```
/**
 * Programming AE2
 * Creates and shows the cipher GUI
 */
public class AssEx2
{
	/**
	 * The main method
	 * @param args the arguments
	 */
	public static void main(String [] args)
	{
		CipherGUI CipherGUI = new CipherGUI();
		CipherGUI.setVisible(true);
	}
}
```



```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
//WHEN COMING BACK:: PRODUCE FREQ FILES
/**
 * Programming AE2
 * Class to display cipher GUI and listen for events
 */
public class CipherGUI extends JFrame implements ActionListener
{
	//instance variables which are the components
	private JPanel top, bottom, middle;
	private JButton monoButton, vigenereButton;
	private JTextField keyField, messageField;
	private JLabel keyLabel, messageLabel;
	private FileReader reader;
	private PrintWriter writer;
	private String keyword;
	//application instance variables
	//including the 'core' part of the text file filename
	//some way of indicating whether encoding or decoding is to be done
	private MonoCipher mcipher;
	private VCipher vcipher;
	private String filename;//file name user enters

	/**
	 * The constructor adds all the components to the frame
	 */
	public CipherGUI()
	{
		this.setSize(400,150);
		this.setLocation(100,100);
		this.setTitle("Cipher GUI");
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);
		this.layoutComponents();
	}

	/**
	 * Helper method to add components to the frame
	 */
	public void layoutComponents()
	{
		//top panel is yellow and contains a text field of 10 characters
		top = new JPanel();
		top.setBackground(Color.yellow);
		keyLabel = new JLabel("Keyword : ");
		top.add(keyLabel);

		//key field
		keyField = new JTextField(10);
		keyField.addActionListener(this);
		top.add(keyField);

		this.add(top,BorderLayout.NORTH);

		//middle panel is yellow and contains a text field of 10 charact
ers
		middle = new JPanel();
		middle.setBackground(Color.yellow);
		messageLabel = new JLabel("Message file : ");
		middle.add(messageLabel);
```

```
                messageField = new JTextField(10);
                middle.add(messageField);
                this.add(middle,BorderLayout.CENTER);

                //bottom panel is green and contains 2 buttons

                bottom = new JPanel();
                bottom.setBackground(Color.green);
                //create mono button and add it to the top panel
                monoButton = new JButton("Process Mono Cipher");
                monoButton.addActionListener(this);
                bottom.add(monoButton);
                //create vigenere button and add it to the top panel
                vigenereButton = new JButton("Process Vigenere Cipher");
                vigenereButton.addActionListener(this);
                bottom.add(vigenereButton);
                //add the top panel
                this.add(bottom,BorderLayout.SOUTH);
        }

        /**
         * Listen for and react to button press events
         * (use helper methods below)
         * @param e the event
         */
        public void actionPerformed(ActionEvent e)
        {
                if(!getKeyword())//if key word is invalid issue error message
                        {
                        JOptionPane.showMessageDialog(null, "keyword is invalid",
                        "Entry error", JOptionPane.ERROR_MESSAGE);
                        keyField.setText("");//clear field

                        }

                else if(!processFileName())//if file name is invalid issue error
 message
                {
                        JOptionPane.showMessageDialog(null, "File name is invalid",
                                        "Entry error", JOptionPane.ERROR_MESSAGE);
                                        messageField.setText("");//clear field n
ame
                }

                else //if key word and file name are valid, create Mono Cipher o
r Vigenere cipher object and read file
                        {

                        try
                                {
                        filename = messageField.getText() + ".txt";//construct fi
le name by reading user entry and appending .txt
                        reader = new FileReader(filename);//read file


                        if(e.getSource()==monoButton)//if the user chose mono ci
pher create object for that and encode/decode content
                                {
                                mcipher = new MonoCipher(keyword);
                                this.processFile(false);//incorporate processFil
e method to encode/decode file
                                System.exit(0);
```



```
                                }

                        else//if the user chose Vigenere cipher create object fo
r that and encode/decode content
                                {
                                vcipher = new VCipher(keyword);
                                this.processFile(true);//incorporate processFile
 method to encode/decode file
                                System.exit(0);
                                }

                                }//end of try

                        catch (FileNotFoundException e1) {
                                JOptionPane.showMessageDialog(null, "No file was fou
nd",
