Name: Shreeyash S. Dongarkar

PRN: 22510025

BTech Final Year CSE

Cryptography and Network Security Lab (B – 1)

Assignment 1: Encryption and Decryption Using Substitution Techniques

**Objectives**: To understand working of different classical substitution ciphers.

1) Caesar Cipher

1. import java.util.Scanner;

2.

3. public class CaesarCipher {

4.

5.     public static String encrypt(String text, int shift) {

6.         StringBuilder result = new StringBuilder();

7.         shift = shift % 26;

8.

9.         for (char c : text.toCharArray()) {

10.             if (Character.isUpperCase(c)) {

11.                 char ch = (char) (((c - 'A' + shift + 26) % 26) + 'A');

12.                 result.append(ch);

13.             } else if (Character.isLowerCase(c)) {

14.                 char ch = (char) (((c - 'a' + shift + 26) % 26) + 'a');

15.                 result.append(ch);

16.             } else {

17.                 result.append(c);

18.             }

19.         }

20.         return result.toString();

21.     }

22.

23.     public static String decrypt(String text, int shift) {

24.         return encrypt(text, -shift);

25.     }

26.

27.     public static void main(String[] args) {

28.         Scanner scanner = new Scanner(System.in);

29.

30.         System.out.println("Caesar Cipher");

31.         System.out.println("1. Encrypt");

32.         System.out.println("2. Decrypt");

33.         System.out.print("Choose an option (1 or 2): ");

34.         int choice = scanner.nextInt();

35.         scanner.nextLine();

36.

37.         System.out.print("Enter the text: ");

38.         String inputText = scanner.nextLine();

39.

40.         System.out.print("Enter the shift value (integer): ");

41.         int shift = scanner.nextInt();

42.

43.         String result;

44.         if (choice == 1) {

45.             result = encrypt(inputText, shift);

46.             System.out.println("Encrypted Text: " + result);

47.         } else if (choice == 2) {

48.             result = decrypt(inputText, shift);

