Write a Java program to create an interface Encryptable with methods encrypt (String data) and decrypt (String encryptedData) that define encryption and decryption operations. Create two classes AES and RSA that implement the Encryptable interface and provide their own encryption and decryption algorithms.

Sample Solution:

Java Code:

```
// Encryptable.java

// Declare the Encryptable interface
interface Encryptable {
    // Declare the abstract method "encrypt" that classes implementing this
    String encrypt(String data);

    // Declare the abstract method "decrypt" that classes implementing this
    String decrypt(String encryptedData);
}
```

```
// AES.java

// Import necessary libraries for cryptographic operations
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Base64;

// Declare the AES class, which implements the Encryptable interface
class AES implements Encryptable {
    // Define the AES algorithm as a constant
    private static final String AES_ALGORITHM = "AES";

    // Declare a SecretKey to store the encryption key
    private SecretKey secretKey;
```

```
    // Constructor to initialize the encryption key
    public AES() {
        try {
            KeyGenerator keyGen = KeyGenerator.getInstance(AES_ALGORITHM);
            keyGen.init(128);
            secretKey = keyGen.generateKey();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Implement the "encrypt" method required by the Encryptable interface
    public String encrypt(String data) {
        try {
            Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

            byte[] encryptedBytes = cipher.doFinal(data.getBytes());
            return Base64.getEncoder().encodeToString(encryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    // Implement the "decrypt" method required by the Encryptable interface
    public String decrypt(String encryptedData) {
        try {
            byte[] encryptedBytes = Base64.getDecoder().decode(encryptedDat

            Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
            cipher.init(Cipher.DECRYPT_MODE, secretKey);

            byte[] decryptedBytes = cipher.doFinal(encryptedBytes);
            return new String(decryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

```
    }
```

```
    // RSA.java

    // Import necessary libraries for cryptographic operations
    import javax.crypto.Cipher;
    import javax.crypto.KeyGenerator;
    import javax.crypto.SecretKey;
    import java.security.KeyPair;
    import java.security.KeyPairGenerator;
    import java.security.PrivateKey;
    import java.security.PublicKey;
    import java.util.Base64;

    // Declare the RSA class, which implements the Encryptable interface
    class RSA implements Encryptable {
        // Define the RSA algorithm as a constant
        private static final String RSA_ALGORITHM = "RSA";

        // Declare a KeyPair to store the public and private keys
        private KeyPair keyPair;

        // Constructor to generate a KeyPair for encryption and decryption
        public RSA() {
            try {
                KeyPairGenerator keyGen = KeyPairGenerator.getInstance(RSA_ALG
                keyGen.initialize(2048);
                keyPair = keyGen.generateKeyPair();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        // Implement the "encrypt" method required by the Encryptable interface
        public String encrypt(String data) {
            try {
                // Get the public key from the KeyPair
                PublicKey publicKey = keyPair.getPublic();
                Cipher cipher = Cipher.getInstance(RSA_ALGORITHM);
                cipher.init(Cipher.ENCRYPT_MODE, publicKey);
```

```
            byte[] encryptedBytes = cipher.doFinal(data.getBytes());
            return Base64.getEncoder().encodeToString(encryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    // Implement the "decrypt" method required by the Encryptable interface
    public String decrypt(String encryptedData) {
        try {
            // Get the private key from the KeyPair
            PrivateKey privateKey = keyPair.getPrivate();
            byte[] encryptedBytes = Base64.getDecoder().decode(encryptedDat

            Cipher cipher = Cipher.getInstance(RSA_ALGORITHM);
            cipher.init(Cipher.DECRYPT_MODE, privateKey);

            byte[] decryptedBytes = cipher.doFinal(encryptedBytes);
            return new String(decryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

```
// Main.java

// Import necessary libraries for cryptographic operations
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Base64;
```

```
// Declare the Main class
public class Main {
    public static void main(String[] args) {
        // Create an instance of the AES class for encryption and decryptic
        Encryptable aes = new AES();

        // Encrypt data using AES encryption and store the result
        String encryptedDataAES = aes.encrypt("Java Interface.");
        System.out.println("AES Encrypted: " + encryptedDataAES);

        // Decrypt the AES-encrypted data and store the result
        String decryptedDataAES = aes.decrypt(encryptedDataAES);
        System.out.println("AES Decrypted: " + decryptedDataAES);

        // Create an instance of the RSA class for encryption and decryptic
        Encryptable rsa = new RSA();

        // Encrypt data using RSA encryption and store the result
        String encryptedDataRSA = rsa.encrypt("Java Interface.");
        System.out.println("RSA Encrypted: " + encryptedDataRSA);

        // Decrypt the RSA-encrypted data and store the result
        String decryptedDataRSA = rsa.decrypt(encryptedDataRSA);
        System.out.println("RSA Decrypted: " + decryptedDataRSA);
    }
}
```

Sample Output:

```
AES Encrypted: SIRcgpS5yDqD7/MhH6Q2pg==
AES Decrypted: Java Interface.
RSA Encrypted: ep7aeBcALwnW44f2YFl+XfWkCE8QNWblnUgOeXjI1u0gnz6pSLLZekff6wBQbujUD
RSA Decrypted: Java Interface.
```

Explanation:

In the above exercise –

- The "AES" class and "RSA" class both implement the Encryptable interface and provide their own implementations of the encrypt() and decrypt() methods. The "AES" class performs AES encryption and decryption, while the "RSA" class performs RSA encryption and decryption.

- In the main() method, we create instances of the "AES" and "RSA" classes. We then call the encrypt() method on each instance, passing in a string to encrypt, and storing the result in a variable. Finally, we call the decrypt() method on each instance, passing in the encrypted string, and printing the decrypted data.

Flowchart of Encryptable Java: