AIM:

2. Write a Program to implement AES.

IMPLEMENTATION

```
1. import javax.crypto.Cipher;
2. import javax.crypto.SecretKey;
3. import javax.crypto.SecretKeyFactory;
4. import javax.crypto.spec.IvParameterSpec;
5. import javax.crypto.spec.PBEKeySpec;
6. import javax.crypto.spec.SecretKeySpec;
7. import java.nio.charset.StandardCharsets;
8. import java.security.InvalidAlgorithmParameterException;
9. import java.security.InvalidKeyException;
10. import java.security.NoSuchAlgorithmException;
11. import java.security.spec.InvalidKeySpecException;
12. import java.security.spec.KeySpec;
13. import java.util.Base64;
14. import javax.crypto.BadPaddingException;
15. import javax.crypto.IllegalBlockSizeException;
16. import javax.crypto.NoSuchPaddingException;
17. public class AESExample
18. {
19.    /* Private variable declaration */
20.    private static final String SECRET_KEY = "123456789";
21.    private static final String SALTVALUE = "abcdefg";
22.
23.    /* Encryption Method */
24.    public static String encrypt(String strToEncrypt)
25.    {
26.    try
27.    {
28.      /* Declare a byte array. */
29.      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

```
30.      IvParameterSpec ivspec = new IvParameterSpec(iv);
31.      /* Create factory for secret keys. */
32.      SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacS
    HA256");
33.      /* PBEKeySpec class implements KeySpec interface. */
34.      KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(), SALTVALUE.getB
    ytes(), 65536, 256);
35.      SecretKey tmp = factory.generateSecret(spec);
36.      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
37.      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
38.      cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);
39.      /* Retruns encrypted value. */
40.      return Base64.getEncoder()
41. .encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));

42.    }
43.    catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlg
    orithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSi
    zeException | NoSuchPaddingException e)
44.    {
45.      System.out.println("Error occured during encryption: " + e.toString());
46.    }
47.    return null;
48.    }
49.
50.    /* Decryption Method */
51.    public static String decrypt(String strToDecrypt)
52.    {
53.    try
54.    {
55.      /* Declare a byte array. */
56.      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
57.      IvParameterSpec ivspec = new IvParameterSpec(iv);
58.      /* Create factory for secret keys. */
```

```
59.      SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacS
    HA256");
60.      /* PBEKeySpec class implements KeySpec interface. */
61.      KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(), SALTVALUE.getB
    ytes(), 65536, 256);
62.      SecretKey tmp = factory.generateSecret(spec);
63.      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
64.      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
65.      cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);
66.      /* Retruns decrypted value. */
67.      return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)))
    ;
68.    }
69.    catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlg
    orithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSi
    zeException | NoSuchPaddingException e)
70.    {
71.      System.out.println("Error occured during decryption: " + e.toString());
72.    }
73.    return null;
74.    }
75.    /* Driver Code */
76.    public static void main(String[] args)
77.    {
78.      /* Message to be encrypted. */
79.      String originalval = "AES Encryption";
80.      /* Call the encrypt() method and store result of encryption. */
81.      String encryptedval = encrypt(originalval);
82.      /* Call the decrypt() method and store result of decryption. */
83.      String decryptedval = decrypt(encryptedval);
84.      /* Display the original message, encrypted message and decrypted message
    on the console. */
85.      System.out.println("Original value: " + originalval);
86.      System.out.println("Encrypted value: " + encryptedval);
```

```
87.         System.out.println("Decrypted value: " + decryptedval);
88.     }
89. }
```

OUTPUT: