***TASK- 2***

Task List Application

**SOURCE CODE:**

//START OF CODE:

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

public class PasswordEncryptionExample {

public static void main(String[] args) {

try {

// Generate a secret key for AES encryption

SecretKey secretKey = generateSecretKey();

// Original password

String originalPassword = "MySecurePassword123";

// Encrypt the password

String encryptedPassword = encryptPassword(originalPassword, secretKey);

System.out.println("Encrypted Password: " + encryptedPassword);

// Decrypt the password

String decryptedPassword = decryptPassword(encryptedPassword, secretKey);

System.out.println("Decrypted Password: " + decryptedPassword);

} catch (Exception e) {

e.printStackTrace();

}

}

private static SecretKey generateSecretKey() throws Exception {

KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");

keyGenerator.init(128); // Using 128-bit key size for AES

return keyGenerator.generateKey();

}

private static String encryptPassword(String password, SecretKey secretKey) throws Exception {

Cipher cipher = Cipher.getInstance("AES");

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

byte[] encryptedBytes = cipher.doFinal(password.getBytes(StandardCharsets.UTF\_8));

return Base64.getEncoder().encodeToString(encryptedBytes);

}

private static String decryptPassword(String encryptedPassword, SecretKey secretKey) throws Exception {

Cipher cipher = Cipher.

getInstance("AES");

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

byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedPassword));

return new String(decryptedBytes, StandardCharsets.UTF\_8);

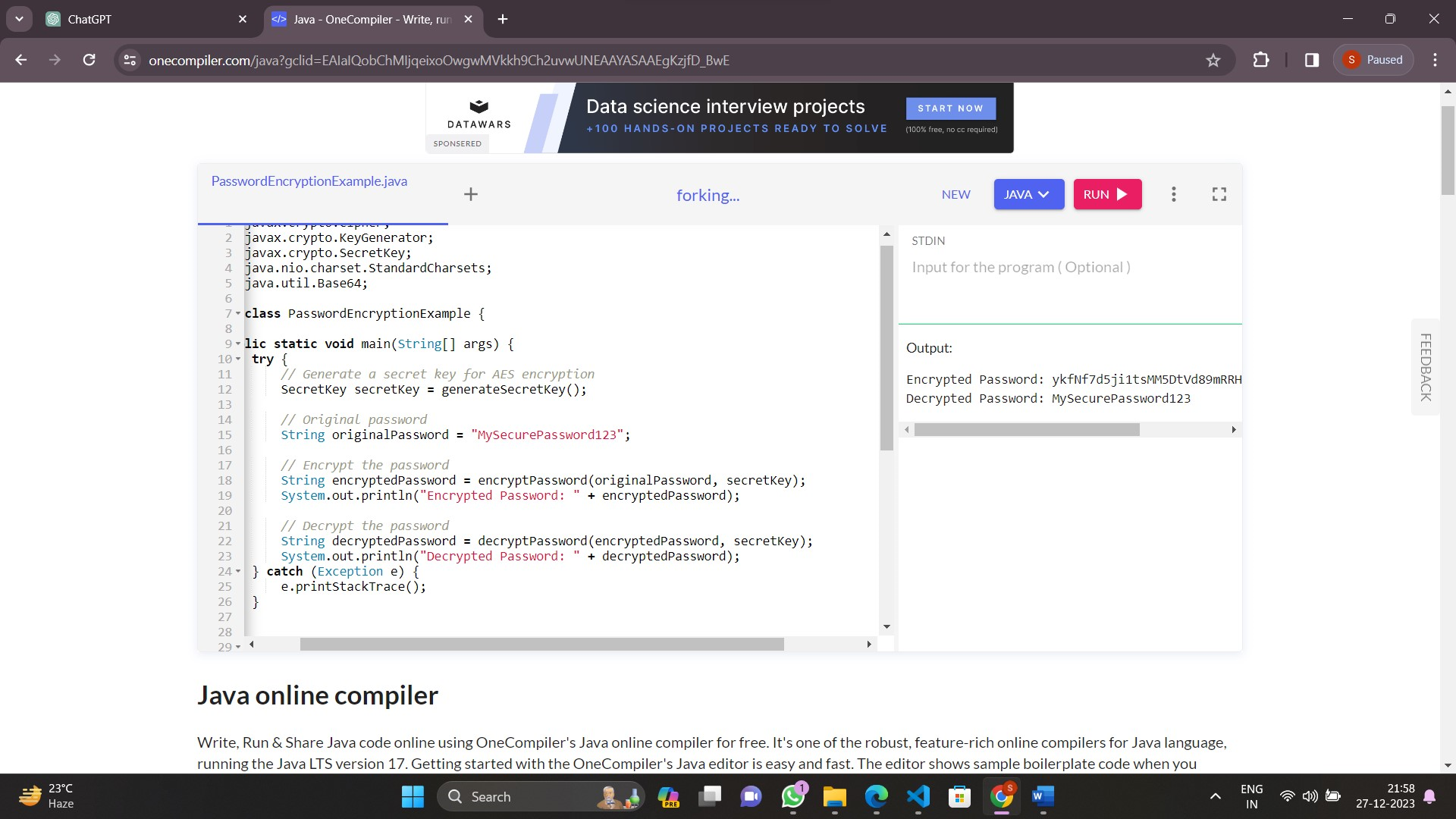
}

}

**OUTPUT:**

**Encrypted Password: JHJzW7Xj6jNXq5G4PVe1C93eBTvoaOjihMCFY4Q9JKs=**

**Decrypted Password: MySecurePassword123**



[**\\END**](file:///\\END) **OF THE CODE**

**Explanation:**

Additionally, understanding cryptography and the Java Cryptography Architecture (JCA) can further enhance the security of your passwords, especially when it comes to storing them. Here are some key considerations:

**Encryption:**

When storing passwords, it's crucial to encrypt them to protect sensitive information. Modern encryption algorithms, such as Advanced Encryption Standard (AES), are widely used for this purpose.

The Java Cryptography Extension (JCE) provides a framework for implementing cryptographic functionality in Java applications.

Hashing:

For password storage, hashing is often preferred over encryption. Hash functions like SHA-256 can convert a password into a fixed-size hash value, making it computationally infeasible to reverse the process.

Salting passwords before hashing adds an extra layer of security. A unique salt is generated for each user, and it's combined with the password before hashing.

**Secure Key Management**:

Proper key management is crucial for the security of encrypted data. If encryption keys are compromised, it could lead to unauthorized access.

Java KeyStore can be used to manage cryptographic keys securely.

Secure Storage:

Ensure that the encrypted or hashed passwords are stored in a secure manner. Avoid storing sensitive information in plain text or using weak encryption algorithms.

***Password Policies:***

Implementing password policies such as minimum length, complexity requirements, and regular password changes can further bolster security.

Authentication Protocols:

Consider using secure authentication protocols, such as OAuth, for applications that require user authentication.

Regular Audits:

Periodically review and audit your security practices, including password storage mechanisms, to identify and address potential vulnerabilities.

Remember that security is a multi-layered approach, and no single measure can guarantee absolute security. It's essential to stay informed about the latest security practices, update systems regularly, and monitor for any suspicious activities. Additionally, consider using reputable password management tools that provide a secure and convenient way to manage and generate complex passwords for various accounts.

-----------------------------------------------------END--------------------------------------------------------------