Programming Assessed Exercise 2

Kevin Wu 0808148w
16/11/2018

My final program consists of five classes, one each for the monoalphabetic cipher, Vignere cipher, letter frequency table, an exceptions class and the main class.

One of my main assumptions that affected the program design was that the program must keep looping until the user enters a valid filename and keyword. I assume that the file must exist and is stored in the same directory as the code. As seen in Figure 1, if the user types a file name that does not end in P or C, an exception will be thrown and the user will be asked to enter the file name again (For example, a file called 'TestS.txt' might exist but we do not wish to load it for the purposes of this assignment). I also assumed that the keyword must only contain capital letters. The user is prompted to re-enter a new keyword should they enter any lower-case letters or special characters in the keyword.

I assumed that it would be best to wrap all functionalities related to the cipher and the letter frequency in their own separate classes. I initially assumed that the Vignere Cipher required a multidimensional A-Z array for encoding/decoding purposes. I realised this was unnecessary as the program worked can function with an inherited 1D alphabetical array. I was not sure whether to move the methods for editing file names/file reading/writing to separate classes. I kept them in main method class as a safe bet.

Lastly, the string format method is relied on heavily to create the letter frequency table. As this does not account for rounding differences, it's possible to get the wrong numbers and calculations. However, the difference is negligible.

My program meets the specification as close as possible; it will encode or decode using the monoalphabetic or Vignere cipher and produce a separate file showing the letter count of each letter in the output message.

To test that the program encrypted correctly, a file containing a simple message was loaded and printed, along with the alphabetic array, cipher array and output text to the console. Decoding the output file with the same keyword should return a file containing the same message as the initial file. In Figure 1, I try to encrypt a file, TestP.txt - containing the message 'GOODBYE' and keyword (BAT). The image also shows the interaction between the program and user if they were to input incorrect filename/keywords. Figure 2 shows the outcome of decoding 'TestC' using the same keyword which as expected, returned the same message that was contained in TestP.txt. I also tested the CommonP.txt/CommonC.txt files using the MA cipher and keyword (FLAMINGO) and got the same output as the example files. Printing the output to the console made me certain that the ciphers were encrypting/decoding properly.

To test that the program only encrypted capital letters, I loaded a file, SmallPP.txt which only contains lower-case letters, special chars and some capital letters at the start. In this scenario, the output file should only have capital letters substituted with all other letters remaining the same, as shown in Figure 3. Initially when I tested this, I noticed in SmallPF.txt, that the values in the frequency % and diff % columns read "NaN" (Figure 4). I realised that this happens when trying to divide 0 by a number. I altered the 'letterFrequency' class so that the method that calculated freq% defaults to 0 if the frequency at which the corresponding letter appears in the output text is 0. This also means there shouldn't be a NaN in the diff% column either (Figure 5).

