14. public class PlayfairCipherDecoder {

public static void main(String[] args) {

String ciphertext = "KXJEY UREBE ZWEHE WRYTU HEYFS\n" +

"KREHE GOYFI WTTTU OLKSY CAJPO\n" +

"BOTEI ZONTX BYBNT GONEY CUZWR\n" +

"GDSON SXBOU YWRHE BAAHY USEDQ";

// Remove all whitespace and convert to uppercase

String plaintext = ciphertext.replaceAll("\\s+", "").toUpperCase();

// Generate the Playfair matrix

char[][] matrix = generateMatrix("JOHNFKENNEDY");

// Decode the ciphertext using the Playfair algorithm

StringBuilder sb = new StringBuilder();

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

char c1 = plaintext.charAt(i);

char c2 = plaintext.charAt(i + 1);

int[] pos1 = findPosition(matrix, c1);

int[] pos2 = findPosition(matrix, c2);

if (pos1[0] == pos2[0]) { // same row

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

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

} else if (pos1[1] == pos2[1]) { // same column

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

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

} else { // different row and column

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

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

}

}

// Print the decoded message

System.out.println(sb.toString());

}

// Generate the Playfair matrix from the keyword

public static char[][] generateMatrix(String keyword) {

String key = keyword.toUpperCase().replaceAll("[^A-Z]", "");

key = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";

char[] keyArray = key.toCharArray();

char[][] matrix = new char[5][5];

int index = 0;

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

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

matrix[i][j] = keyArray[index];

index++;

}

}

return matrix;

}

// Find the position of a character in the Playfair matrix

public static int[] findPosition(char[][] matrix, char c) {

int[] pos = new int[2];

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

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

if (matrix[i][j] == c) {

pos[0] = i;

pos[1] = j;

break;

}

}

}

return pos;

}

}

Output:

IOODDOODYGFFDGDOODFFGDAEOIODGDENAEDIFFFFHKIODBDHFJYNODDIFKFFADCMNDFKYEDFFFNDDIFKFFYNODFFGDAYCFDOODDI