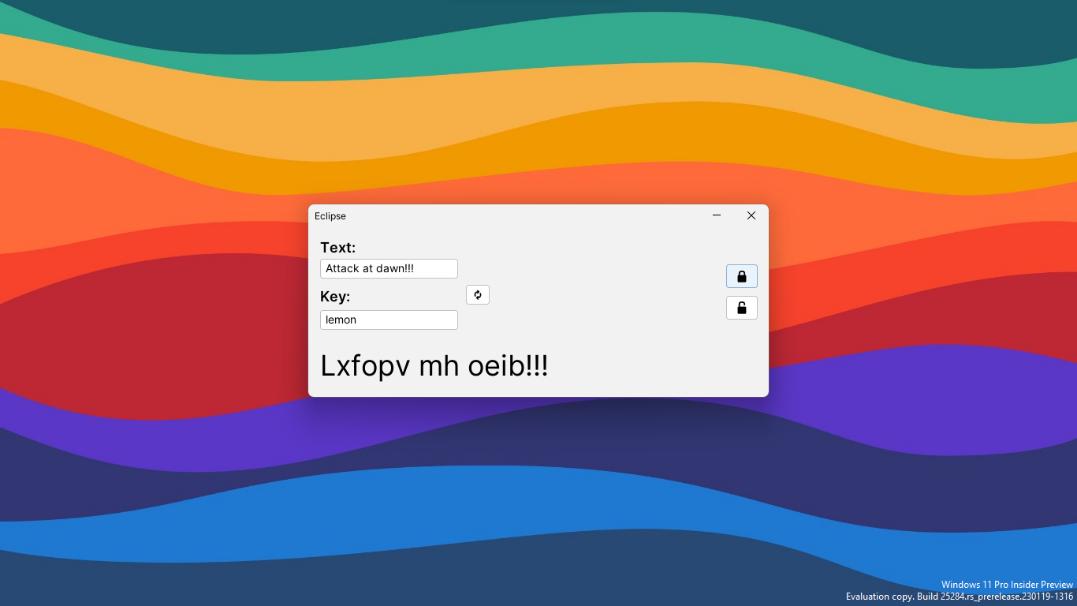
Fumnanya Kavan Mowete CSC313 Project   
20CG028103 Encryption System using the Vigenère cipher (#5)  
300 Level, Computer Science <https://github.com/poopsicles/Eclipse>

**Introduction:**

The project was to implement a small GUI application, using the Java Swing toolkit, that allows the user to encrypt and decrypt text using the [Vigenère cipher](https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher).

I did this, and added a few quality-of-life improvements, like a scrambling animation and text swap button.

**Screenshots:**

* The application when opened:  
    
  
* The application encrypting a string:  
    
  
* The application playing the scrambling animation:  
    
  
* The application decrypting a string:  
    
  

**Button logic:**

There’re three buttons in the application: one to encrypt the text, one to decrypt, and another one to copy the output into the input text box, named encryptButton, decryptButton, and swapButton, respectively.

The logic for swapButton is the easiest:

swapButton.addActionListener(e -> {  
 String output = outputText.getText();  
  
 inputText.setText(output);  
 outputText.setText("");  
});

This simply sets the inputTextbox text to the current output and then clears the output, effectively “swapping” the contents and allowing the encryption/decryption to be reversed.

encryptButton and decryptButton are slightly more involved:

encryptButton.addActionListener(e -> {  
 String input = inputText.getText();  
 String key = keyText.getText();  
  
 if (Objects.equals(input, "") || Objects.equals(key, "")) {  
 return;  
 }  
  
 output = encrypt(input, key);  
   
 scrambleTimer.start();  
 endScrambleTimer.start();  
});  
  
decryptButton.addActionListener(e -> {  
 String input = inputText.getText();  
 String key = keyText.getText();  
  
 if (Objects.equals(input, "") || Objects.equals(key, "")) {  
 return;  
 }  
  
 output = decrypt(input, key);  
   
 scrambleTimer.start();  
 endScrambleTimer.start();  
});

First, we get the text from the inputText and keyText textboxes (while making sure they’re not empty). The strings are then passed to the respective encrypt/decrypt function and then the result is kept in output.

scrambleTimer() then begins, which triggers the playing of the scramble animation – which changes outputText to random characters every 50 milliseconds. At the same time, another timer starts, endscrambleTimer(), which waits for a second and then sets outputText to output, ending the animation.

