Caesar Cipher

```
import java.util.Scanner;

public class EncryptDecrypt {

    // Encrypt function
    public static String encrypt(String text, int key) {
        StringBuilder encryptedText = new StringBuilder();

        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);
            if (Character.isLetterOrDigit(ch)) {
                if (Character.isLowerCase(ch)) {
                    ch = (char) ((ch - 'a' + key) % 26 + 'a');
                } else if (Character.isUpperCase(ch)) {
                    ch = (char) ((ch - 'A' + key) % 26 + 'A');
                } else if (Character.isDigit(ch)) {
                    ch = (char) ((ch - '0' + key) % 10 + '0');
                }
                encryptedText.append(ch);
            } else {
                System.out.println("Invalid Message");
                return null;
            }
        }
        return encryptedText.toString();
    }

    // Decrypt function
```

```
public static String decrypt(String text, int key) {
    StringBuilder decryptedText = new StringBuilder();

    for (int i = 0; i < text.length(); i++) {
        char ch = text.charAt(i);
        if (Character.isLetterOrDigit(ch)) {
            if (Character.isLowerCase(ch)) {
                ch = (char) ((ch - 'a' - key + 26) % 26 + 'a');
            } else if (Character.isUpperCase(ch)) {
                ch = (char) ((ch - 'A' - key + 26) % 26 + 'A');
            } else if (Character.isDigit(ch)) {
                ch = (char) ((ch - '0' - key + 10) % 10 + '0');
            }
            decryptedText.append(ch);
        } else {
            System.out.println("Invalid Message");
            return null;
        }
    }
    return decryptedText.toString();
}

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

    // Input for the message to encrypt and the key
    System.out.print("Enter a message to encrypt: ");
    String text = scanner.nextLine();

    System.out.print("Enter the key: ");
    int key = scanner.nextInt();
```

```
        // Encrypt the message
        String encryptedMessage = encrypt(text, key);
        if (encryptedMessage != null) {
            System.out.println("Encrypted message: " + encryptedMessage);

            // Decrypt the message
            String decryptedMessage = decrypt(encryptedMessage, key);
            if (decryptedMessage != null) {
                System.out.println("Decrypted message: " + decryptedMessage);
            }
        }

        scanner.close();
    }
}
```

Substitution Cipher

```
import java.util.Scanner;

public class SubstitutionCipher {

    // Encrypt function
    public static void encrypt(StringBuilder message, String key) {
        for (int i = 0; i < message.length(); i++) {
            char ch = message.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                message.setCharAt(i, key.charAt(ch - 'a'));
            }
    }
}

// Decrypt function
public static void decrypt(StringBuilder message, String key) {
    for (int i = 0; i < message.length(); i++) {
        char ch = message.charAt(i);
        if (ch >= 'a' && ch <= 'z') {
            for (int j = 0; j < 26; j++) {
                if (ch == key.charAt(j)) {
                    message.setCharAt(i, (char) ('a' + j));
                    break;
                }
            }
        }
    }
}

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

    // Input for the substitution key
    System.out.print("Enter the substitution key (26 lowercase letters in random order): ");
    String key = scanner.nextLine();

    if (key.length() != 26) {
        System.out.println("Invalid key length. Please provide 26 letters.");
        return;
    }

    // Validate key contains only lowercase letters
```

```
        for (int i = 0; i < 26; i++) {
            if (key.charAt(i) < 'a' || key.charAt(i) > 'z') {
                System.out.println("Invalid key. Please provide only lowercase letters.");
                return;
            }
        }

        // Input for the message to encrypt
        System.out.print("Enter the message to encrypt: ");
        String message = scanner.nextLine();

        // Convert the message to a mutable StringBuilder
        StringBuilder messageBuilder = new StringBuilder(message);

        // Encrypt the message
        encrypt(messageBuilder, key);
        System.out.println("Encrypted message: " + messageBuilder.toString());

        // Decrypt the message
        decrypt(messageBuilder, key);
        System.out.println("Decrypted message: " + messageBuilder.toString());

        scanner.close();
    }
}
```

PlayFair Cipher

```
import java.util.Scanner;

public class PlayfairCipher {
```

```
static final int SIZE = 5;

// Method to generate the key table
public static void generateKeyTable(String key, char[][] keyTable) {
    boolean[] dict = new boolean[26];
    int k = 0, l = 0;

    // Populate the keyTable with unique characters from the key
    for (int i = 0; i < key.length(); i++) {
        char ch = key.charAt(i);
        if (ch != 'j' && !dict[ch - 'a']) {
            keyTable[k][l] = ch;
            dict[ch - 'a'] = true;
            l++;
            if (l == SIZE) {
                l = 0;
                k++;
            }
        }
    }

    // Fill the remaining slots with other alphabets
    for (int i = 0; i < 26; i++) {
        if (!dict[i] && i != ('j' - 'a')) {
            keyTable[k][l] = (char) ('a' + i);
            l++;
            if (l == SIZE) {
                l = 0;
                k++;
            }
        }
    }
}

// Method to find the positions of two characters in the key table
public static void search(char[][] keyTable, char a, char b, int[] pos) {
    if (a == 'j') a = 'i';
    if (b == 'j') b = 'i';

    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (keyTable[i][j] == a) {
                pos[0] = i;
                pos[1] = j;
            } else if (keyTable[i][j] == b) {
                pos[2] = i;
                pos[3] = j;
            }
        }
    }
}

// Method to encrypt the message
public static String encrypt(String str, char[][] keyTable) {
    StringBuilder encrypted = new StringBuilder(str);

    for (int i = 0; i < encrypted.length(); i += 2) {
        int[] pos = new int[4];
        search(keyTable, encrypted.charAt(i), encrypted.charAt(i + 1), pos);

        if (pos[0] == pos[2]) { // Same row
            encrypted.setCharAt(i, keyTable[pos[0]][(pos[1] + 1) % SIZE]);
            encrypted.setCharAt(i + 1, keyTable[pos[2]][(pos[3] + 1) % SIZE]);
        } else if (pos[1] == pos[3]) { // Same column
            encrypted.setCharAt(i, keyTable[(pos[0] + 1) % SIZE][pos[1]]);
            encrypted.setCharAt(i + 1, keyTable[(pos[2] + 1) % SIZE][pos[3]]);
        } else { // Rectangle
            encrypted.setCharAt(i, keyTable[pos[0]][pos[3]]);
            encrypted.setCharAt(i + 1, keyTable[pos[2]][pos[1]]);
        }
    }

    return encrypted.toString();
}

// Method to decrypt the message
public static String decrypt(String str, char[][] keyTable) {
    StringBuilder decrypted = new StringBuilder(str);

    for (int i = 0; i < decrypted.length(); i += 2) {
        int[] pos = new int[4];
        search(keyTable, decrypted.charAt(i), decrypted.charAt(i + 1), pos);

        if (pos[0] == pos[2]) { // Same row
            decrypted.setCharAt(i, keyTable[pos[0]][(pos[1] + SIZE - 1) % SIZE]);
            decrypted.setCharAt(i + 1, keyTable[pos[2]][(pos[3] + SIZE - 1) % SIZE]);
        } else if (pos[1] == pos[3]) { // Same column
            decrypted.setCharAt(i, keyTable[(pos[0] + SIZE - 1) % SIZE][pos[1]]);
            decrypted.setCharAt(i + 1, keyTable[(pos[2] + SIZE - 1) % SIZE][pos[3]]);
        } else { // Rectangle
            decrypted.setCharAt(i, keyTable[pos[0]][pos[3]]);
            decrypted.setCharAt(i + 1, keyTable[pos[2]][pos[1]]);
        }
        }

        return decrypted.toString();
    }

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

        // Input key
        System.out.print("Enter the key (in lowercase, without 'j'): ");
        String key = scanner.nextLine().replace("j", "");

        // Generate the key table
        char[][] keyTable = new char[SIZE][SIZE];
        generateKeyTable(key, keyTable);

        // Input message
        System.out.print("Enter the message to encrypt/decrypt (in lowercase, without 'j'): ");
        String str = scanner.nextLine().replace("j", "");

        // Pad the message if necessary
        if (str.length() % 2 != 0) {
            str += 'x';
        }

        // Encrypt the message
        String encryptedMessage = encrypt(str, keyTable);
        System.out.println("Encrypted Message: " + encryptedMessage);

        // Decrypt the message
        String decryptedMessage = decrypt(encryptedMessage, keyTable);
        System.out.println("Decrypted Message: " + decryptedMessage);

        scanner.close();
    }
}
```

Vignere

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        String plaintext = "GEEKSFORGEEKS";
        String key = "AYUSHAYUSHAYU";

        // Convert key to lowercase
        key = key.toLowerCase();

        // Convert plaintext to lowercase
        plaintext = plaintext.toLowerCase();

        // Create hash table (2D list)
        List<List<Integer>> hash = new ArrayList<>();
        for (int i = 0; i < 26; i++) {
            List<Integer> row = new ArrayList<>();
            int startIndex = i;
            for (int j = 0; j < 26; j++) {
                row.add(((97 + j + startIndex) - 97) % 26);
            }
            hash.add(row);
        }
        int size = plaintext.length();
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < size; i++) {
            int a = plaintext.charAt(i) - 97;
            int b = key.charAt(i) - 97;
            result.append((char) (hash.get(a).get(b) + 97));
        }

        // Print the result
        System.out.println(result);
    }
}
```