49.             System.out.println("Decrypted Text: " + result);

50.         } else {

51.             System.out.println("Invalid choice. Please select 1 or 2.");

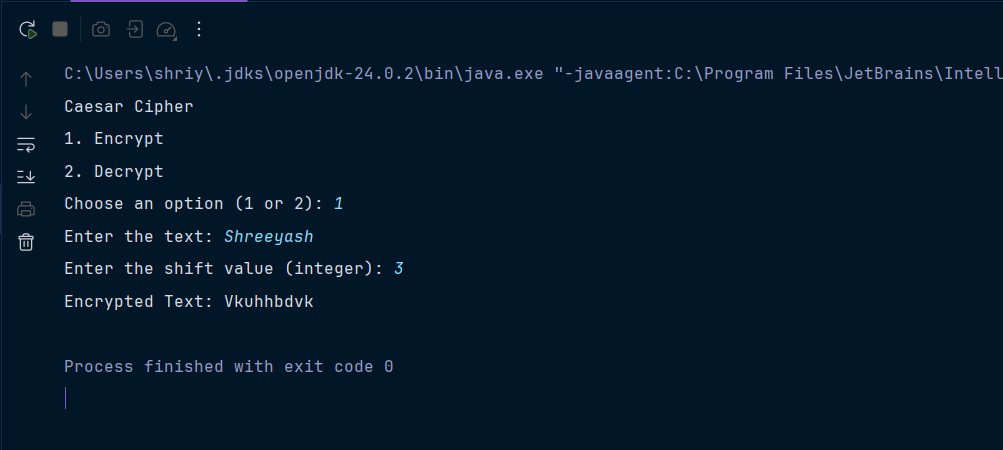
52.         }

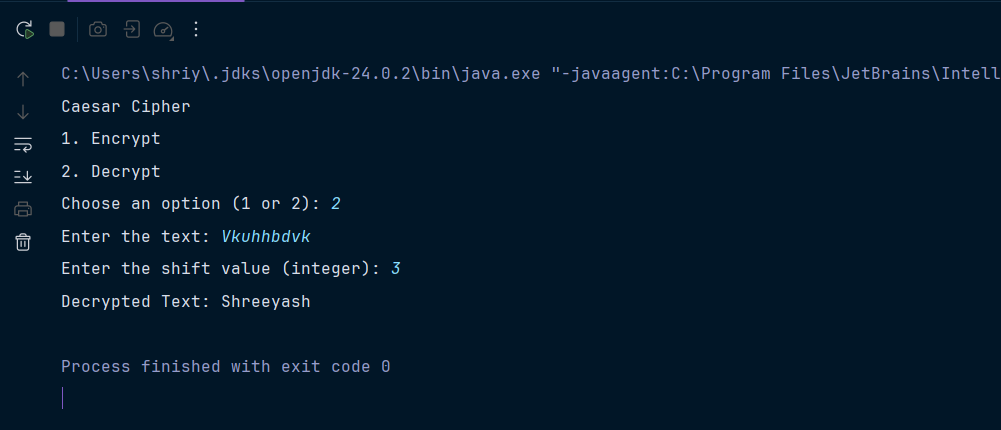
53.

54.         scanner.close();

55.     }

56. }





2) Playfair Cipher

1. import java.util.\*;

2.

3. public class PlayfairCipher {

4.

5.     private static final char[][] matrix = new char[5][5];

6.

7.     private static void generateMatrix(String key) {

8.         boolean[] used = new boolean[26];

9.         key = key.toUpperCase().replaceAll("[^A-Z]", "").replace("J", "I");

10.         StringBuilder sb = new StringBuilder();

11.

12.         for (char c : key.toCharArray()) {

13.             if (!used[c - 'A']) {

14.                 sb.append(c);

15.                 used[c - 'A'] = true;

16.             }

17.         }

18.

19.         for (char c = 'A'; c <= 'Z'; c++) {

20.             if (c == 'J') continue;

21.             if (!used[c - 'A']) {

22.                 sb.append(c);

23.                 used[c - 'A'] = true;

24.             }

25.         }

26.

27.         int k = 0;

28.         for (int i = 0; i < 5; i++) {

29.             for (int j = 0; j < 5; j++) {

30.                 matrix[i][j] = sb.charAt(k++);

31.             }

32.         }

33.     }

34.

35.     private static int[] getPosition(char c) {

36.         if (c == 'J') c = 'I';

37.         for (int i = 0; i < 5; i++)

38.             for (int j = 0; j < 5; j++)

39.                 if (matrix[i][j] == c)

40.                     return new int[]{i, j};

41.         return null;

42.     }

43.

44.     private static List<String> formPairs(String text) {

45.         text = text.toUpperCase().replaceAll("[^A-Z]", "").replace("J", "I");

46.         List<String> pairs = new ArrayList<>();

47.

48.         for (int i = 0; i < text.length(); i++) {

49.             char a = text.charAt(i);

50.             char b = (i + 1 < text.length()) ? text.charAt(i + 1) : 'X';

51.

52.             if (a == b) {

53.                 pairs.add("" + a + 'X');

54.             } else {

55.                 pairs.add("" + a + b);

56.                 i++; // skip next char

57.             }

58.         }

59.

60.         if (pairs.getLast().length() == 1) {

61.             pairs.set(pairs.size() - 1, pairs.getLast() + "X");

62.         }

63.

64.         return pairs;

65.     }

66.

67.     private static String encrypt(String plaintext, String key) {

68.         generateMatrix(key);

69.         List<String> pairs = formPairs(plaintext);

70.         StringBuilder cipher = new StringBuilder();

71.

72.         for (String pair : pairs) {

73.             char a = pair.charAt(0);

74.             char b = pair.charAt(1);

75.             int[] pos1 = getPosition(a);

76.             int[] pos2 = getPosition(b);

77.

78.             assert pos1 != null;

79.             assert pos2 != null;

80.             if (pos1[0] == pos2[0]) {

81.                 cipher.append(matrix[pos1[0]][(pos1[1] + 1) % 5]);

82.                 cipher.append(matrix[pos2[0]][(pos2[1] + 1) % 5]);

83.             } else if (pos1[1] == pos2[1]) {

84.                 cipher.append(matrix[(pos1[0] + 1) % 5][pos1[1]]);

85.                 cipher.append(matrix[(pos2[0] + 1) % 5][pos2[1]]);

86.             } else {

87.                 cipher.append(matrix[pos1[0]][pos2[1]]);

88.                 cipher.append(matrix[pos2[0]][pos1[1]]);

89.             }

90.         }

91.