**Code:**

Main.java

// external imports for theme and font

import com.formdev.flatlaf.FlatLaf;  
import com.formdev.flatlaf.FlatLightLaf;  
import com.formdev.flatlaf.fonts.inter.FlatInterFont;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
import java.util.Objects;  
import java.util.Random;  
  
public class Main {  
 // static imports for components that can be modified globally  
 static JTextField inputText;  
 static JTextField outputText;  
 static Timer scrambleTimer;   
 static Timer endScrambleTimer;   
 static String output;   
   
 public static void main(String[] args) {   
 // set font to Inter   
 FlatInterFont.install();   
 FlatLaf.setPreferredFontFamily( FlatInterFont.FAMILY );   
 FlatLaf.setPreferredLightFontFamily( FlatInterFont.FAMILY\_LIGHT );   
 FlatLaf.setPreferredSemiboldFontFamily( FlatInterFont.FAMILY\_SEMIBOLD );   
   
 // fix anti-aliasing issue   
 System.setProperty("awt.useSystemAAFontSettings","on");   
   
 // round component corners   
 UIManager.put("TextComponent.arc", 7);   
 UIManager.put("Button.arc", 7);   
   
 // setup L&F   
 FlatLightLaf.setup();   
   
 JFrame frame = new JFrame("Eclipse");   
 frame.setSize(600, 253);   
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
 frame.setResizable(false);   
   
 JPanel panel = new JPanel();   
 frame.add(panel);   
 placeComponents(panel);   
   
 frame.setVisible(true);   
 frame.setLocationRelativeTo(null);   
 }   
   
 // create components in a panel for easy adding   
 // manually layout code because automated layouts are bad   
 private static void placeComponents(JPanel panel) {   
 panel.setLayout(null);   
   
 JLabel inputLabel = new JLabel("Text:");   
 inputLabel.setFont(UIManager.getFont("h2.font"));   
 inputLabel.setBounds(15, 10, 100, 30);   
 panel.add(inputLabel);   
   
 inputText = new JTextField(0);   
 inputText.setFont(UIManager.getFont("large.font"));   
 inputText.setBounds(15, 40, 175, 25);   
 panel.add(inputText);   
   
 JLabel keyLabel = new JLabel("Key:");   
 keyLabel.setFont(UIManager.getFont("h2.font"));   
 keyLabel.setBounds(15, 75, 80, 25);   
 panel.add(keyLabel);   
   
 JTextField keyText = new JTextField(0);   
 keyText.setFont(UIManager.getFont("large.font"));   
 keyText.setBounds(15, 105, 175, 25);   
 panel.add(keyText);   
   
 outputText = new JTextField(0);   
 outputText.setFont(UIManager.getFont("h00.font"));   
 outputText.setEditable(false);   
 outputText.setBackground(null);   
 outputText.setBorder(null);   
 outputText.setBounds(15, 150, 460, 50);   
 outputText.setText("");   
 panel.add(outputText);   
   
 JButton encryptButton = new JButton("🔒");   
 encryptButton.setFont(UIManager.getFont("h2.font"));   
 encryptButton.setBounds(530, 47, 40, 30);   
 panel.add(encryptButton);   
   
 JButton decryptButton = new JButton("🔓");   
 decryptButton.setFont(UIManager.getFont("h2.font"));   
 decryptButton.setBounds(530, 87, 40, 30);   
 panel.add(decryptButton);   
   
 JButton swapButton = new JButton("🔃");   
 swapButton.setBounds(200, 73, 30, 25);   
 panel.add(swapButton);   
   
 scrambleTimer = new Timer(50, new ScrambleAction());   
 endScrambleTimer = new Timer(1000, new EndScrambleAction());   
   
