Experiment No. 11

Aim: Program to Implement Solution using Multiple paradigm

Description:

- A JavaFX Ceasar Cipher Encrypter and Decrypter is a program that allows users to encrypt and decrypt messages using the Ceasar Cipher method. The program has a graphical user interface (GUI) built using JavaFX, which makes it easy for users to interact with the program and perform the required actions.
- To use the program, the user first enters the message they want to encrypt or decrypt in the designated text field. They can then choose whether they want to encrypt or decrypt the message by clicking the appropriate button.
- When encrypting a message, the user can also specify the shift value, which determines how many places each letter in the message should be shifted. For example, if the shift value is 3, then the letter "A" will be replaced with the letter "D", "B" will be replaced with "E", and so on.
- When decrypting a message, the user can specify the shift value that was used to encrypt the message. The program will then use this shift value to reverse the encryption and display the original message.
- In addition to encrypting and decrypting messages, the program also has a feature that allows the user to crack a message that has been encrypted with an unknown shift value. To do this, the user can click the "Crack" button and the program will attempt to decrypt the message using all possible shift values and display the results.
- Overall, the JavaFX Ceasar Cipher Encrypter and Decrypter is a simple but effective tool for encrypting and decrypting messages using the Ceasar Cipher method. It is user-friendly and easy to use, making it suitable for both beginners and experienced Java programmers.

Source Code:

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
```

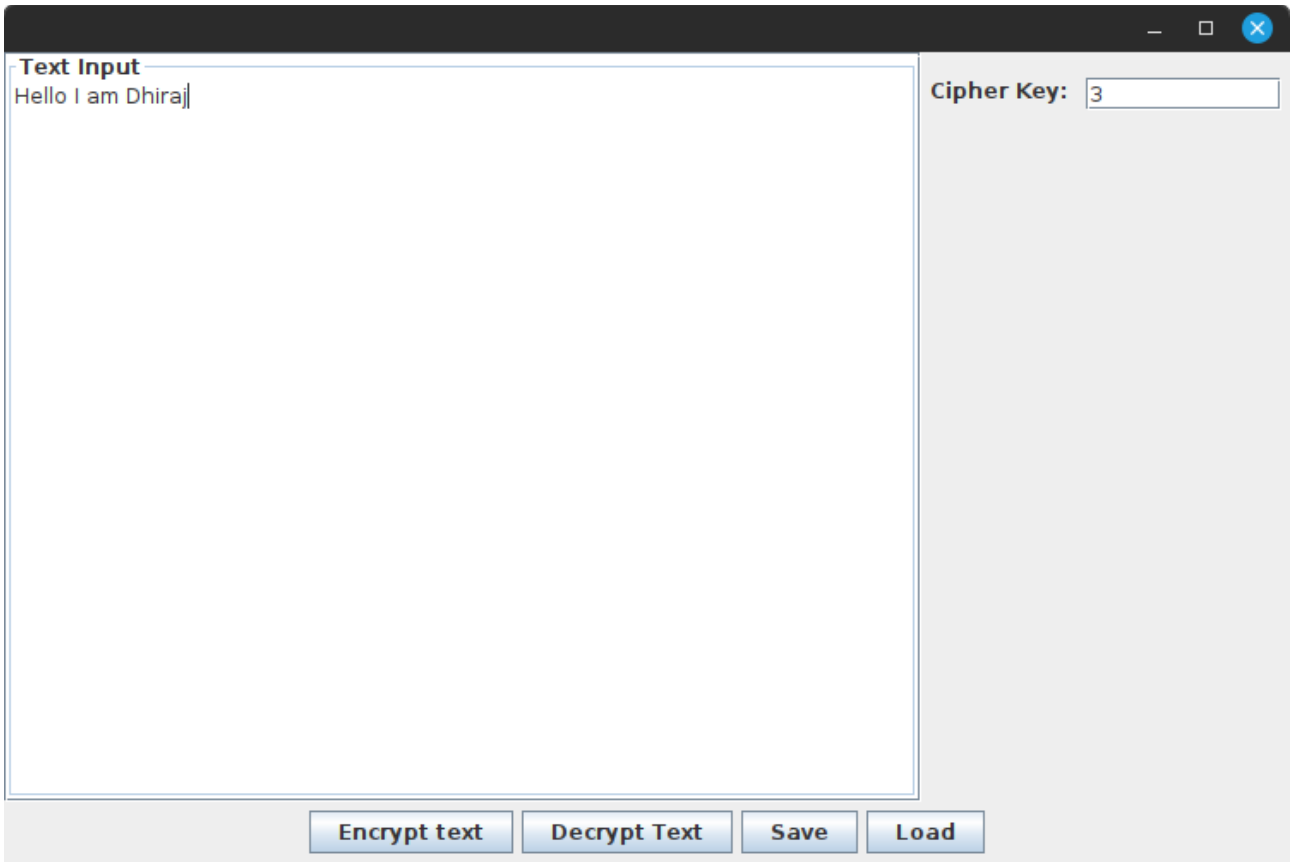
```
import java.nio.file.Files;
import java.nio.file.Paths;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollBar;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class gui extends JFrame implements ActionListener {
	public static final String alpha = "abcdefghijklmnopqrstuvwxyz";