                                "File Not Found error", JOptionPane.ERROR_MESSAGE);
                                messageField.setText("");//clear field name
                                }//end of catch


                        }//end of else
        }//end of action performed method


        /**
         * Obtains cipher keyword
         * If the keyword is invalid, a message is produced
         * @return whether a valid keyword was entered
         */
        private boolean getKeyword()

        {
                int checks = 0;//incremented when each condition is met (non-emp
ty && upper case && no letters repeated)

                //first condition - non-empty
                keyword = keyField.getText();
                if(!keyword.trim().isEmpty())//
                        checks++;//if not empty increment checks

                //second condition - checks repetition
                int rep = 0;
                for(int i = 0; i< keyword.length(); i++)
                {
                        //checks one letter at a time for repetition
                        for(int j = 0; j<keyword.length(); j++)
                                {
                                if(keyword.charAt(i) == keyword.charAt(j))
                                        rep++;

                                }
                }
                if(rep == keyword.length())//each letter is compared with all le
tters including itself. So, if rep=keyword length then each letter exist only on
ce
                                checks++;

                //third condition - upper case
                int isUpper = 0;//incremented every time a letter is checked to
be in upper case
```

```
                for(int i = 0; i<keyword.length(); i++)
                {
                        switch(keyword.charAt(i))
                        {
                        case 'A':
                        case 'B':
                        case 'C':
                        case 'D':
                        case 'E':
                        case 'F':
                        case 'G':
                        case 'H':
                        case 'I':
                        case 'J':
                        case 'K':
                        case 'L':
                        case 'M':
                        case 'N':
                        case 'O':
                        case 'P':
                        case 'Q':
                        case 'R':
                        case 'S':
                        case 'T':
                        case 'U':
                        case 'V':
                        case 'W':
                        case 'X':
                        case 'Y':
                        case 'Z':


                                isUpper++;
                        }//end of switch
                }//end of for statement
                if(isUpper==keyword.length())//if all letters are upper case, in
crements checks
                        checks++;


                if(checks == 3)//if all conditions were met return true
                        {
                        checks=0;//to start fresh when user enters new keyword
                        return true;
                        }
                else
                        {
                        checks=0;//to start fresh when user enters new keyword
                        return false;
                        }
        }//end of method

        /**
         * Obtains filename from GUI
         * The details of the filename and the type of coding are extracted
         * If the filename is invalid, a message is produced
         * The details obtained from the filename must be remembered
         * @return whether a valid filename was entered
         */
        private boolean processFileName()
        {
                String fname = messageField.getText();
```



```
                int fname_length = fname.length();

                if(fname.isEmpty())
                        return false;//if message file is empty return false

                else if(fname.charAt(fname_length-1) == 'P' || fname.charAt(fnam
e_length-1) == 'C')
                        return true;//if it's not empty then check if it ends wi
th 'P' or 'C' and return true if so

                else
                        return false;//return false if it ends with a different
letter

        }

        /**
         * Reads the input text file character by character
         * Each character is encoded or decoded as appropriate
         * and written to the output text file
         * @param vigenere whether the encoding is Vigenere (true) or Mono (fals
e)
         * @return whether the I/O operations were successful
         */
        private boolean processFile(boolean vigenere)
        {
                LetterFrequencies let = new LetterFrequencies();//create letter
frequencies object to access its methods


                //to extract last letter of file name without the .txt
                String mField = messageField.getText();
                int len = mField.length();//length of keyword

                String cFile = mField.substring(0, len-1) + "C" + ".txt";//file n
ame for encoding

                String fileFreq = mField.substring(0, len-1) + "F" + ".txt";//rep
