import java.util.StringTokenizer;

import javax.swing.JOptionPane;

**public class TestEncryption**{

**public static void main(String[] args)** {

String code, output = "";

String text = JOptionPane.showInputDialog("Enter message");

output += "The original message is \n" + text + "\n";

Cipher c = new Caesar(text);

c.encrypt();

code = c.getEncodedMessage();

output += "\nCeasar Cipher\nThe encrypted message is \n" + code + "\n";

c.decrypt(code);

code = c.getDecodedMessage();

output += "The decrypted message is \n" + code + "\n";

c = new Transpose(text);

c.encrypt();

code = c.getEncodedMessage();

output += "\nTranspose\nThe encrypted Transpose message is \n" + code + "\n";

c.decrypt(code);

code = c.getDecodedMessage();

output += "The decripted Transpose message is \n" + code + "\n";

Reverser r = new Reverser(text);

r.encrypt();

code = r.reverseText(text);

output += "\nReverser\nThe encrypted Reverse message is \n" + code + "\n";

code = r.decode(code);

output += "The decrypted Reverse message is \n" + code;

System.out.println(output);

display(output);

}

**static void display(String s)**{

JOptionPane.showMessageDialog(null, s, "Encrypt/decrypt", JOptionPane.INFORMATION\_MESSAGE);

}

}

**interface Constants**{

int WRAP\_AROUND = 26;

int ENCODE\_SHIFT = 3;

int DECODE\_SHIFT = 23;

public abstract String encode(String s);

public abstract String decode(String s);

}

**abstract class Cipher implements Constants**{

private String message;

StringBuffer encrypted\_message, decrypted\_message;

public Cipher(String text){

message = text;

}

**public final void encrypt()**{

encrypted\_message = new StringBuffer("");

StringTokenizer words = new StringTokenizer(message);

while (words.hasMoreTokens()){

String s = words.nextToken();

s = encode(s) + " ";

encrypted\_message.append(s);

}

}

**public final void decrypt(String message)**{

decrypted\_message = new StringBuffer("");

StringTokenizer words = new StringTokenizer(message);

while (words.hasMoreTokens()){

String s = words.nextToken();

s = decode(s) + " ";

decrypted\_message.append(s);

}

}

**String getEncodedMessage()**{

return encrypted\_message.toString();

}

**String getDecodedMessage()**{

return decrypted\_message.toString();

}

@Override

**public abstract String encode(String s);**

@Override

**public abstract String decode(String s);**

}

**class Caesar extends Cipher**{

**public Caesar(String s)**{

super(s);

}

@Override

**public String encode(String word)**{

StringBuffer result = new StringBuffer();

for (int i = 0; i < word.length(); i++){

char ch = word.charAt(i);

ch = determineCharacter(ch, Constants.ENCODE\_SHIFT);

result.append(ch);

}

return result.toString();

}

@Override

**public String decode(String word)**{

StringBuffer result = new StringBuffer();

for (int i = 0; i < word.length(); i++){

char ch = word.charAt(i);

ch = determineCharacter(ch, Constants.DECODE\_SHIFT);

result.append(ch);

}

return result.toString();

}

**public char determineCharacter(char ch, int shift)**{

if (Character.isLowerCase(ch))

ch = (char)('a' + (ch - 'a' + shift) % Constants.WRAP\_AROUND);

else if(Character.isUpperCase(ch))

ch = (char)('A' + (ch - 'A' + shift) % Constants.WRAP\_AROUND);

return ch;

}

}

**class Transpose extends Cipher**{

String reverseWord;

Transpose(String s){

super(s);

}

@Override

**public String encode(String word)**{

StringBuffer sb = new StringBuffer(word);

sb = sb.reverse();

return sb.toString();

}

@Override

**public String decode(String word)**{

return encode(word);

}

**public char determineCharacter(char ch, int shift)**{

if (Character.isLowerCase(ch))

ch = (char)('a' + (ch - 'a' + shift) % Constants.WRAP\_AROUND);

return ch;

}

}

**class Reverser extends Transpose**{

**Reverser(String s)**{

super(s);

}

**String reverseText(String word)**{

StringBuffer result = new StringBuffer();

int j = word.length();

j -= 1;

for (int i = j; i >= 0; i--){

char ch = word.charAt(i);

result.append(ch);

}

return result.toString();

}

}