	int[][] puzzle = new int[9][9];
	FlowLayout experimentLayout = new FlowLayout();
	private JPanel controls = new JPanel();
	private JPanel write = new JPanel();
	JTextArea canvas = new JTextArea();
	JTextField CipherKeyIn = new JTextField(10);
	JFileChooser fc = new JFileChooser();
	private JButton decrypt = new JButton("Decrypt Text");
	private JButton encrypt = new JButton("Encrypt text");
	private JButton save = new JButton("Save");
	private JButton load = new JButton("Load");
	private JButton rules = new JButton("Rules");
	JScrollPane sp = new JScrollPane(write);
	JScrollBar bar = new JScrollBar();
	String Location = null;
	boolean justRead = false;
	boolean justSolved = false;
	boolean justGen = false;
	String Error = "NA";

   public gui() {
	   getContentPane().setLayout(new BorderLayout());
	   encrypt.addActionListener(this);
	   decrypt.addActionListener(this);
	   save.addActionListener(this);
	   load.addActionListener(this);
	   rules.addActionListener(this);
	   controls.add(encrypt);
	   controls.add(decrypt);
	   controls.add(save);
	   controls.add(load);

	   JScrollPane scrollPane = new JScrollPane(canvas);
	   setPreferredSize(new Dimension(6000, 6000));
	   JPanel pane = new JPanel();
```

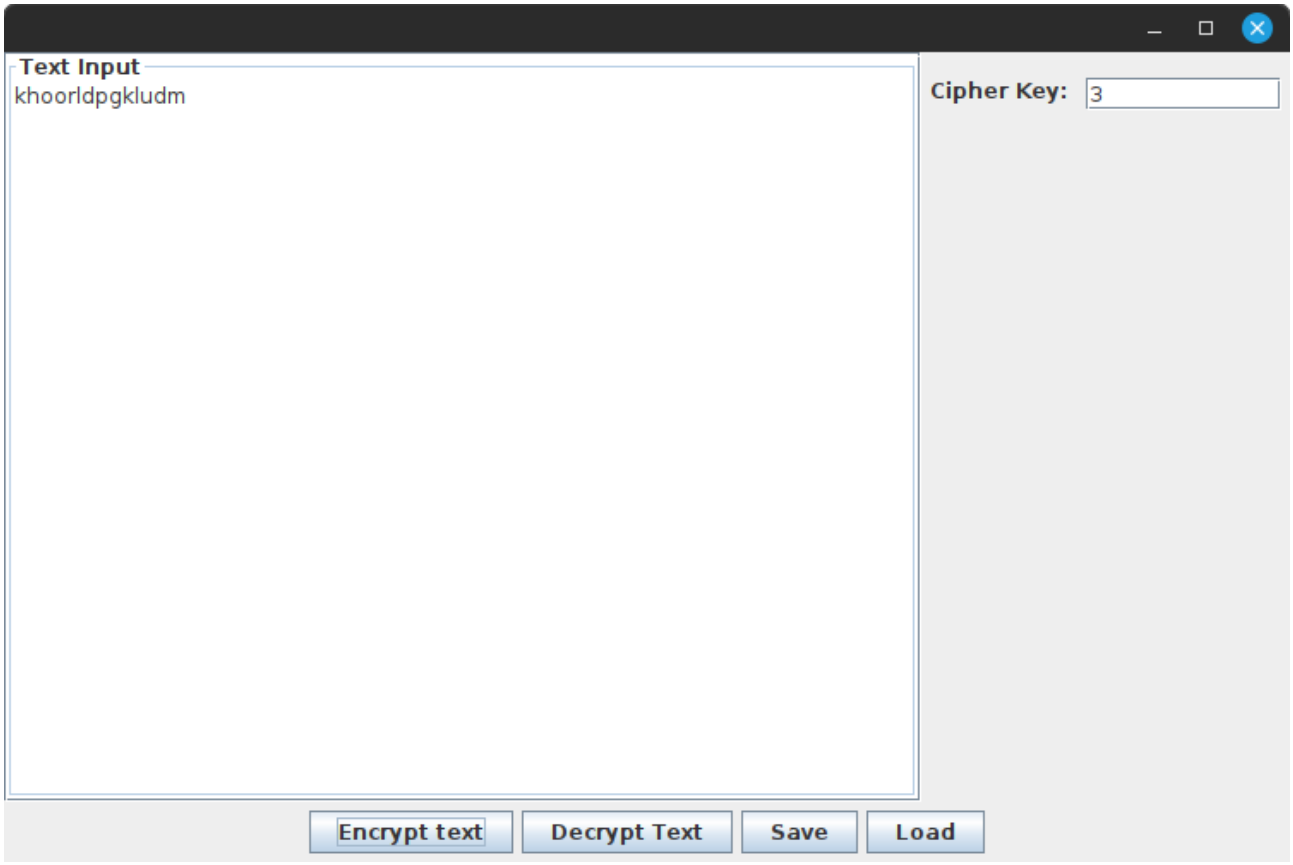
```
        JPanel encryptionControls = new JPanel();
        encryptionControls.setLayout(new FlowLayout());
        pane.setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        if (true) {
        //natural height, maximum width
        c.fill = GridBagConstraints.HORIZONTAL;
        }

