Aim: To implement Modified Caesar Cipher
Modified Caesar cipher is an extension to Caesar cipher. Caesar cipher is not good because it can be analyzed by any attacker easily, so new concept was implemented to complicate the Caesar Cipher & increase the complexity of the attacker to decode it.

In the program we are implementing Modified Caesar cipher which is an example of substitution cipher. Program consist of two methods encrypt and decrypt. The encrypt method has two parameter one the plain text and second is key. In Modified Caesar cipher each alphabet of plain text is may not necessarily replaced by key bits down the order instead the value of key is incremented and then it is replaced with new key value. The decryption method also has two parameters one encrypted message and key. It does opposite process of encryption.

Program
```java
import javax.swing.JOptionPane;
public class ModifiedCaesar_Cipher {
public static void main(String[] args) {
    String plain_text;
    String key;
    String cipher1,cipher2;
    plain_text=JOptionPane.showInputDialog("Input the string to encrypt:");
    key=JOptionPane.showInputDialog("Input the key:");
    cipher1=encrypt(plain_text,key);
    JOptionPane.showMessageDialog (null, "Cipher Text is   " + cipher1, "Encryption Process", JOptionPane.PLAIN_MESSAGE);
    cipher2=decrypt(cipher1,key);
    JOptionPane.showMessageDialog (null, "Plain Text is  " + cipher2, "Decryption Process", JOptionPane.PLAIN_MESSAGE);
}
public static String encrypt(String str1,String key1) {
    int keylen=Integer.parseInt(key1);
    String encrypted="";
    for(int i=0;i<str1.length();i++)
    {
        int c=str1.charAt(i);
        if(Character.isUpperCase(c))
        {
            c=c+(keylen%26);
            if(c>'Z')
                c=c-26;
        }
        else if(Character.isLowerCase(c))
        {
            c=c+(keylen%26);
            if(c>'z')
                c=c-26;
        }
        keylen++;
        encrypted+=(char)c;
    }
    return encrypted;
}
public static String decrypt(String str1,String key1) {
    int keylen=Integer.parseInt(key1);
    String decrypted="";
    for(int i=0;i<str1.length();i++)
    {
        int c=str1.charAt(i);
        if(Character.isUpperCase(c))
        {
            c=c-(keylen%26);
            if(c>'Z')
                c=c+26;
        }
        else if(Character.isLowerCase(c))
        {
            c=c-(keylen%26);
            if(c>'z')
                c=c+26;
        }
        keylen++;
        decrypted+=(char)c;
    }
    return decrypted;
}
```
public static String decrypt(String str1, String key1) {
    int keylen = Integer.parseInt(key1);
    String decrypted = "";
    for (int i = 0; i < str1.length(); i++) {
        int c = str1.charAt(i);
        if (Character.isUpperCase(c)) {
            c = c - (keylen % 26);
            if (c < 'A')
                c = c + 26;
        } else if (Character.isLowerCase(c)) {
            c = c - (keylen % 26);
            if (c < 'a')
                c = c + 26;
        }
        decrypted += (char) c;
        keylen++;
    }
    return decrypted;
}

Output: