Assignment 6: Digital Signature Implementation in Java

Name: Shubhankar jakate
PRN : 22310371
Roll No = 382019

Objectives

1. Understand the concept of **digital signatures** in information security.
2. Learn how to use **Java Cryptography Architecture (JCA)** for signing and verification.
3. Implement a **menu-driven Java program** to generate keys, sign messages, and verify signatures.

Theory Summary

What Is a Digital Signature?

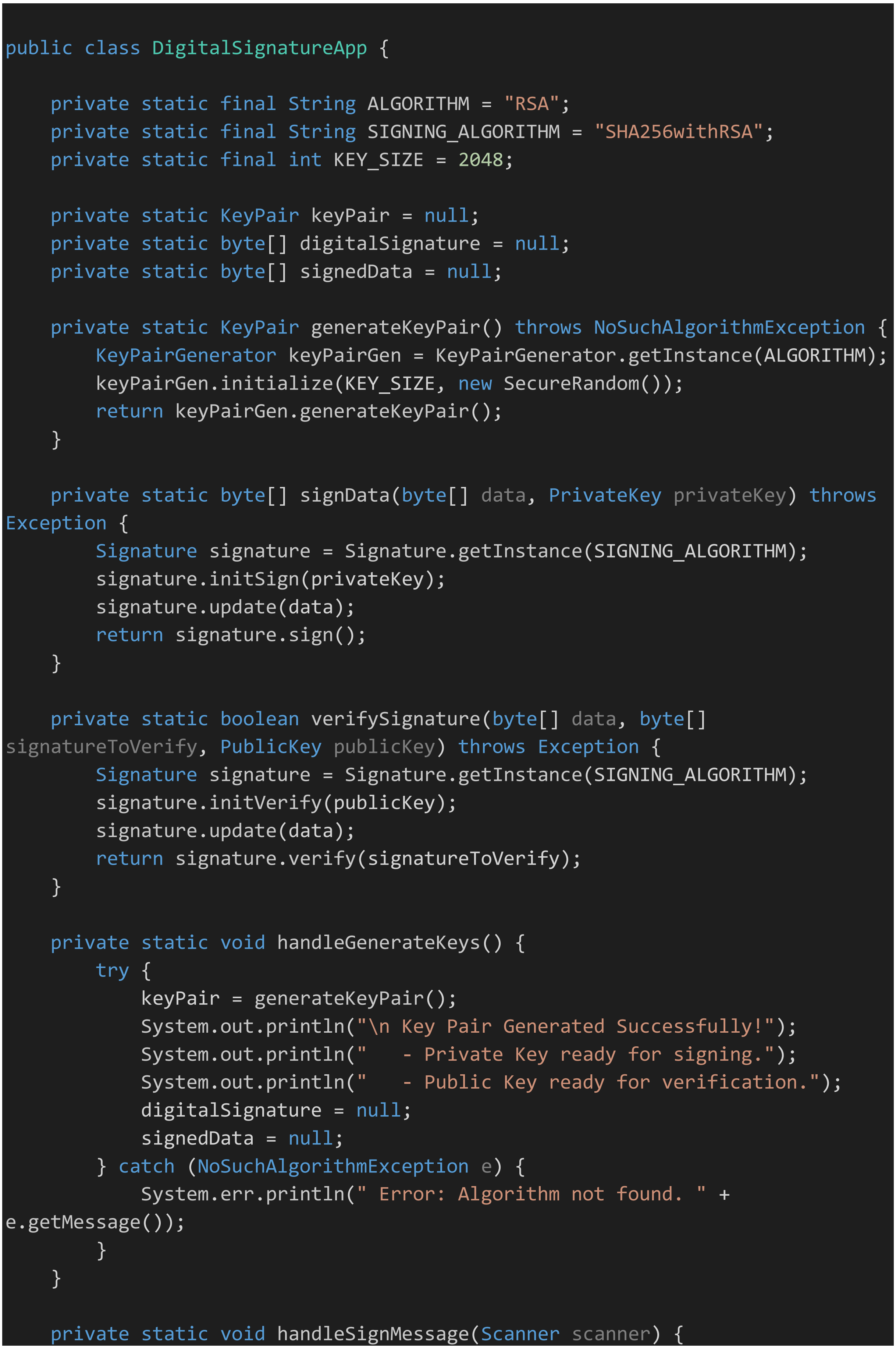
- A **digital signature** is a cryptographic technique used to verify the **authenticity** and **integrity** of digital data.
- It uses **asymmetric encryption**:
  - **Private key** → for signing
  - **Public key** → for verification
- Ensures:
  - **Integrity**: Message hasn't been altered.
  - **Authentication**: Sender is verified.
  - **Non-repudiation**: Sender cannot deny the message.

How It Works

1. **Key Generation**: Create a public-private key pair.
2. **Signing**: Hash the message and encrypt the hash using the private key.
3. **Verification**: Decrypt the signature using the public key and compare with the hash of the received message.

JAVA PROGRAM :

```java
import java.security.*;
import java.util.Base64;
import java.util.Scanner;
```

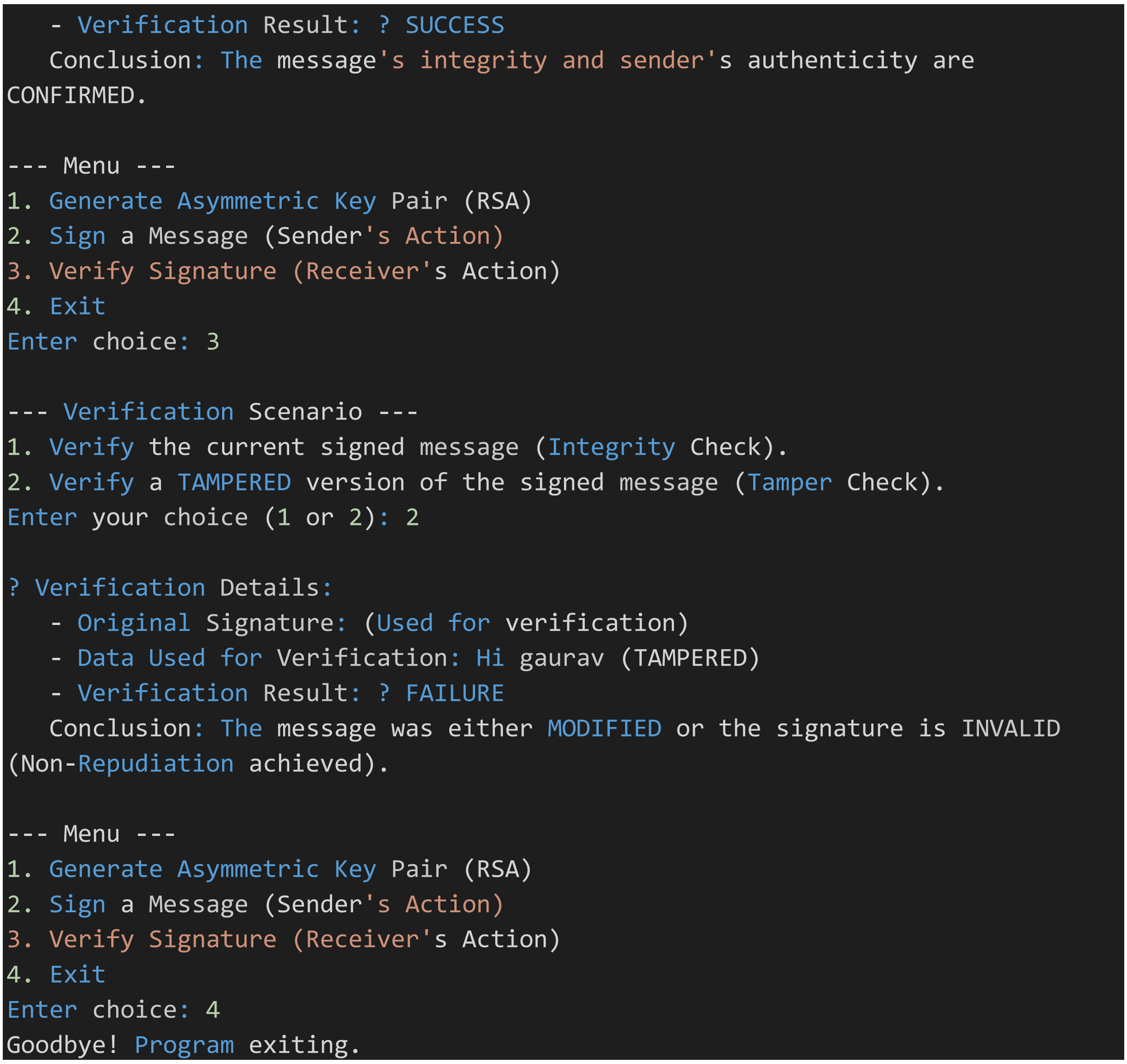


```
public class DigitalSignatureApp {

    private static final String ALGORITHM = "RSA";
    private static final String SIGNING_ALGORITHM = "SHA256withRSA";
    private static final int KEY_SIZE = 2048;

    private static KeyPair keyPair = null;
    private static byte[] digitalSignature = null;
    private static byte[] signedData = null;

    private static KeyPair generateKeyPair() throws NoSuchAlgorithmException {
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance(ALGORITHM);
        keyPairGen.initialize(KEY_SIZE, new SecureRandom());
        return keyPairGen.generateKeyPair();
    }

    private static byte[] signData(byte[] data, PrivateKey privateKey) throws
Exception {
        Signature signature = Signature.getInstance(SIGNING_ALGORITHM);
        signature.initSign(privateKey);
        signature.update(data);
        return signature.sign();
    }

    private static boolean verifySignature(byte[] data, byte[]
signatureToVerify, PublicKey publicKey) throws Exception {
        Signature signature = Signature.getInstance(SIGNING_ALGORITHM);
        signature.initVerify(publicKey);
        signature.update(data);
        return signature.verify(signatureToVerify);
    }

    private static void handleGenerateKeys() {
        try {
            keyPair = generateKeyPair();
            System.out.println("\n Key Pair Generated Successfully!");
            System.out.println("   - Private Key ready for signing.");
            System.out.println("   - Public Key ready for verification.");
            digitalSignature = null;
            signedData = null;
        } catch (NoSuchAlgorithmException e) {
            System.err.println(" Error: Algorithm not found. " +
e.getMessage());
        }
    }

    private static void handleSignMessage(Scanner scanner) {
```

```java
        if (keyPair == null) {
            System.out.println("\n Please generate a Key Pair first (Option 1).");
            return;
        }

        System.out.print("\n Enter the message to sign: ");
        String message = scanner.nextLine();

        try {
            signedData = message.getBytes("UTF-8");
            digitalSignature = signData(signedData, keyPair.getPrivate());

            System.out.println("\n Message Signed Successfully!");
            System.out.println("   - Original Data: '" + message + "'");
            System.out.println("   - Digital Signature (Base64 Encoded): " +
                    Base64.getEncoder().encodeToString(digitalSignature));

        } catch (Exception e) {
            System.err.println(" Signing failed: " + e.getMessage());
        }
    }

    private static void handleVerifySignature(Scanner scanner) {