Programming Assessed Exercise 2

Kevin Wu 0808148w
16/11/2018

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
1  /*
2   * This contains the main method for reading the file, writing the file and c
3   */
4  import java.io.FileWriter;
5  import java.util.Scanner;
6  import java.io.IOException;
7  import java.io.FileNotFoundException;
8  import java.io.FileReader;
9
10 public class AE2 {
11     public static void main(String[] args) {
12         Scanner s =  new Scanner(System.in);
13         String fN = "";
14         String fN2 = "";
15         String inputText = "";
16
17         /*
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (17 Nov
Please enter name of the input file:
TestPP
TestPP.txt (The system cannot find the file specified)
Please enter name of the input file:
TestS
You've entered an incorrect file name
File name must end in 'P' or 'C'
Please enter file name again:
TestP
Please enter keyword:
bat
Keyword can only contain capital letters!
Please enter keyword:
b@t
Keyword can only contain capital letters!
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Encrypted Order:    B A T C D E F G H I J K L M N O P Q R S U V W X Y Z
Normal Text:   GOODBYE
Encypted Text: FNNCAYD
```

Figure 1: Encrypting TestP.txt (Contents 'GOODBYE') using the MA cipher. The program checks that user inputs correct file name. It will ask the user to re-attempt if file does not exist or the file name ends in an incorrect letter. It also checks that the keyword only contains capital letters. To test that the program worked, I printed the alphabetic array, encrypted array, input text and output text to console and compared.

Programming Assessed Exercise 2

Kevin Wu 0808148w

16/11/2018

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
107             } els
108                 edString = encryptedString + inputFile.charAt(i); //i
109             }
110         }
111         System.out.print("Input Text:    " + inputFile);
112         System.out.print("Encypted Text: " + encryptedString);
113         return encryptedString;
114     }
115     /*@Param String toBeDecoded contains the lines of text from the file tha
116      *Method for decoding a string
117      *
118      */
119     public String decodeStringUsingMonoAlphabetArrayCipher(String toBeDecode
120         String decodedString = "";
121         for (int j = 0; j < toBeDecoded.length(); j++) {
122             if ((int)toBeDecoded.charAt(j) >=65 && (int) toBeDecoded.charAt(
123             int ofLetterJ = (int) toBeDecoded.charAt(j); //gets the int repr
124             int indexNumberOfCharInAlphabeticalArray = getNthIndex(ofLetterJ
125             int letterToBeAdded1 = getIntOfCharInAlphabeticArray(indexNumber
126             decodedString = decodedString + (char)letterToBeAdded1; //add th
127         } else {
128             decodedString = decodedString + toBeDecoded.charAt(j);//if charA
129             }
130         }
131         System.out.print("Input Text:    " + toBeDecoded);
132         System.out.print("Decoded Text: " + decodedString);
133         return decodedString;
134     }
135 }
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (17 No
Please enter name of the input file:
TestC
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Encrypted Order:    B A T C D E F G H I J K L M N O P Q R S U V W X Y Z
Input Text:    FNNCAYD
Decoded Text: GOODBYE
```

Figure 2: Decoding TestC.txt which was written in the proceeding step with the same keyword

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
11
12      public VignereCipher(String keyword) {
13          super(keyword);
14          //      this.multiAlphabeticalArray = new int[keyword.length()][ARRAY_
15          //      makeMultiAlphabeticArray();
16          this.vignereCipherArray = new int[keyword.length()][ARRAY_SIZE];//ciph
17          populateCipherArray(keyword);
18      }
19
20      /**
21       * @param Each char of the keyword will be kept in the first index of each
22       * This method is used to populate the cipher array which has been instant
23       */
24      public void populateCipherArray(String keyword) {
25          for (int i = 0; i < this.vignereCipherArray.length; i++) {//iterate th
26              int letter = (int) keyword.charAt(i); //get the int representatio
27              for (int j = 0; j<this.vignereCipherArray[i].length;j++) {//iterat
28                  this.vignereCipherArray[i][j] = letter; // start each row with
29                  if (letter == 90) { //we check once Z has been set, so that we
30                      letter = 64;
31                  }
32                  System.out.print((char) this.vignereCipherArray[i][j] + " ");
33                  letter++; //increment the letter so that each index gets the n
34              }
35              System.out.println();
36          }
37      }
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (17 Nov
Please enter name of the input file:
TestP
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
Input Text:    GOODBYE
Encypted Text: HOHEBRF
```

Figure 3: Encrypting TestP.txt using the Vignere cipher and keyword BAT. Note that the encrypted text is different compared when using the MA cipher previously

Programming Assessed Exercise 2

Kevin Wu 0808148w

16/11/2018

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
11
12      public VignereCipher(String keyword) {
13          super(keyword);
14          //      this.multiAlphabeticalArray = new int[keyword.length()][ARRAY_S
15          //      makeMultiAlphabeticArray();
16          this.vignereCipherArray = new int[keyword.length()][ARRAY_SIZE];//ciphe
17          populateCipherArray(keyword);
18      }
19
20      /**
21       * @param Each char of the keyword will be kept in the first index of each
22       * This method is used to populate the cipher array which has been instanti
23       */
24      public void populateCipherArray(String keyword) {
25          for (int i = 0; i < this.vignereCipherArray.length; i++) {//iterate thr
26              int letter = (int) keyword.charAt(i); //get the int representation
27              for (int j = 0; j<this.vignereCipherArray[i].length;j++) {//iterate
28                  this.vignereCipherArray[i][j] = letter; // start each row with
29                  if (letter == 90) { //we check once Z has been set, so that we
30                      letter = 64;
31                  }
32                  System.out.print((char) this.vignereCipherArray[i][j] + " ");
33                  letter++; //increment the letter so that each index gets the ne
34              }
35              System.out.println();
36          }
37      }
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (17 Nov 2
Please enter name of the input file:
TestC
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
Input Text:   HOHEBRF
Decoded Text: GOODBYE
```