        JLabel cipherKey = new JLabel("Cipher Key:");
       //c.fill = GridBagConstraints.HORIZONTAL;
        c.anchor = GridBagConstraints.NORTH;
        c.weightx = 0.5;
        c.gridx = 0;
        c.gridy = 0;
        c.insets = new Insets(10,0,0,0);
        pane.add(cipherKey, c);
        CipherKeyIn = new JTextField(10);
        // c.fill = GridBagConstraints.HORIZONTAL;
        c.anchor = GridBagConstraints.BASELINE;
        c.weightx = 0.5;
        c.gridx = 1;
        c.gridy = 0;
        c.insets = new Insets(10,10,0,0);
        pane.add(CipherKeyIn, c);
        encryptionControls.add(pane);

   add(scrollPane, BorderLayout.CENTER);
   canvas.setBorder(BorderFactory.createTitledBorder("Text Input"));
   canvas.setPreferredSize(new Dimension (250, 250));
   //this will enable the word wrapping feature
   canvas.setLineWrap(true);
   canvas.setWrapStyleWord(true);
  //write.add(canvas);
  //getContentPane().add("North", write);
   getContentPane().add("South", controls);
   getContentPane().add("East", encryptionControls);

   getContentPane().setSize(400, 700);
} // init()


      public void actionPerformed(ActionEvent e) {
              String encryptedText;
              String plainText;
              String currentKey = CipherKeyIn.getText();
              int shiftKey = Integer.parseInt(currentKey);
              if(e.getSource() == encrypt){
      plainText = "";
                      plainText = canvas.getText();
                      //plainText = plainText.replaceAll(" ", "");
                      encryptedText = "";
```

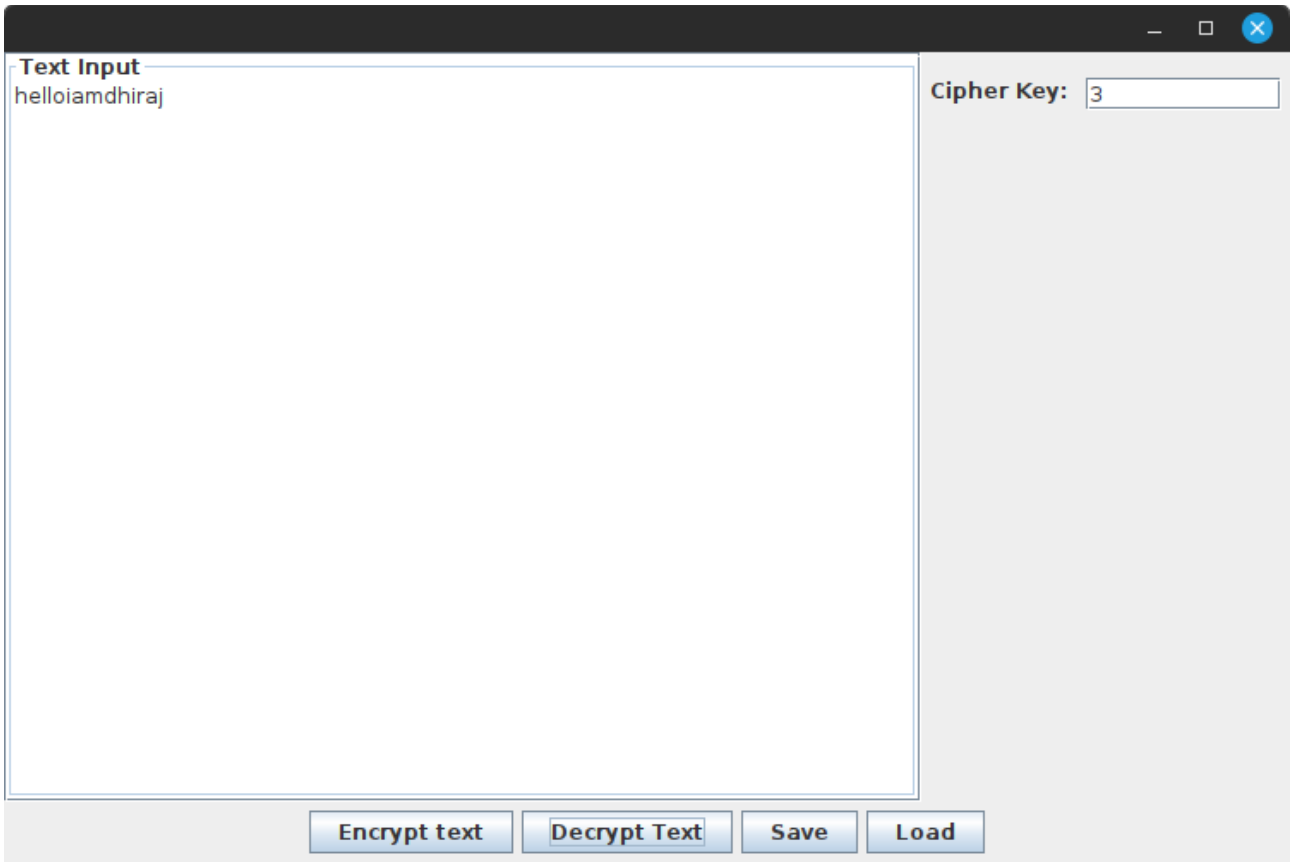
```
                cipherEnc encryptor = new cipherEnc();
                encryptedText = encryptor.enc(plainText, shiftKey);
                canvas.setText(encryptedText);

        }if(e.getSource() == decrypt){

                encryptedText = canvas.getText();
                cipherDec decryptor = new cipherDec();
                plainText = decryptor.dec(encryptedText, shiftKey);
                canvas.setText(plainText);
        }else if(e.getSource() == load){

                int returnVal = fc.showOpenDialog(gui.this);

                if (returnVal == JFileChooser.APPROVE_OPTION) {
   File file = fc.getSelectedFile();
   String local = file.getAbsolutePath();
   String loaded = "";
   try {
                                loaded =readFile(local, Charset.defaultCharset());
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
   canvas.setText(loaded);
}

        }
        else if(e.getSource() == save){
                JFileChooser chooser = new JFileChooser();
           chooser.setCurrentDirectory(new File(""));
           int retrival = chooser.showSaveDialog(null);
           if (retrival == JFileChooser.APPROVE_OPTION) {
              try(FileWriter fw = new FileWriter(chooser.getSelectedFile())) {
                 fw.write(canvas.getText());
              } catch (Exception ex) {
                 ex.printStackTrace();
              }
           }
        }

