength; /** Length of the longest word */
        /** An If statement to figure out whether there is one argument or two argum
ent */
        if(args.length != 1 && args.length != 2){
            System.out.println("Usage: java Jumble (maxWordLength) filename");
```

```java
            System.exit(0);
        }
        /** An if-else statement to initialize Dictionary and File */
        if(args.length==1){
            wordLength = 10;
            dictionary = new Dictionary();
            dictionaryFile = new File(args[0]);
        }
        else{
            wordLength = Integer.parseInt(args[0]);
            dictionary = new Dictionary(wordLength);
            dictionaryFile = new File(args[1]);
        }
        /** A try-catch to find the FileNotFoundException*/
        try{
            Scanner scan = new Scanner(dictionaryFile);
            while(scan.hasNextLine()){
                String Line = scan.nextLine();
                if(Line.length() > wordLength){
                    break;
                }
                dictionary.addEntry(Line);
                ++wordNums;
            }

            System.out.printf("Read in %d words\n",wordNums);

        }
        catch(FileNotFoundException ex){
            System.out.println("I don't find the file");
        }
        /** An while loop to repeat the whole thing*/
        while(true){
            System.out.println();
            System.out.println("Enter a scrambled word: ");
            Scanner input = new Scanner(System.in);
            String scrWord = input.nextLine();
            int fail = 0;
            ArrayList<String> fitWords = new ArrayList<>();
            if(scrWord.length() > wordLength){    /** A if statement to test if it is w
rong*/
                System.out.printf("Please enter a String whose length is between 1 to %d.\n",wordLength);
            }
            /** When "Enter" comes in, end the program*/
            else{
                if(scrWord.equals("")){
                    System.out.println();
                    System.out.println("Goodbye!");
                    break;
                }
                else{
                    for(int j = 0; j<findPermutations(scrWord).length;++j){
                        if(dictionary.lookup(findPermutations(scrWord)[j])){
                            fitWords.add(findPermutations(scrWord)[j]);
                        }    /** If fall == the length of the permutation array the string does
not fit*/
                        else{
                            fail++;
                            if(fail == (findPermutations(scrWord).length)){
                                System.out.printf("No words are formed from '%s'",scrWord);
                                System.out.println();
                            }
                        }
                    }
                }
            }
            /** A if statement to print out all of the fit words*/
            if(fitWords.size() != 0){
                System.out.printf("The words formed from '%s' are:\n", scrWord);
                for(int m = 0; m<fitWords.size();++m){
                    System.out.println(fitWords.get(m));
                }
            }
        }
    }
}
```

```
}
```

```
############################################
############################################


############# Compiled Result ##############


Source Code Compilation:


Dictionary.java:
20:
 warning:
 [unchecked] unchecked cast
   wordList = (ArrayList<String>[])new ArrayList[maxWordLength];
                         ^
  required:
 ArrayList<String>[]
 found:
   ArrayList[]
1 warning


############################################


############################################
############# Execution Result #############
############################################


############################################
Test1 output – testOutput1.txt


Command:
 java Jumble us10.dic < input1.txt >> testOutput1.txt 2>&1

Read in 82556 words

Enter a scrambled word:

The words formed from 'ginor' are:

groin
```

-5 (execution) never create more than one Scanner on System.in (instead while loop, so multiple are created)

```
Enter a scrambled word:

Exception in thread "main" java.util.NoSuchElementException:
 No line found
       at java.base/java.util.Scanner.nextLine(Scanner.java:
1651)
       at Jumble.main(Jumble.java:
106)


############################################
Test2 output – testOutput2.txt


Command:
 java Jumble 4 sparse.dic < input2.txt >> testOutput2.txt 2>&1\n

Read in 9 words

Enter a scrambled word:
```

-1 (execution) doesn't check input.hasNext to end program gracefully, once fixed all tests are correct

```
The words formed from 'tosc' are:

cost

Enter a scrambled word:

Exception in thread "main" java.util.NoSuchElementException:
 No line found
       at java.base/java.util.Scanner.nextLine(Scanner.java:
1651)
       at Jumble.main(Jumble.java:
106)


############################################
Test3 output – testOutput3.txt


Command:
 java Jumble 15 long.dic < input3.txt >> testOutput3.txt 2>&1\n

Read in 36 words

Enter a scrambled word:

The words formed from 'iblia' are:

alibi

Enter a scrambled word:

Exception in thread "main" java.util.NoSuchElementException:
 No line found
       at java.base/java.util.Scanner.nextLine(Scanner.java:
1651)
       at Jumble.main(Jumble.java:
106)


############################################
Test4 output – testOutput4.txt


Command:
 java Jumble nofile.dic >> testOutput4.txt 2>&1\n

I don't find the file

Enter a scrambled word:


Goodbye!


############################################
Test5 output – testOutput5.txt


Command:
 java Jumble four sparse.dic >> testOutput5.txt 2>&1\n
```

-3 (execution) unhandled exception

```
Exception in thread "main" java.lang.NumberFormatException:
 For input string:
 "four"
       at java.base/java.lang.NumberFormatException.forInputString(NumberFormat
Exception.java:
65)
       at java.base/java.lang.Integer.parseInt(Integer.java:
```

```
652)
        at java.base/java.lang.Integer.parseInt(Integer.java:
770)
        at Jumble.main(Jumble.java:
81)
```