Figure 4: Decoding TestC.txt using the Vignere Cipher and keyword (BAT). Note we get the same output as the contents in TestP.txt

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
39          } catch (AssEx2Exception e) {
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (17 Nov
Please enter name of the input file:
CommonP
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
Input Text:   THIS IS
A FILE
TO BE
ENCODED IN JAVA
WITH
SOME MORE
0123
TEXT AND
SOME
&&
SYMBOLS
**()))
THAT WILL
BE IGNORED!!
Encypted Text: UHBT IL
B FBME
MP BX
FNVPDXE IG KAOB
WBUH
LPMX NOKF
0123
TXYT TOD
LPMX
&&
TYFCOET
**()))
TABT PJLE
CE BHNHSEW!!
```

Figure 5: Encrypting CommonP.txt using the Vignere Cipher and keyword (BAT)

Programming Assessed Exercise 2

Kevin Wu 0808148w

16/11/2018

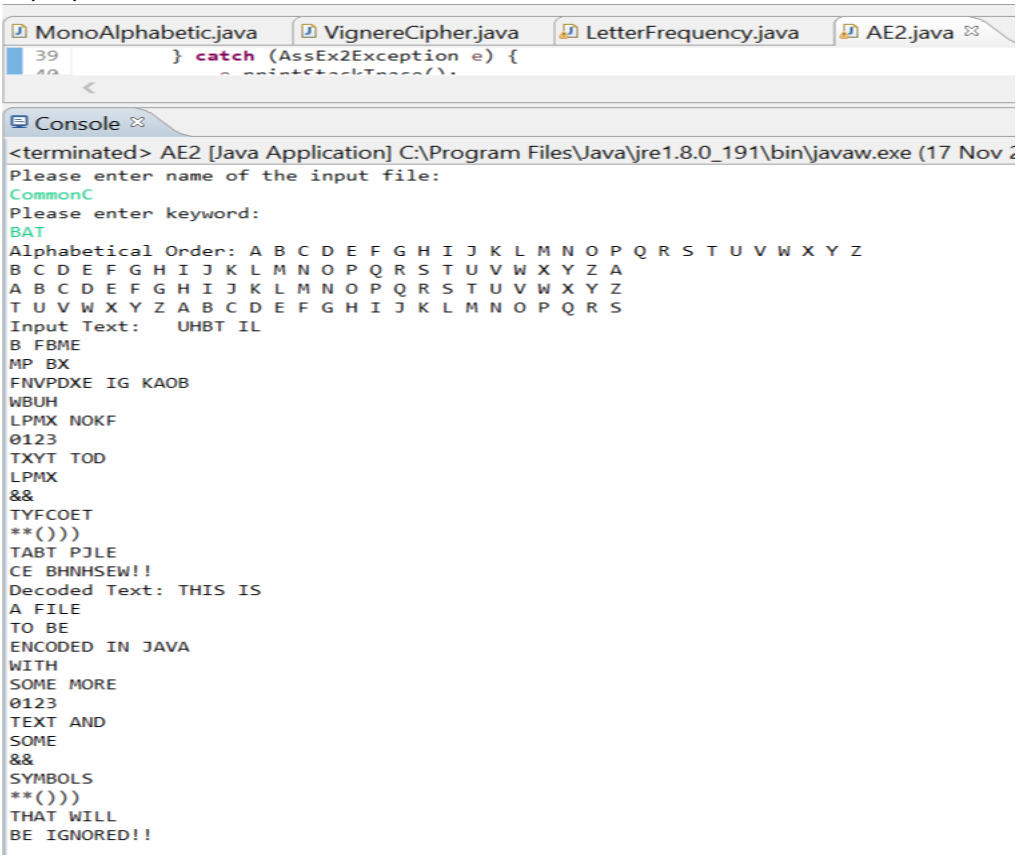