92.         return cipher.toString();

93.     }

94.

95.     private static String decrypt(String ciphertext, String key) {

96.         generateMatrix(key);

97.         List<String> pairs = formPairs(ciphertext);

98.         StringBuilder plain = new StringBuilder();

99.

100.         for (String pair : pairs) {

101.             char a = pair.charAt(0);

102.             char b = pair.charAt(1);

103.             int[] pos1 = getPosition(a);

104.             int[] pos2 = getPosition(b);

105.

106.             assert pos1 != null;

107.             assert pos2 != null;

108.             if (pos1[0] == pos2[0]) {

109.                 plain.append(matrix[pos1[0]][(pos1[1] + 4) % 5]);

110.                 plain.append(matrix[pos2[0]][(pos2[1] + 4) % 5]);

111.             } else if (pos1[1] == pos2[1]) {

112.                 plain.append(matrix[(pos1[0] + 4) % 5][pos1[1]]);

113.                 plain.append(matrix[(pos2[0] + 4) % 5][pos2[1]]);

114.             } else {

115.                 plain.append(matrix[pos1[0]][pos2[1]]);

116.                 plain.append(matrix[pos2[0]][pos1[1]]);

117.             }

118.         }

119.

120.         return plain.toString();

121.     }

122.

123.     public static void main(String[] args) {

124.         Scanner scanner = new Scanner(System.in);

125.

126.         System.out.println("Playfair Cipher");

127.         System.out.println("1. Encrypt");

128.         System.out.println("2. Decrypt");

129.         System.out.print("Choose an option (1 or 2): ");

130.         int choice = scanner.nextInt();

131.         scanner.nextLine();

132.

133.         System.out.print("Enter the key: ");

134.         String key = scanner.nextLine();

135.

136.         System.out.print("Enter the text: ");

137.         String text = scanner.nextLine();

138.

139.         String result;

140.         if (choice == 1) {

141.             result = encrypt(text, key);

142.             System.out.println("Encrypted Text: " + result);

143.         } else if (choice == 2) {

144.             result = decrypt(text, key);