lace last letter with 'F' and add txt extension

                String dFile = mField.substring(0, len-1) + "D" + ".txt";//file n
ame for decoding



                if(mField.charAt(len-1)=='P')
                {//ENCODE

                        try {

                        writer = new PrintWriter(cFile);//open new file that end
s with 'C'
                        int c = 0;//this variable is to store integer value for
each letter read
                                char cchar;

                        for(;;)//to read all letters in file until the end
                                {
                                try {
                                        c = reader.read();//read letter by lette
r
                                        if(c == -1)//break upon EOF
```

```
                                        break;

                                    if(vigenere == false)//if mono cipher en
coder
                                    {
                                        cchar = mcipher.encode((char) c)
;//encode character
                                        let.addChar(cchar);//
                                    }

                                    else//if vigenere encoder
                                    {
                                        cchar = vcipher.encode((char) c)
;
                                        let.addChar(cchar);
                                    }

                                    writer.print(cchar);

                                } catch (IOException e) {
                                    // TODO Auto-generated catch block
                                    System.err.println("problem occured");;
                                    }//end of catch
                                }//end of for
                    writer.close();//when all letters are encoded, save file.


                    //to write into frequency file
                    writer = new PrintWriter(fileFreq);//open file that ends wit
h 'F'
                    writer.println(let.getReport());//pull frequency result from
 LetterFrequency class and print it to file

                    writer.close();//to save content to frequency file
                }//end of try
                 catch (FileNotFoundException e1) {
                        JOptionPane.showMessageDialog(null, "No file was found",
                        "File Not Found error", JOptionPane.ERROR_MESSAGE);
                        }
                    return true;


                }//END OF ENCODE

                else if(mField.charAt(len-1)=='C')
                { //DECODE
                        try {

                        writer = new PrintWriter(dFile);//open file that ends wi
th 'D'
                        int c = 0;//to save integer value of each character
                                char cchar;
                        for(;;)//to read all content until EOF
                        {
                                try {
                                        c = reader.read();//read letter by lette
r
                                        if(c == -1)
                                                break;

                                        if(vigenere == false)
                                        {
```



```
                                                cchar = mcipher.decode((char) c)
;//decode according to mono cipher
                                                let.addChar(cchar);
                                        }


                                        else
                                        {
                                                cchar = vcipher.decode((char) c)
;//decode according to vigenere
                                                let.addChar(cchar);
                                        }

                                        writer.print(cchar);//print to file

                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        System.err.println("problem occured");;
                                }//end of catch
                            }//end of endless for
                            writer.close();//to save content to D file

                            //to generate frequency file
                             writer = new PrintWriter(fileFreq);//open file that end
s with 'F'
                                writer.println(let.getReport());//pull frequenc
y result and save it to file

                                writer.close();//save content to frequency file
                        }//end of try
                        catch (FileNotFoundException e1) {
                                JOptionPane.showMessageDialog(null, "No file was found",
                                "File Not Found error", JOptionPane.ERROR_MESSAGE);
                                messageField.setText("");//clear field name
                                }//end of catch
                        return true;
                    }//END OF DECODE

                    else
                                JOptionPane.showMessageDialog(null, "invalid file name", "Inval
id File", JOptionPane.ERROR_MESSAGE);
                        return false;

                }//end of process file method
}//end of class
```

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Programming AE2
 * Processes report on letter frequencies
 */
public class LetterFrequencies
{
        /** Size of the alphabet */
        private final int SIZE = 26;

        /** Count for each letter */
        private int [] alphabetCounts;

        /** The alphabet */
        private char [] alphabet;


        /** Average frequency counts */
        private double [] avgCounts = {8.2, 1.5, 2.8, 4.3, 12.7, 2.2, 2.0, 6.1,
7.0,
                                                                0.2, 0.8, 4.0, 2.
4, 6.7, 7.5, 1.9, 0.1, 6.0,
                                                                    6.3, 9.1, 2.8
, 1.0, 2.4, 0.2, 2.0, 0.1};

        /** Character that occurs most frequently */
        private char maxCh;

        /** Total number of characters encrypted/decrypted */
        private int totChars;

        /**FileReader object to read C or D files*/
        private FileReader reader;

        /** the string that will hold the report */
        private String freq;

        /**
         * Instantiates a new letterFrequencies object.
         */
        public LetterFrequencies()
        {
                //initialize alphabet array with letters
                alphabet = new char [SIZE];
                for (int i = 0; i < SIZE; i++)
                        alphabet[i] = (char)('A' + i);

                //create alphabetCounts array and initialize to 0's
                alphabetCounts = new int[alphabet.length];
                for(int j = 0; j<alphabet.length; j++)
                        alphabetCounts[j] = 0;

                freq = "";//initialize string to empty

                totChars = 0;//initialize to 0
        }

        /**
```



```
         * Increases frequency details for given character
         * @param ch the character just read
         */
        public void addChar(char ch)
        {
                for(int i = 0; i<alphabet.length; i++)
                {
                        if(alphabet[i] == (char) ch)
                        alphabetCounts[i]++;
                }
        }

        /**
         * Gets the maximum frequency (double)
         * @return the maximum frequency
         */
        private double getMaxPC()
    {
                double max = (double)alphabetCounts[0] / totChars;//assume A has
 the most frequency

                for(int i = 0; i< alphabet.length; i++)//go through all alphabet
                {
                        if((double)alphabetCounts[i] / totChars >= max)//compare
 to max
                        {
                                max = (double)alphabetCounts[i] / totChars;
                        }
                }
            return max;  // return max frequency amount(double)
        }
        /**
         * Gets the maximum frequency (character)
         * @return the maximum frequency character
         */
        private char getMaxCh()
    {
        double max = (double)alphabetCounts[0] / totChars;//assume A has the mos
t frequency

        for(int i = 0; i<alphabet.length; i++)//go through all alphabet
        {
                if((double)alphabetCounts[i] / totChars >= max)//compare to max
                {
                        max = (double)alphabetCounts[i] / totChars;
                        maxCh = alphabet[i];
                }
        }
    return maxCh;  // return max frequency letter(char)
}


        /**
         * Returns a String consisting of the full frequency report
         * @return the report
         */
        public String getReport()
        {


                                //to calculate totChars
```

```
                                for(int v = 0; v<alphabet.length; v++)

                                        if(alphabetCounts[v] != 0)
                                                totChars++;

                                //TO GENERATE FREQ STRING
                                for(int x = 0; x<alphabet.length; x++)

                                        freq += alphabet[x] + "\t\t" + alphabetCou
nts[x] + "\t\t"+ String.format("%.2f", (double)alphabetCounts[x] / totChars)

                                        + "\t\t" + avgCounts[x]

                                        + "\t\t" + String.format(" %05.2f", (double
)alphabetCounts[x] / totChars − avgCounts[x])


                                        + "\n";



                    //return string for file's data
                return "Letter" + " \t" + "Freq" +  "\t" + "Freq%" + "\t\t"

                        + "AvgFreq%" + " \t" + "Diff"+ "\n" + freq

                        +"\n" + "The most frequent letter is " + getMaxCh() + " at " + String.for
mat("%.2f", getMaxPC());



        }//end of getReport method
}//end of class
```



```
/**
 * Programming AE2
 * Contains monoalphabetic cipher and methods to encode and decode a character.
 */
public class MonoCipher
{
        /** The size of the alphabet. */
        private final int SIZE = 26;

        /** The alphabet. */
        private char [] alphabet;

        /** The cipher array. */
        private char [] cipher;

        int rep; // stays zero if letter is not mentioned in keyword
        /**
         * Instantiates a new mono cipher.
         * @param keyword the cipher keyword
         */
        public MonoCipher(String keyword)
        {

                //fill alphabet array
                alphabet = new char [SIZE];
                for (int i = 0; i < SIZE; i++)
                        alphabet[i] = (char)('A' + i);

