Berikut adalah 3 buah enkripsi klasik dan 1 buah enkripsi modern beserta implementasinya pada Java:

1. Caesar Cipher

Caesar Cipher adalah salah satu algoritma enkripsi klasik yang paling sederhana. Algoritma ini menggeser setiap huruf pada pesan sejauh n kali, dengan n merupakan kunci enkripsi. Berikut adalah contoh implementasi Caesar Cipher pada Java:

```
public static String encryptCaesar(String plainText, int shift) {
    StringBuilder cipherText = new StringBuilder();
    for (int i = 0; i < plainText.length(); i++) {
        char ch = (char) (((int) plainText.charAt(i) + shift - 65) % 26 + 65);
        cipherText.append(ch);
    }
    return cipherText.toString();
}
```

2. Vigenere Cipher

Vigenere Cipher adalah algoritma enkripsi klasik yang menggunakan tabel Vigenere untuk mengenkripsi pesan. Tabel ini terdiri dari 26 baris dan 26 kolom, di mana setiap baris dan kolom mewakili alfabet A-Z. Berikut adalah contoh implementasi Vigenere Cipher pada Java:

```
public static String encryptVigenere(String plainText, String key) {
    StringBuilder cipherText = new StringBuilder();
    int keyIndex = 0;
    for (int i = 0; i < plainText.length(); i++) {
        char ch = (char) (((int) plainText.charAt(i) + (int) key.charAt(keyIndex) - 2 * 65) % 26 +
        cipherText.append(ch);
        keyIndex = ++keyIndex % key.length();
    }
    return cipherText.toString();
}
```

3. Hill Cipher

Hill Cipher adalah algoritma enkripsi klasik yang menggunakan matriks sebagai kunci enkripsi. Algoritma ini mengenkripsi pesan dengan membagi pesan menjadi blok-blok huruf, kemudian mengalikan setiap blok dengan matriks kunci. Berikut adalah contoh implementasi Hill Cipher pada Java:

```
public static String encryptHill(String plainText, int[][] key) {
    StringBuilder cipherText = new StringBuilder();
    int n = key.length;
    int[] plainTextVector = new int[n];
    for (int i = 0; i < plainText.length(); i += n) {
        for (int j = 0; j < n; j++) {
            plainTextVector[j] = plainText.charAt(i + j) - 65;
        }
        for (int j = 0; j < n; j++) {
            int sum = 0;
            for (int k = 0; k < n; k++) {
                sum += key[j][k] * plainTextVector[k];
            }
            cipherText.append((char) ((sum % 26) + 65));
        }
    }
    return cipherText.toString();
}
```

4. Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) adalah algoritma enkripsi modern yang digunakan secara luas untuk mengamankan data. AES menggunakan blok cipher dengan ukuran blok 128 bit dan kunci enkripsi dengan panjang 128, 192, atau 256 bit. Berikut adalah contoh implementasi AES pada Java menggunakan library Bouncy Castle:

```
import org.bouncycastle.crypto.BlockCipher;
import org.bouncycastle.crypto.engines.AESEngine;
import org.bouncycastle.crypto.modes.CBCBlockCipher;
import org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher;
import org.bouncycastle.crypto.params.KeyParameter;
import org.bouncycastle.crypto.params.ParametersWithIV;

public static byte[] encryptAES(byte[] plainText, byte[] key, byte[] iv) throws Exception {
    BlockCipher engine = new AESEngine();
    CBCBlockCipher cipher = new CBCBlockCipher(engine);
    PaddedBufferedBlockCipher paddedCipher = new PaddedBufferedBlockCipher(cipher);
    KeyParameter keyParam = new KeyParameter(key);
    ParametersWithIV params = new ParametersWithIV(keyParam, iv);
    paddedCipher.init(true, params);
    byte[] cipherText = new byte[paddedCipher.getOutputSize(plainText.length)];
    int outputLen = paddedCipher.processBytes(plainText, 0, plainText.length, cipherText, 0);
    paddedCipher.doFinal(cipherText, outputLen);
```

```
        return cipherText;
    }
```

0s completed at 9:45 AM