 // onclick event handlers   
 encryptButton.addActionListener(e -> {   
 String input = inputText.getText();   
 String key = keyText.getText();   
   
 if (Objects.equals(input, "") || Objects.equals(key, "")) {   
 return;   
 }   
   
 output = encrypt(input, key);   
   
 scrambleTimer.start();   
 endScrambleTimer.start();   
 });   
   
 decryptButton.addActionListener(e -> {   
 String input = inputText.getText();   
 String key = keyText.getText();   
   
 if (Objects.equals(input, "") || Objects.equals(key, "")) {   
 return;   
 }   
   
 output = decrypt(input, key);   
   
 scrambleTimer.start();   
 endScrambleTimer.start();   
 });   
   
 swapButton.addActionListener(e -> {   
 String output = outputText.getText();   
   
 inputText.setText(output);   
 outputText.setText("");   
 });   
 }   
   
 // take two characters and return the cipher of a by b   
 private static char wrappingAdd(char a, char b) {   
 return (char) (97 + ((a + b - 97) % 123) % 97);   
 }   
   
 // decipher the original character   
 private static char wrappingSub(char c, char b) {   
 return (char) (97 + ((Math.floorMod((Math.floorMod(c - b - 97, 123) - 97), 97)) % 26));   
 }   
   
 // to store original capitalisation   
 private static ArrayList<Boolean> getCapsArray(String s) {   
 ArrayList<Boolean> capsArray = new ArrayList<>();   
   
 for (int i = 0; i < s.length(); i++) {   
 if (Character.isUpperCase(s.charAt(i))) {   
 capsArray.add(true);   
 } else {   
 capsArray.add(false);   
 }   
 }   
   
 return capsArray;   
 }   
   
 // grab inputs and return encrypted version   
 // skip non-letters   
 public static String encrypt(String input, String key) {   
 int keyLen = key.length();   
 int inpLen = input.length();   
 StringBuilder res = new StringBuilder(inpLen);   
 ArrayList<Boolean> capsArray = getCapsArray(input);   
   
 input = input.toLowerCase();   
 key = key.toLowerCase();   
   
 for (int i = 0; i < input.length(); i++) {   
 char c = input.charAt(i);   
   
 if (Character.isLetter(c)) {   
 c = wrappingAdd(input.charAt(i), key.charAt(i % keyLen));   
   
 res.append(   
 capsArray.get(i) ? Character.toUpperCase(c) : c   
 );   
 } else {   
 res.append(c);   
 }   
 }   
   
 return res.toString();   
 }   
   
 // grab inputs and return decrypted version   
 // skip non-letters   
 public static String decrypt(String input, String key) {   
 int keyLen = key.length();   
 int inpLen = input.length();   
 StringBuilder res = new StringBuilder(inpLen);   
 ArrayList<Boolean> capsArray = getCapsArray(input);   
   
 input = input.toLowerCase();   
 key = key.toLowerCase();   
   
 for (int i = 0; i < input.length(); i++) {   
 char c = input.charAt(i);   
   
 if (Character.isLetter(c)) {   
 c = wrappingSub(input.charAt(i), key.charAt(i % keyLen));   
   
 res.append(   
 capsArray.get(i) ? Character.toUpperCase(c) : c   
 );   
 } else {   
 res.append(c);   
 }   
 }   
   
 return res.toString();   
 }   
   
 public static String scrambleText(String originalText) {   
 String ALL\_POSSIBLE\_CHARACTERS = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!\"#$%&'()\*+,-./:;<=>?@[\\]^\_`{|}~";   
 Random RANDOM = new Random();   
   
 char[] originalTextArray = originalText.toCharArray();   
 java.util.List<Character> characters = new ArrayList<>();   
 for (char c : originalTextArray) {   
 characters.add(c);   
 }   
 for (int i = 0; i < characters.size(); i++) {   
 int randomIndex = RANDOM.nextInt(characters.size());   
 char randomChar = ALL\_POSSIBLE\_CHARACTERS.charAt(RANDOM.nextInt(ALL\_POSSIBLE\_CHARACTERS.length()));   
 characters.set(randomIndex, randomChar);   
 }   
 StringBuilder scrambledText = new StringBuilder();   
 for (char c : characters) {   
 scrambledText.append(c);   
 }   
 return scrambledText.toString();   
 }   
   
 private static class ScrambleAction implements ActionListener {   
 @Override   
 public void actionPerformed(ActionEvent e) {   
 String scrambledText = scrambleText(inputText.getText());   
 outputText.setText(scrambledText);   
 }   
 }   
   
 private static class EndScrambleAction implements ActionListener {   
 @Override   
 public void actionPerformed(ActionEvent e) {   
 scrambleTimer.stop();   
 endScrambleTimer.stop();   
   
 outputText.setText(output);   
 }   
 }   
}