                //create cipher array
                cipher = new char[SIZE];

                int len = keyword.length(); //keyword length

                //to copy keyword into the first section of cipher array
                for(int j = 0; j<len; j++)
                        {cipher[j] = keyword.charAt(j);
                        }

                //to copy rest of alphabet in the remaining section of cipher ar
ray
                //loops through all alphabet
                        for(int i = SIZE−1; i>=0; i−−)
                        {
                                //loops each letter through first section of cip
her to check if it's there
                                for(int x = 0; x<len; x++)

                                        if(alphabet[i] == cipher[x])

                                                rep++;//if it's there increment
this variable


                                        if(rep==0)//if not repeated add to secon
d section of cipher
                                        {
                                        cipher[len] = alphabet[i];
                                        len++;
                                        }

                                        else
                                                rep = 0;
                        }
```

```
                        //to print alphabet and cipher arrays as requested in AE
2 description
                        for(int i = 0; i<SIZE; i++)
                                System.err.print(alphabet[i]);
                        System.err.println();

                        for(int i = 0; i<SIZE; i++)
                                System.err.print(cipher[i]);
                        System.err.println();


                // create first part of cipher from keyword
                // create remainder of cipher from the remaining characters of t
he alphabet
                // print cipher array for testing and tutors
        }

        /**
         * Encode a character
         * @param ch the character to be encoded
         * @return the encoded character
         */
        public char encode(char ch)
        {
                int loc;//location of letter on alphabet array
                int isValid=0;//incremented when ch is an upper case letter, sta
ys 0 if it is space or punctuation

                for(int i = 0; i<SIZE; i++)
                        {
            if( alphabet[i] == ch)
                isValid++;
                        }
                if(isValid==0)//when space or punctuation return as is
            return ch;


                else//else, return encoded letter
                        {

                          loc =  ch - 'A';
                          return cipher[loc];
                        }//end of else
        }//end of encode

        /**
         * Decode a character
         * @param ch the character to be encoded
         * @return the decoded character
         */
        public char decode(char ch)
        {
                int isValid=0;//incremented when ch is an upper case letter, sta
ys 0 if it is space or punctuation
                int foundAt = 0;

                for(int i = 0; i<SIZE; i++)
                {
            if( alphabet[i] == ch)
                isValid++;
```



```
                }
                if(isValid==0)//when space or punctuation return as is
            return ch;


                else//else, return decoded letter letter
                        {
                        for(int i = 0; i<SIZE; i++)
                                {
                                if(cipher[i] ==ch)//find where letter is located
 in cipher and save it's place
                                        {
                                                foundAt = i;
                                                break;
                                        }
                                }
                        return alphabet[foundAt];//return equivalent letter in a
lphabet

                        }//end of else
                }//end of decode
}//end of class
```

```
/**
 * Programming AE2
 * Class contains Vigenere cipher and methods to encode and decode a character
 */
public class VCipher
{
        private char [] alphabet;   //the letters of the alphabet
        private final int SIZE = 26;
        private int countEn = 0;//to keep track of the order of character to be
encoded
        private int countDe = 0;//to keep track of the order of character to be
decoded
        private int len;//keyword length
        private char[][] vig;//Vigenere character array
        /**
         * The constructor generates the cipher
         * @param keyword the cipher keyword
         */
        public VCipher(String keyword)
        {

                //create alphabet
                alphabet = new char [SIZE];
                for (int i = 0; i < SIZE; i++)
                        alphabet[i] = (char)('A' + i);

                //keyword length
                len = keyword.length();

            //create Vigenere array with  number of rows equal to keyword length
                //and number of columns equal to the number of alphabet letters
            vig = new char[len][SIZE];

            //initialize it with 0's
            for(int i=0;i<len;i++)
                for(int j = 0; j<SIZE; j++)
                        vig[i][j] = 0;

