Name – Jay S. Chatpalliwar

PRN-21510018

Assignment no 06

import java.nio.charset.StandardCharsets;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import java.security.spec.KeySpec;

import java.util.Base64;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

public class AESExample

{

    /\* Private variable declaration \*/

    private static final String SECRET\_KEY = "123456789";

    private static final String SALTVALUE = "abcdefg";

    /\* Encryption Method \*/

    public static String encrypt(String strToEncrypt)

    {

    try

    {

      /\* Declare a byte array. \*/

      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

      IvParameterSpec ivspec = new IvParameterSpec(iv);

      /\* Create factory for secret keys. \*/

      SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");

      /\* PBEKeySpec class implements KeySpec interface. \*/

      KeySpec spec = new PBEKeySpec(SECRET\_KEY.toCharArray(), SALTVALUE.getBytes(), 65536, 256);

      SecretKey tmp = factory.generateSecret(spec);

      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");

      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

      cipher.init(Cipher.ENCRYPT\_MODE, secretKey, ivspec);

      /\* Retruns encrypted value. \*/

      return Base64.getEncoder()

.encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF\_8)));

    }

    catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e)

    {

      System.out.println("Error occured during encryption: " + e.toString());

    }

    return null;

    }

    /\* Decryption Method \*/

    public static String decrypt(String strToDecrypt)

    {

    try

    {

      /\* Declare a byte array. \*/

      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

      IvParameterSpec ivspec = new IvParameterSpec(iv);

      /\* Create factory for secret keys. \*/

      SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");

      /\* PBEKeySpec class implements KeySpec interface. \*/

      KeySpec spec = new PBEKeySpec(SECRET\_KEY.toCharArray(), SALTVALUE.getBytes(), 65536, 256);

      SecretKey tmp = factory.generateSecret(spec);

      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");

      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");

      cipher.init(Cipher.DECRYPT\_MODE, secretKey, ivspec);

      /\* Retruns decrypted value. \*/

      return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));

    }

    catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e)

    {

      System.out.println("Error occured during decryption: " + e.toString());

    }

    return null;

    }

    /\* Driver Code \*/

    public static void main(String[] args)

    {

        /\* Message to be encrypted. \*/

        String originalval = "AES Encryption";

        /\* Call the encrypt() method and store result of encryption. \*/

        String encryptedval = encrypt(originalval);

        /\* Call the decrypt() method and store result of decryption. \*/

        String decryptedval = decrypt(encryptedval);

        /\* Display the original message, encrypted message and decrypted message on the console. \*/

        System.out.println("Original value: " + originalval);

        System.out.println("Encrypted value: " + encryptedval);

        System.out.println("Decrypted value: " + decryptedval);

    }

}