        if (keyPair == null || digitalSignature == null) {
            System.out.println("\n Please Generate Keys (Option 1) and Sign a Message (Option 2) first.");
            return;
        }

        System.out.println("\n--- Verification Scenario ---");
        System.out.println("1. Verify the current signed message (Integrity Check).");
        System.out.println("2. Verify a TAMPERED version of the signed message (Tamper Check).");
        System.out.print("Enter your choice (1 or 2): ");
        String choice = scanner.nextLine();

        byte[] dataToVerify = signedData;
        String dataDisplay = new String(signedData);

        if ("2".equals(choice)) {
            dataDisplay += " (TAMPERED)";
            dataToVerify = (dataDisplay + " ").getBytes();
        }

        try {
            boolean isVerified = verifySignature(dataToVerify,
digitalSignature, keyPair.getPublic());

            System.out.println("\n Verification Details:");
            System.out.println("   - Original Signature: (Used for
verification)");
            System.out.println("   - Data Used for Verification: " +
dataDisplay);
            System.out.println("   - Verification Result: " + (isVerified ? "
SUCCESS" : " FAILURE"));

            if (isVerified) {
                System.out.println("   Conclusion: The message's integrity and
sender's authenticity are CONFIRMED.");
            } else {
                System.out.println("   Conclusion: The message was either
MODIFIED or the signature is INVALID (Non-Repudiation achieved).");
            }

        } catch (Exception e) {
            System.err.println(" Verification failed: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        System.out.println("=============================================");
        System.out.println("  JAVA DIGITAL SIGNATURE IMPLEMENTATION (RSA)  ");
        System.out.println("=============================================");

        while (true) {
            System.out.println("\n--- Menu ---");
            System.out.println("1. Generate Asymmetric Key Pair (RSA)");
            System.out.println("2. Sign a Message (Sender's Action)");
            System.out.println("3. Verify Signature (Receiver's Action)");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");

            try {
                choice = Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e) {
                System.out.println("\n Invalid input. Please enter a number
from the menu.");
                continue;
            }
            switch (choice) {
                case 1:
                    handleGenerateKeys();
                    break;
                case 2:
                    handleSignMessage(scanner);
                    break;
                case 3:
                    handleVerifySignature(scanner);
                    break;
                case 4:
                    System.out.println("Goodbye! Program exiting.");
                    scanner.close();
                    return;
                default:
                    System.out.println("\n Invalid choice. Please try 
again.");
            }
        }
    }
}
```