            //loop through all rows and adds letters until we reach 'Z'
            for(int i = 0; i<len; i++)
                {
                        vig[i][0] = keyword.charAt(i);//fill first spot of each
row with a letter from keyword
                        int loc =  keyword.charAt(i) - 'A';//calculate position
of each keyword letter on alphabet array

                        //after filling first spot of each row, loop to all posi
tion of each row
                        for(int j = 1; j<SIZE; j++)
                                {
                                        if(j + loc < SIZE)//if we're still below
 'Z'
                                                vig[i][j] = alphabet[j+loc];//ke
ep adding letters after the letter from keyword
                                }//end of j loop
                }//end of i loop

            //now to fill remaining spots on each row after 'Z'
            for(int i = 0; i<len; i++)//loop through all rows
                {
                        for(int j = 1; j<SIZE; j++)//loop every row after the fi
rst letter(the one from the keyword)
```



```
                                {
                                        if(vig[i][0] == 'Z')//special case - if
the letter from the keyword is Z

                                                {
                                                        int a = 0;//used to incr
ement alphabet
                                                        for(int x = 1; x < SIZE-
1; x++)//go to all spots following 'Z' which is at the column 0
                                                        {
                                                                vig[i][x
] = alphabet[0+a];//add letters starting with 'A' until the end
                                                                a++;
                                                        }
                                                }//end of if

                                        else if(vig[i][j] == 'Z')//this is for a
ll other scenarios where 'Z' could come at a column other than the first one
                                                {
                                                        int a = 0;//used to incr
ement alphabet
                                                        for(int x = j; x<SIZE-1;
 x++)
                                                        {
                                                                vig[i][x
+1] = alphabet[0+a];//add letters starting with 'A'
                                                                a++;
                                                        }
                                                }//end of if
                                }//end of j loop
                }//end of i loop


            //this is to print alphabet and Vigenere arrays
            //ALPHABET
            for(int i = 0; i<SIZE; i++)
                        System.err.print(alphabet[i]);
            System.err.println();

            //VIGENERE
            for(int i=0;i<len;i++)
                {
                        for(int j = 0; j<SIZE; j++)
                                        System.err.print(vig[i][j]);
                        System.err.println();
                }
            }//end of method

        /**
         * Encode a character
         * @param ch the character to be encoded
         * @return the encoded character
         */
        public char encode(char ch)
        {

                int isValid=0;//incremented when ch is an upper case letter, sta
ys 0 if it is space or punctuation

                for(int i = 0; i<SIZE; i++)
                        {
                            if( alphabet[i] == ch)
```

```
                        isValid++;
                        }
                if(isValid==0)//when space or punctuation return as is
            return ch;

                else//when actual upper case letter then encode
                        {
                                int loc = ch - 'A';//calculate location on alpha
bet array - column

                                //start with row 0,1,2..until end of array and t
hen calculate row by
                                //order_of_character in the document % length of
 keyword
                                ch = vig[countEn%len][loc];
                                countEn++;//increment order for next character

                            return ch;  // return encoded character
                        }
        }//end of encode method

        /**
         * Decode a character
         * @param ch the character to be decoded
         * @return the decoded character
         */
        public char decode(char ch)
        {

                int isValid=0;//incremented when ch is an upper case letter, sta
ys 0 if it is space or punctuation

                for(int i = 0; i<SIZE; i++)
                        {
                        if( alphabet[i] == ch)
                                isValid++;
                        }

                if(isValid==0)//when space or punctuation return as is
                    return ch;

                else//else return decoded character
                        {
                                //start with row 0 - find letter to be decoded o
n that row
                                //and return matching letter on alphabet array
                                int i;
                                for(i = 0; i < SIZE; i++)
                                        if(vig[countDe%len][i] == ch)//calculate
 which row we need by - order of character % length of keyword
                                                break;//stop when we find it
                                countDe++;//increment counter for next letter

                    return alphabet[i];//return matching character on alphabet a
rray
                        }
        }//end of decode
}//end of class
```