}

public class cipherEnc {
        public String enc(String plainText, int shiftKey) {
                plainText = plainText.toLowerCase();
                plainText = plainText.replaceAll(" ", "");
                String cipherText = "";
                for (int i = 0; i < plainText.length(); i++) {
                int charPosition = alpha.indexOf(plainText.charAt(i));
                int keyVal = (shiftKey + charPosition) % 26;
                char replaceVal = alpha.charAt(keyVal);
                cipherText += replaceVal;
```

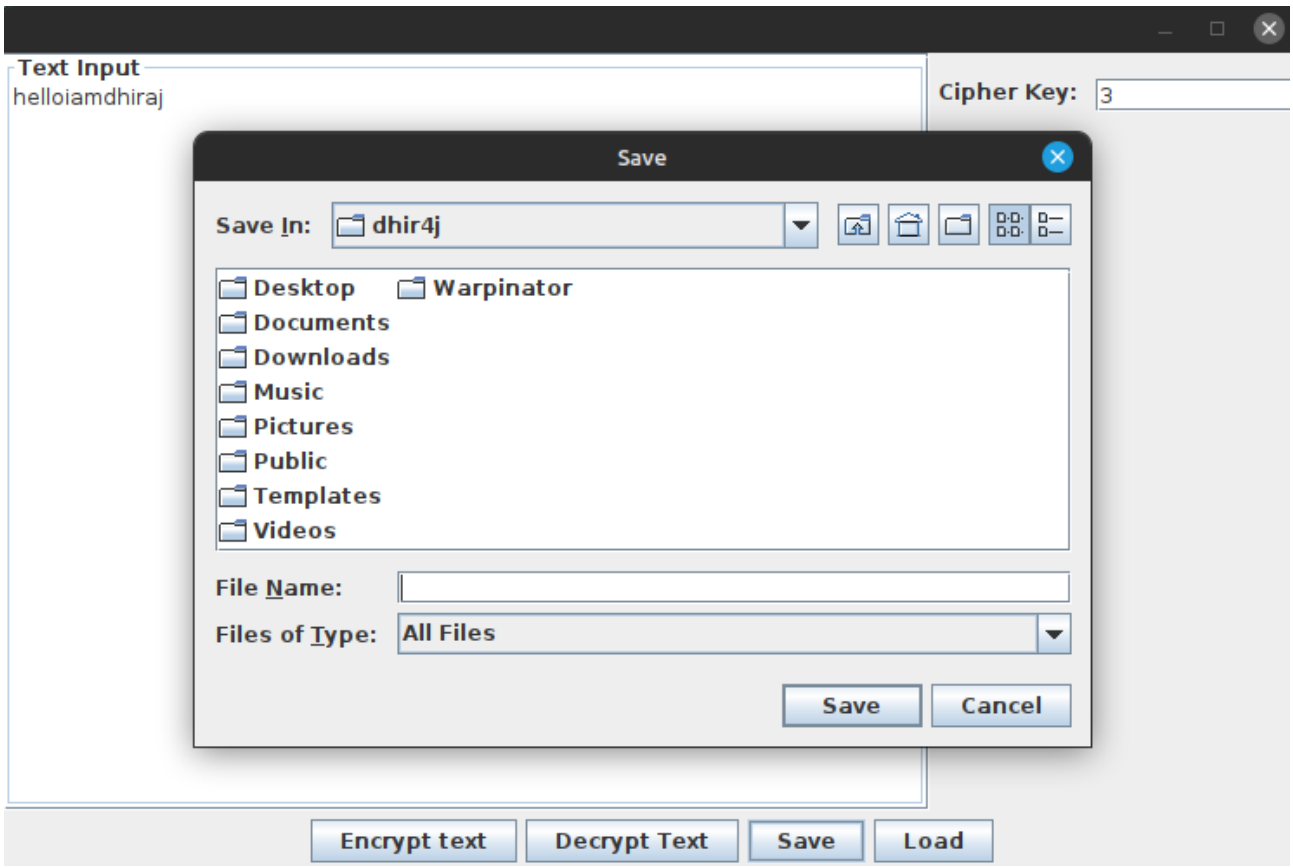
```
                }
                return cipherText;
            }
        }

        public class cipherDec{
            public String dec(String cipherText, int shiftKey) {
                cipherText = cipherText.toLowerCase();
                String message = "";
                for (int ii = 0; ii < cipherText.length(); ii++) {
                  int charPosition = alpha.indexOf(cipherText.charAt(ii));
                  int keyVal = (charPosition - shiftKey) % 26;
                  if (keyVal < 0) {
                        keyVal = alpha.length() + keyVal;
                  }
                  char replaceVal = alpha.charAt(keyVal);
                  message += replaceVal;
                }
                return message;
             }
        }


        static String readFile(String path, Charset encoding)throws IOException{
            byte[] encoded = Files.readAllBytes(Paths.get(path));
            return encoding.decode(ByteBuffer.wrap(encoded)).toString();
        }
        public static void main(String[] args){
            gui maingui = new gui();