Figure 6: Decoding CommonC.txt using the Vignere Cipher and same keyword (BAT). Getting the same message as CommonP.txt so program working

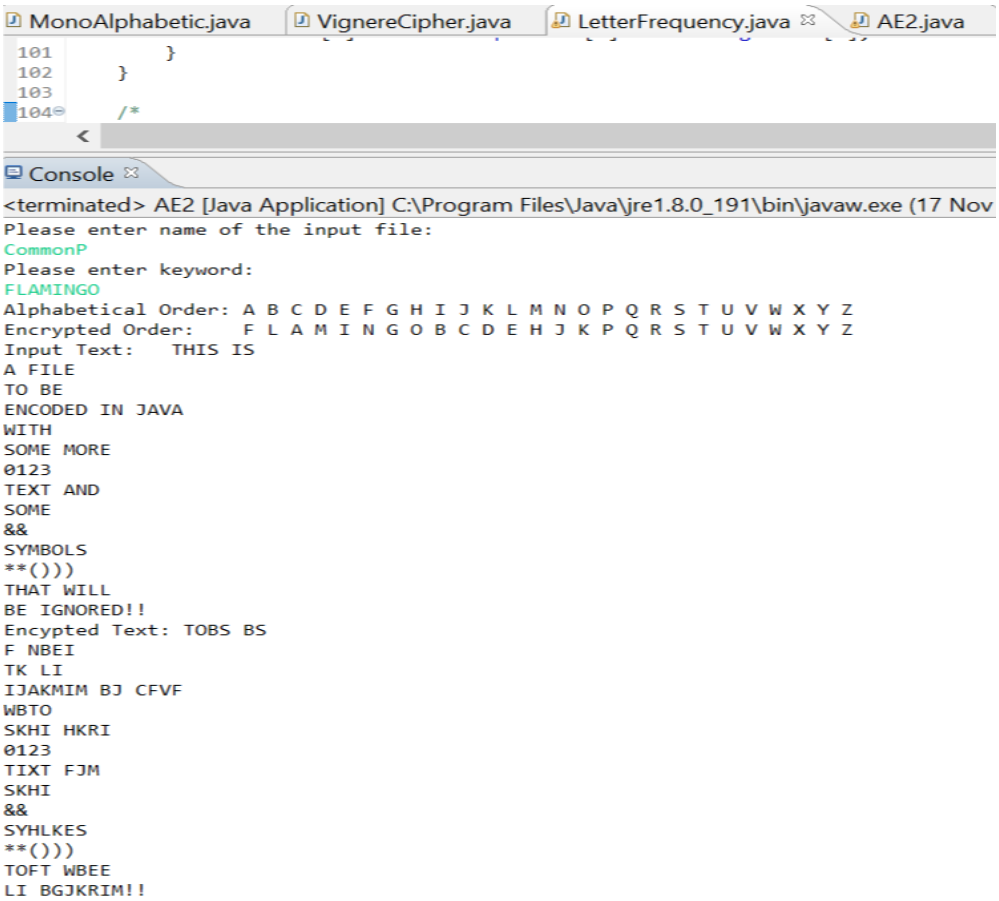

Figure 6: Encrypting CommonP.txt using the MA cipher and keyword (FLAMINGO)

Programming Assessed Exercise 2

Kevin Wu 0808148w

16/11/2018

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
101         }
102     }
103
104     /*
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (17 Nov
Please enter file name again:
CommonC
Please enter keyword:
FLAMINGO
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Encrypted Order:    F L A M I N G O B C D E H J K P Q R S T U V W X Y Z
Input Text:   TOBS BS
F NBEI
TK LI
IJAKMIM BJ CFVF
WBTO
SKHI HKRI
0123
TIXT FJM
SKHI
&&
SYHLKES
**()))
TOFT WBEE
LI BGJKRIM!!
Decoded Text: THIS IS
A FILE
TO BE
ENCODED IN JAVA
WITH
SOME MORE
0123
TEXT AND
SOME
&&
SYMBOLS
**()))
THAT WILL
BE IGNORED!!
```

