Serializable Assignment – 1

"Java Case Study - Serialization" assignment centres around mastering the concept of Java Serialization, covering topics such as serialization basics, implementing the Serializable interface, handling transient variables, and deserialization. Participants will gain hands-on experience in designing, serializing, and deserializing objects, with a focus on creating resilient and efficient serialization mechanisms.

Problem Statement 1: You are working on a banking application that requires storing and retrieving customer information securely. Design a Java program that serializes and deserializes customer objects, ensuring sensitive information is properly handled. Implement a mechanism to encrypt and decrypt the serialized data for enhanced security.

Learning Outcomes:

- Mastery of applying serialization in a secure customer data storage scenario.
- Proficiency in implementing encryption and decryption during serialization.
- Understanding the importance of secure data handling in real-world applications.

//code

```
import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.io.*;

import java.util.Base64;


class Customer implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;

    private String accountNumber;

    private double balance;

   public Customer(String name, String accountNumber, double balance) {

        this.name = name;

        this.accountNumber = accountNumber;

        this.balance = balance;

    }
```

```
    public String getName() {
        return name;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    @Override
    public String toString() {
        return "Customer{" +
                "name='" + name + '\'' +
                ", accountNumber='" + accountNumber + '\'' +
                ", balance=" + balance +
                '}';
    }
}

public class BankingApplication {
    private static final String ENCRYPTION_ALGORITHM = "AES";
    private static final String ENCRYPTED_FILE = "encrypted_customer_data.dat";

    public static void main(String[] args) throws Exception {
        // Generate AES Key
        SecretKey secretKey = generateKey();
```

```
        // Customer data
        Customer customer = new Customer("John Doe", "123456789", 10000.50);

        // Serialize and Encrypt Customer Data
        serializeAndEncrypt(customer, secretKey);

        // Deserialize and Decrypt Customer Data
        Customer decryptedCustomer = decryptAndDeserialize(secretKey);

        // Display Customer Info
        System.out.println("Decrypted Customer Data:");
        System.out.println(decryptedCustomer);
    }

    // Generate a random AES key
    private static SecretKey generateKey() throws Exception {
        KeyGenerator keyGenerator = KeyGenerator.getInstance(ENCRYPTION_ALGORITHM);
        keyGenerator.init(128); // 128-bit AES
        return keyGenerator.generateKey();
    }

    // Serialize and encrypt the customer object
    private static void serializeAndEncrypt(Customer customer, SecretKey secretKey) throws Exception
{
        // Serialize Customer Object to Byte Array
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);
        objectOutputStream.writeObject(customer);
        objectOutputStream.close();
        byte[] serializedData = byteArrayOutputStream.toByteArray();
```

```
        // Encrypt Data
        Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedData = cipher.doFinal(serializedData);

        // Save Encrypted Data to File
        try (FileOutputStream fileOutputStream = new FileOutputStream(ENCRYPTED_FILE)) {
            fileOutputStream.write(encryptedData);
        }

        System.out.println("Customer data encrypted and saved.");
    }

    // Decrypt and deserialize the customer object
    private static Customer decryptAndDeserialize(SecretKey secretKey) throws Exception {
        // Read Encrypted Data from File
        byte[] encryptedData;
        try (FileInputStream fileInputStream = new FileInputStream(ENCRYPTED_FILE)) {
            encryptedData = fileInputStream.readAllBytes();
        }
        // Decrypt Data
        Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedData = cipher.doFinal(encryptedData);

        // Deserialize Customer Object
        ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(decryptedData);
        ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
        return (Customer) objectInputStream.readObject();
    }
}
```