      maingui.setSize(750,500);
      maingui.setVisible(true);
      maingui.addWindowListener(new WindowAdapter() {// Quit the application
        public void windowClosing(WindowEvent e) {
          System.exit(0);
        }
      });
    }
}
//@dhir4j <3!
```

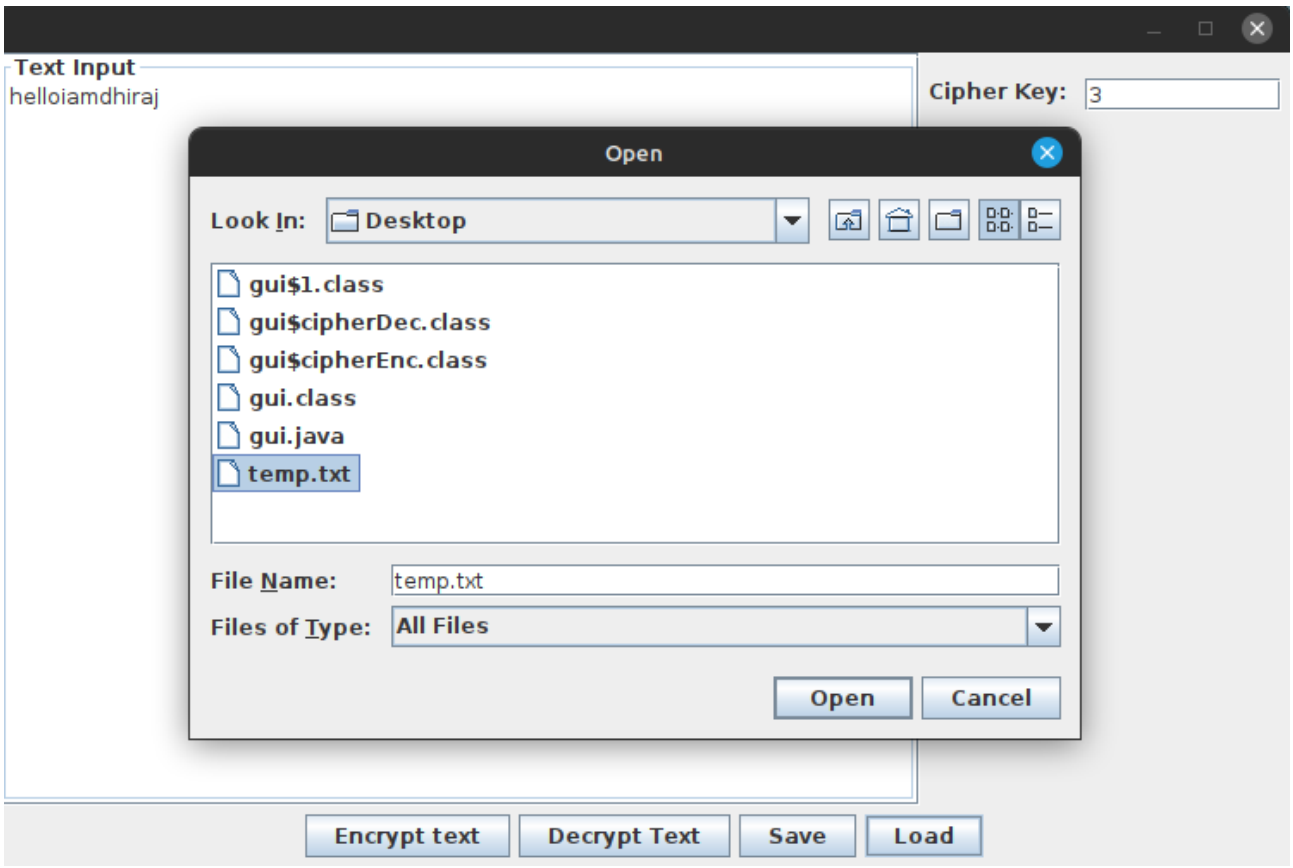
Screenshots:






Text Input
helloiamdhiraj
Cipher Key:
3
Encrypt text
Decrypt Text
Save
Load


Text Input
helloiamdhiraj
Cipher Key:
3
Save
Save In:
dhir4j
Desktop
Warpinator
Documents
Downloads
Music
Pictures
Public
Templates
Videos
File Name:
Files of Type:
All Files
Save
Cancel
Encrypt text
Decrypt Text
Save
Load



Conclusion: Thus, we have created a program to implement solution using multiple paradigm.