**Ex.No:1a Encryption and Decryption Using Ceaser Cipher**

**Date:**

**Aim:**

To encrypt and decrypt the given message by using  Ceaser Cipher encryption algorithm.

**Algorithm:**

1. In Ceaser Cipher each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.
2. For example, with a **left shift of 3**, **D** would be replaced by **A**, **E** would become **B**, and so on.
3. The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, **A = 0, B = 1, Z = 25.**
4. Encryption of a letter x by a shift n can be described mathematically as, ***En(x) = (x + n) mod26***
5. Decryption is performed similarly, ***Dn (x)=(x - n) mod26***

**Program:**

public class CeaserCipher {

public static String encode(String enc, int offset) {

offset = offset % 26 + 26;

StringBuilder encoded = new StringBuilder();

for (char i : enc.toCharArray()) {

if (Character.isLetter(i)) {

if (Character.isUpperCase(i)) {

encoded.append((char) ('A' + (i - 'A' + offset) % 26));

} else {

encoded.append((char) ('a' + (i - 'a' + offset) % 26));

}

} else {

encoded.append(i);

}

}

returnencoded.toString();

}

public static String decode(String enc, int offset) {

return encode(enc, 26 - offset);

}

public static void main(String[] args) throws java.lang.Exception {

String msg = "Velammal College";

System.out.println("Simulating Caesar Cipher\n------------------------");

System.out.println("Input :" + msg);

System.out.printf("Encrypted Message : ");

System.out.println(CeaserCipher.encode(msg, 3));

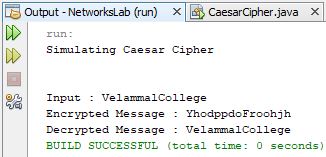
System.out.printf("Decrypted Message : ");

System.out.println(CeaserCipher.decode(CeaserCipher.encode(msg, 3), 3));

}

}

**Output:**



**Result:**

Thus the program for CeaserCipher encryption and decryption algorithm has been implemented and the output verified successfully.

**Ex.No:1b Encryption and Decryption Using Playfair Cipher**

**Date:**

**Aim:**

To implement a program to encrypt a plain text and decrypt a cipher text using play fair Cipher substitution technique.

**Algorithm:**

1. To encrypt a message, one would break the message into digrams (groups of 2 letters)
2. For example, "HelloWorld" becomes "HE LL OW OR LD".
3. These digrams will be substituted using the key table.
4. Since encryption requires pairs of letters, messages with an odd number of characters usually append an uncommon letter, such as "X", to complete the final digram.
5. The two letters of the digram are considered opposite corners of a rectangle in the key table. To perform the substitution, apply the following 4 rules, in order, to each pair of letters in the plaintext

**Program:**

importjava.awt.Point;

classplayfairCipher {

private static char[][] charTable;

private static Point[] positions;

private static String prepareText(String s, booleanchgJtoI) {

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

returnchgJtoI ? s.replace("J","I") : s.replace("Q", "");

}

private static void createTbl(String key, booleanchgJtoI) {

charTable = new char[5][5];

positions = new Point[26];

String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ",

chgJtoI);

intlen = s.length();

for (inti = 0, k = 0; i<len; i++) {

char c = s.charAt(i);

if (positions[c - 'A'] == null) {

charTable[k / 5][k % 5] = c;

positions[c - 'A'] = new Point(k % 5, k / 5);

k++;

}

}

}

private static String codec(StringBuilder txt, intdir) {

intlen = txt.length();

for (inti = 0; i<len; i += 2) {

char a = txt.charAt(i);

char b = txt.charAt(i + 1);

int row1 = positions[a - 'A'].y;

int row2 = positions[b - 'A'].y;

int col1 = positions[a - 'A'].x;

int col2 = positions[b - 'A'].x;

if (row1 == row2) {

col1 = (col1 + dir) % 5;

col2 = (col2 + dir) % 5;

} else if (col1 == col2) {

row1 = (row1 + dir) % 5;

row2 = (row2 + dir) % 5;

} else {

inttmp = col1;

col1 = col2;

col2 = tmp;

}

txt.setCharAt(i, charTable[row1][col1]);

txt.setCharAt(i + 1, charTable[row2][col2]);

}

returntxt.toString();

}

private static String encode(String s) {

StringBuildersb = new StringBuilder(s);

for (inti = 0; i<sb.length(); i += 2) {

if (i == sb.length() - 1) {

sb.append(sb.length() % 2 == 1 ? 'X': "");

} else if (sb.charAt(i) == sb.charAt(i + 1)) {

sb.insert(i + 1, 'X');

}

}

return codec(sb, 1);

}

private static String decode(String s) {

return codec(new StringBuilder(s), 4);

}

public static void main(String[] args) throws java.lang.Exception {

String key = "CSE";

String txt = "Engineering Lab"; /\* make sure string length is even \*/ /\* change J

to I \*/

booleanchgJtoI = true;

createTbl(key, chgJtoI);

String enc = encode(prepareText(txt, chgJtoI));

System.out.println("Simulating Playfair Cipher\n----------------------");

System.out.println("Input Message : " + txt);

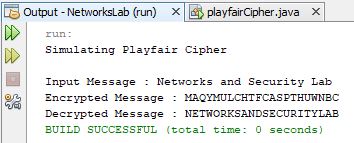
System.out.println("Encrypted Message : " + enc);

System.out.println("Decrypted Message : " + decode(enc));

}

}

**Output:**



**Result:**

Thus the program for playfair cipher encryption and decryption algorithm has been implemented and the output verified successfully.