Figure 7: Decoding CommonC.txt using the MA cipher and keyword (FLAMINGO). Getting the same message as CommonP.txt

MonoAlphabetic.java | VignereCipher.java | LetterFrequency.java | AE2.java

```
45          */
46          MonoAlphabetic array = new MonoAlphabetic(keyWord); //monoalphabetic
47  //      VignereCipher array = new VignereCipher(keyWord); //vignere cipher o
48
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (19 Nc
Please enter name of the input file:
SmallPP
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Encrypted Order:    B A T C D E F G H I J K L M N O P Q R S U V W X Y Z
Input Text:   THIS is
a file
to be
encoded in java
with
some more
0123
text and
some
&&
symbols
**()))
that will
be ignored!!
Encypted Text: SGHR is
a file
to be
encoded in java
with
some more
0123
text and
some
&&
symbols
**()))
that will
be ignored!!
```

Encrypting using MA cipher, keyword BAT

MonoAlphabetic.java | *VignereCipher.java | LetterFrequency.java | AE2.ja

```
17          populateCipherArray(keyword);
18      }
```

Console

```
<terminated> AE2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (19
Please enter name of the input file:
SmallPP
Please enter keyword:
BAT
Alphabetical Order: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
Input Text:   THIS is
a file
to be
encoded in java
with
some more
0123
text and
some
&&
symbols
**()))
that will
be ignored!!
Encypted Text: UHBT is
a file
to be
encoded in java
with
some more
0123
text and
some
&&
symbols
**()))
that will
```

Encoding using Vignere cipher, keyword BAT

Figure 8: Encrypting SmallPP.txt - a file that contains a mix of lower-case letters, upper-case letters and special characters. Note that only the capital letters are altered leaving the rest untouched.

Programming Assessed Exercise 2

Kevin Wu 0808148w
16/11/2018

```
LETTER ANALYSIS

Letter   Freq   Freq%   AvgFreq%    Diff
   A      0       NaN      8.20       NaN
   B      0       NaN      1.50       NaN
   C      0       NaN      2.80       NaN
   D      0       NaN      4.30       NaN
   E      0       NaN     12.70       NaN
   F      0       NaN      2.20       NaN
   G      0       NaN      2.00       NaN
   H      0       NaN      6.10       NaN
   I      0       NaN      7.00       NaN
   J      0       NaN      0.20       NaN
   K      0       NaN      0.80       NaN
   L      0       NaN      4.00       NaN
   M      0       NaN      2.40       NaN
   N      0       NaN      6.70       NaN
   O      0       NaN      7.50       NaN
   P      0       NaN      1.90       NaN
   Q      0       NaN      0.10       NaN
   R      0       NaN      6.00       NaN
   S      0       NaN      6.30       NaN
   T      0       NaN      9.10       NaN
   U      0       NaN      2.80       NaN
   V      0       NaN      1.00       NaN
   W      0       NaN      2.40       NaN
   X      0       NaN      0.20       NaN
   Y      0       NaN      2.00       NaN
   Z      0       NaN      0.10       NaN

The most frequent letter is A at NaN%
```

Figure 9: Letter Frequency table when encrypting SmallP.txt. NaN values for Freq% and Diff when trying to divide 0 by a number

SmallF - Notepad

File Edit Format View Help

```
LETTER ANALYSIS

Letter   Freq   Freq%   AvgFreq%    Diff
   A      0      0.00      8.20     -8.20
   B      0      0.00      1.50     -1.50
   C      0      0.00      2.80     -2.80
   D      0      0.00      4.30     -4.30
   E      0      0.00     12.70    -12.70
   F      0      0.00      2.20     -2.20
   G      0      0.00      2.00     -2.00
   H      0      0.00      6.10     -6.10
   I      0      0.00      7.00     -7.00
   J      0      0.00      0.20     -0.20
   K      0      0.00      0.80     -0.80
   L      0      0.00      4.00     -4.00
   M      0      0.00      2.40     -2.40
   N      0      0.00      6.70     -6.70
   O      0      0.00      7.50     -7.50
   P      0      0.00      1.90     -1.90
   Q      0      0.00      0.10     -0.10
   R      0      0.00      6.00     -6.00
   S      0      0.00      6.30     -6.30
   T      0      0.00      9.10     -9.10
   U      0      0.00      2.80     -2.80
   V      0      0.00      1.00     -1.00
   W      0      0.00      2.40     -2.40
   X      0      0.00      0.20     -0.20
   Y      0      0.00      2.00     -2.00
   Z      0      0.00      0.10     -0.10

The most frequent letter is Z at 0.00%
```

Figure 10: Correcting the NaN values to display 0. No capital letters counted in the output file