145.             System.out.println("Decrypted Text: " + result);

146.         } else {

147.             System.out.println("Invalid choice.");

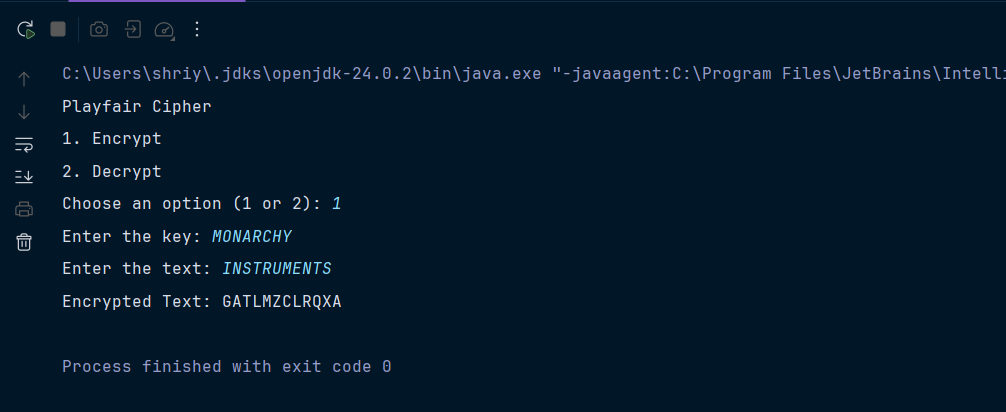
148.         }

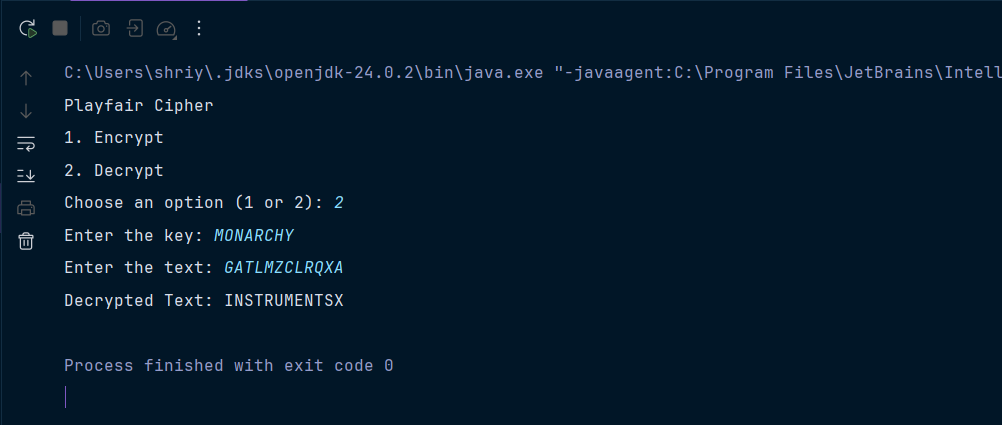
149.

150.         scanner.close();

151.     }

152. }





3) Hill Cipher

1. import java.util.\*;

2.

3. public class HillCipher {

4.

5.     private static int charToInt(char c) {

6.         return c - 'A';

7.     }

8.

9.     private static char intToChar(int n) {

10.         return (char) (n + 'A');

11.     }

12.

13.     private static int[] multiplyMatrix(int[][] key, int[] vector) {

14.         int[] result = new int[2];

15.         for (int i = 0; i < 2; i++) {

16.             result[i] = (key[i][0] \* vector[0] + key[i][1] \* vector[1]) % 26;

17.             if (result[i] < 0) result[i] += 26;

18.         }

19.         return result;

20.     }

21.

22.     private static int modInverse(int a, int m) {

23.         a = a % m;

24.         for (int x = 1; x < m; x++) {

25.             if ((a \* x) % m == 1) return x;

26.         }

27.         return -1;

28.     }

29.

30.     private static int[][] inverseMatrix(int[][] key) {

31.         int det = key[0][0] \* key[1][1] - key[0][1] \* key[1][0];

32.         det = ((det % 26) + 26) % 26;

33.         int invDet = modInverse(det, 26);

34.         if (invDet == -1) {

35.             throw new IllegalArgumentException("Key matrix is not invertible mod 26.");

36.         }

37.

38.         int[][] invKey = new int[2][2];

39.         invKey[0][0] = (key[1][1] \* invDet) % 26;

40.         invKey[0][1] = (-key[0][1] \* invDet) % 26;

41.         invKey[1][0] = (-key[1][0] \* invDet) % 26;

42.         invKey[1][1] = (key[0][0] \* invDet) % 26;

43.

44.         // Ensure positive values

45.         for (int i = 0; i < 2; i++)

46.             for (int j = 0; j < 2; j++)

47.                 invKey[i][j] = (invKey[i][j] + 26) % 26;

48.

49.         return invKey;

50.     }

51.

52.     private static String encrypt(String plaintext, int[][] key) {

53.         plaintext = plaintext.toUpperCase().replaceAll("[^A-Z]", "");

54.         if (plaintext.length() % 2 != 0) plaintext += "X";

55.

56.         StringBuilder cipher = new StringBuilder();

57.         for (int i = 0; i < plaintext.length(); i += 2) {

58.             int[] vector = {charToInt(plaintext.charAt(i)), charToInt(plaintext.charAt(i + 1))};

59.             int[] result = multiplyMatrix(key, vector);

60.             cipher.append(intToChar(result[0])).append(intToChar(result[1]));

61.         }

62.         return cipher.toString();

63.     }

64.

65.     private static String decrypt(String ciphertext, int[][] key) {

66.         int[][] invKey = inverseMatrix(key);

67.         ciphertext = ciphertext.toUpperCase().replaceAll("[^A-Z]", "");

68.