**Ex.No:1c Encryption and Decryption Using Hill Cipher**

**Date:**

**Aim:**

To implement a program to encrypt and decrypt using the Hill cipher substitution technique

**Algorithm:**

1. In the Hill cipher Each letter is represented by a number modulo 26.
2. To encrypt a message, each block of n letters is multiplied by an invertible ***n x n*** matrix, again ***modulus 26***.
3. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.
4. The matrix used for encryption is the cipher key, and it should be chosen randomly from the ***set of invertible n × n matrices (modulo 26).***
5. The cipher can, be adapted to an alphabet with any number of letters.
6. All arithmetic just needs to be done modulo the number of letters instead of modulo 26.

**Program:**

classhillCipher {

/\* 3x3 key matrix for 3 characters at once \*/

public static int[][] keymat = new int[][] { { 1, 2, 1 }, { 2, 3, 2 },

{ 2, 2, 1 } }; /\* key inverse matrix \*/

public static int[][] invkeymat = new int[][] { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1 } };

public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

private static String encode(char a, char b, char c) {

String ret = "";

int x, y, z;

intposa = (int) a - 65;

intposb = (int) b - 65;

intposc = (int) c - 65;

x = posa \* keymat[0][0] + posb \* keymat[1][0] + posc \* keymat[2][0];

y = posa \* keymat[0][1] + posb \* keymat[1][1] + posc \* keymat[2][1];

z = posa \* keymat[0][2] + posb \* keymat[1][2] + posc \* keymat[2][2];

a = key.charAt(x % 26);

b = key.charAt(y % 26);

c = key.charAt(z % 26);

ret = "" + a + b + c;

return ret;

}

private static String decode(char a, char b, char c) {

String ret = "";

int x, y, z;

intposa = (int) a - 65;

intposb = (int) b - 65;

intposc = (int) c - 65;

x = posa \* invkeymat[0][0] + posb \* invkeymat[1][0] + posc \* invkeymat[2][0];

y = posa \* invkeymat[0][1] + posb \* invkeymat[1][1] + posc \* invkeymat[2][1];

z = posa \* invkeymat[0][2] + posb \* invkeymat[1][2] + posc \* invkeymat[2][2];

a = key.charAt((x % 26 <0) ? (26 + x % 26) : (x % 26));

b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));

c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));

ret = "" + a + b + c;

return ret;

}

public static void main(String[] args) throws java.lang.Exception {

String msg;

String enc = "";

String dec = "";

int n;

msg = ("FinalyearLaboratory");

System.out.println("simulation of Hill Cipher\n-------------------------");

System.out.println("Input message : " + msg);

msg = msg.toUpperCase();

msg = msg.replaceAll("\\s", "");

/\* remove spaces \*/ n = msg.length() % 3;

/\* append padding text X \*/ if (n != 0) {

for (inti = 1; i<= (3 - n); i++) {

msg += 'X';

}

}

System.out.println("padded message : " + msg);

char[] pdchars = msg.toCharArray();

for (inti = 0; i<msg.length(); i += 3) {

enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);

}

System.out.println("encoded message : " + enc);

char[] dechars = enc.toCharArray();

for (inti = 0; i<enc.length(); i += 3) {

dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);

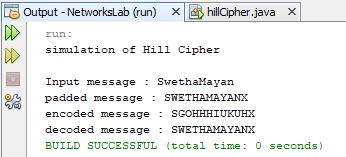
}

System.out.println("decoded message : " + dec);

}

}

**Output:**



**Result:**

Thus the program for hill cipher encryption and decryption algorithm has been implemented and the output verified successfully.

**Ex.No:1d Encryption and Decryption Using Vigenere Cipher**

**Date:**

**Aim:**

To implement a program for encryption and decryption using vigenere cipher substitution technique

**Algorithm:**

1. The Vigenere cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword.
2. It is a simple form of *polyalphabetic* substitution.
3. To encrypt, a table of alphabets can be used, termed a Vigenere square, or Vigenere table.
4. It consists of the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers.
5. At different points in the encryption process, the cipher uses a different alphabet from one of the rows used.
6. The alphabet at each point depends on a repeating keyword.

**Program:**

public class vigenereCipher {

static String encode(String text, final String key) {

String res = "";

text = text.toUpperCase();

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

char c = text.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

res += (char) ((c + key.charAt(j) - 2 \* 'A') % 26 + 'A');

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

}

return res;

}

static String decode(String text, final String key) {

String res ="";

text = text.toUpperCase();

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

char c = text.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');

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

}

return res;

}

public static void main(String[] args) throws java.lang.Exception {

String key = "VIGENERECIPHER";

String msg = "ComputerLaboratory";

System.out.println("Simulating Vigenere Cipher\n------------------------");

System.out.println("Input Message : " + msg);

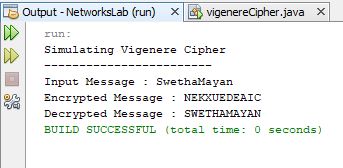
String enc = encode(msg, key);

System.out.println("Encrypted Message : " + enc);

System.out.println("Decrypted Message : " + decode(enc, key));

}}

**Output:**



**Result:**

Thus the program for vigenere cipher encryption and decryption algorithm has been implemented and the output verified successfully.