OUTPUT :

```
=============================================
  JAVA DIGITAL SIGNATURE IMPLEMENTATION (RSA)
=============================================

--- Menu ---
1. Generate Asymmetric Key Pair (RSA)
2. Sign a Message (Sender's Action)
3. Verify Signature (Receiver's Action)
4. Exit
Enter choice: 1

? Key Pair Generated Successfully!
   - Private Key ready for signing.
   - Public Key ready for verification.

--- Menu ---
1. Generate Asymmetric Key Pair (RSA)
2. Sign a Message (Sender's Action)
3. Verify Signature (Receiver's Action)
4. Exit
Enter choice: 2

?? Enter the message to sign: Hi gaurav

? Message Signed Successfully!
   - Original Data: 'Hi gaurav'
   - Digital Signature (Base64 Encoded):
lGaXea93L8Cc5sMbGzcgdUAKgwwPnIEZdSa+eYlOin64chCXpdEWWxmXQ9rVO2eSQMQvtPMT0/vq4k
dv5Jyj0y1ctGLfumziUr9baPwKgVlfldp+PFf+WNlyBfylIECyfPPV12O94cu5dCG0hbEgl1lJ8zEM
UcDXwUMoj4BnQVrU5j2FBhkYNUrB7vt4EBvPKd5Wke65hdMYhQEkdP78xtae49fTcjjn9bVzfi+Rhf
9AWKtOH653QamjAKfRhBSLfBiokpTqN9w/rCw0dgGVM+uNood5KwZoHyrwBlDdpKT7ZZwWcPUt2Uzh
JpBFiN+MpcTxaQ5fzuIEWF/pnYjCRA==

--- Menu ---
1. Generate Asymmetric Key Pair (RSA)
2. Sign a Message (Sender's Action)
3. Verify Signature (Receiver's Action)
4. Exit
Enter choice: 3

--- Verification Scenario ---
1. Verify the current signed message (Integrity Check).
2. Verify a TAMPERED version of the signed message (Tamper Check).
Enter your choice (1 or 2): 1

? Verification Details:
   - Original Signature: (Used for verification)
   - Data Used for Verification: Hi gaurav
```



```
   - Verification Result: ? SUCCESS
   Conclusion: The message's integrity and sender's authenticity are
CONFIRMED.

--- Menu ---
1. Generate Asymmetric Key Pair (RSA)
2. Sign a Message (Sender's Action)
3. Verify Signature (Receiver's Action)
4. Exit
Enter choice: 3

--- Verification Scenario ---
1. Verify the current signed message (Integrity Check).
2. Verify a TAMPERED version of the signed message (Tamper Check).
Enter your choice (1 or 2): 2

? Verification Details:
   - Original Signature: (Used for verification)
   - Data Used for Verification: Hi gaurav (TAMPERED)
   - Verification Result: ? FAILURE
   Conclusion: The message was either MODIFIED or the signature is INVALID
(Non-Repudiation achieved).

--- Menu ---
1. Generate Asymmetric Key Pair (RSA)
2. Sign a Message (Sender's Action)
3. Verify Signature (Receiver's Action)
4. Exit
Enter choice: 4
Goodbye! Program exiting.
```