69.         StringBuilder plain = new StringBuilder();

70.         for (int i = 0; i < ciphertext.length(); i += 2) {

71.             int[] vector = {charToInt(ciphertext.charAt(i)), charToInt(ciphertext.charAt(i + 1))};

72.             int[] result = multiplyMatrix(invKey, vector);

73.             plain.append(intToChar(result[0])).append(intToChar(result[1]));

74.         }

75.         return plain.toString();

76.     }

77.

78.     public static void main(String[] args) {

79.         Scanner sc = new Scanner(System.in);

80.

81.         System.out.println("Hill Cipher (2x2 Matrix)");

82.         System.out.println("1. Encrypt");

83.         System.out.println("2. Decrypt");

84.         System.out.print("Choose an option (1 or 2): ");

85.         int choice = sc.nextInt();

86.         sc.nextLine();

87.

88.         int[][] key = new int[2][2];

89.         System.out.println("Enter 2x2 key matrix (integers only, space-separated):");

90.         for (int i = 0; i < 2; i++)

91.             for (int j = 0; j < 2; j++)

92.                 key[i][j] = sc.nextInt();

93.         sc.nextLine();

94.

95.         System.out.print("Enter the text: ");

96.         String text = sc.nextLine();

97.

98.         String result;

99.         if (choice == 1) {

100.             result = encrypt(text, key);

101.             System.out.println("Encrypted Text: " + result);

102.         } else if (choice == 2) {

103.             try {

104.                 result = decrypt(text, key);

105.                 System.out.println("Decrypted Text: " + result);

106.             } catch (IllegalArgumentException e) {

107.                 System.out.println("Error: " + e.getMessage());

108.             }

109.         } else {

110.             System.out.println("Invalid choice.");

111.         }

112.

113.         sc.close();

114.     }

115. }





4) Vigenère Cipher

1. import java.util.Scanner;

2.

3. public class VigenereCipher {

4.

5.     private static String encrypt(String text, String key) {

6.         StringBuilder result = new StringBuilder();

7.         text = text.toUpperCase().replaceAll("[^A-Z]", "");

8.         key = key.toUpperCase();

9.

10.         for (int i = 0, j = 0; i < text.length(); i++) {

11.             char c = text.charAt(i);

12.             int shift = key.charAt(j) - 'A';

13.             char encryptedChar = (char) ((c - 'A' + shift) % 26 + 'A');

14.             result.append(encryptedChar);

15.             j = (j + 1) % key.length();

16.         }

17.         return result.toString();

18.     }

19.

20.     private static String decrypt(String text, String key) {

21.         StringBuilder result = new StringBuilder();

22.         text = text.toUpperCase().replaceAll("[^A-Z]", "");

23.         key = key.toUpperCase();

24.

25.         for (int i = 0, j = 0; i < text.length(); i++) {

26.             char c = text.charAt(i);

27.             int shift = key.charAt(j) - 'A';

28.             char decryptedChar = (char) ((c - 'A' - shift + 26) % 26 + 'A');

29.             result.append(decryptedChar);

30.             j = (j + 1) % key.length();

31.         }

32.         return result.toString();

33.     }

34.

35.     public static void main(String[] args) {

36.         Scanner sc = new Scanner(System.in);

37.

38.         System.out.println("Vigenère Cipher");

39.         System.out.println("1. Encrypt");

40.         System.out.println("2. Decrypt");

41.         System.out.print("Choose an option (1 or 2): ");

42.         int choice = sc.nextInt();

43.         sc.nextLine();

44.

45.         System.out.print("Enter the key: ");

46.         String key = sc.nextLine();

47.

48.         System.out.print("Enter the text: ");

49.         String text = sc.nextLine();

50.

51.         String result;

52.         if (choice == 1) {

53.             result = encrypt(text, key);

54.             System.out.println("Encrypted Text: " + result);

55.         } else if (choice == 2) {

56.             result = decrypt(text, key);

57.             System.out.println("Decrypted Text: " + result);

58.         } else {

59.             System.out.println("Invalid choice.");

60.         }

61.

62.         sc.close();

63.     }

64. }

