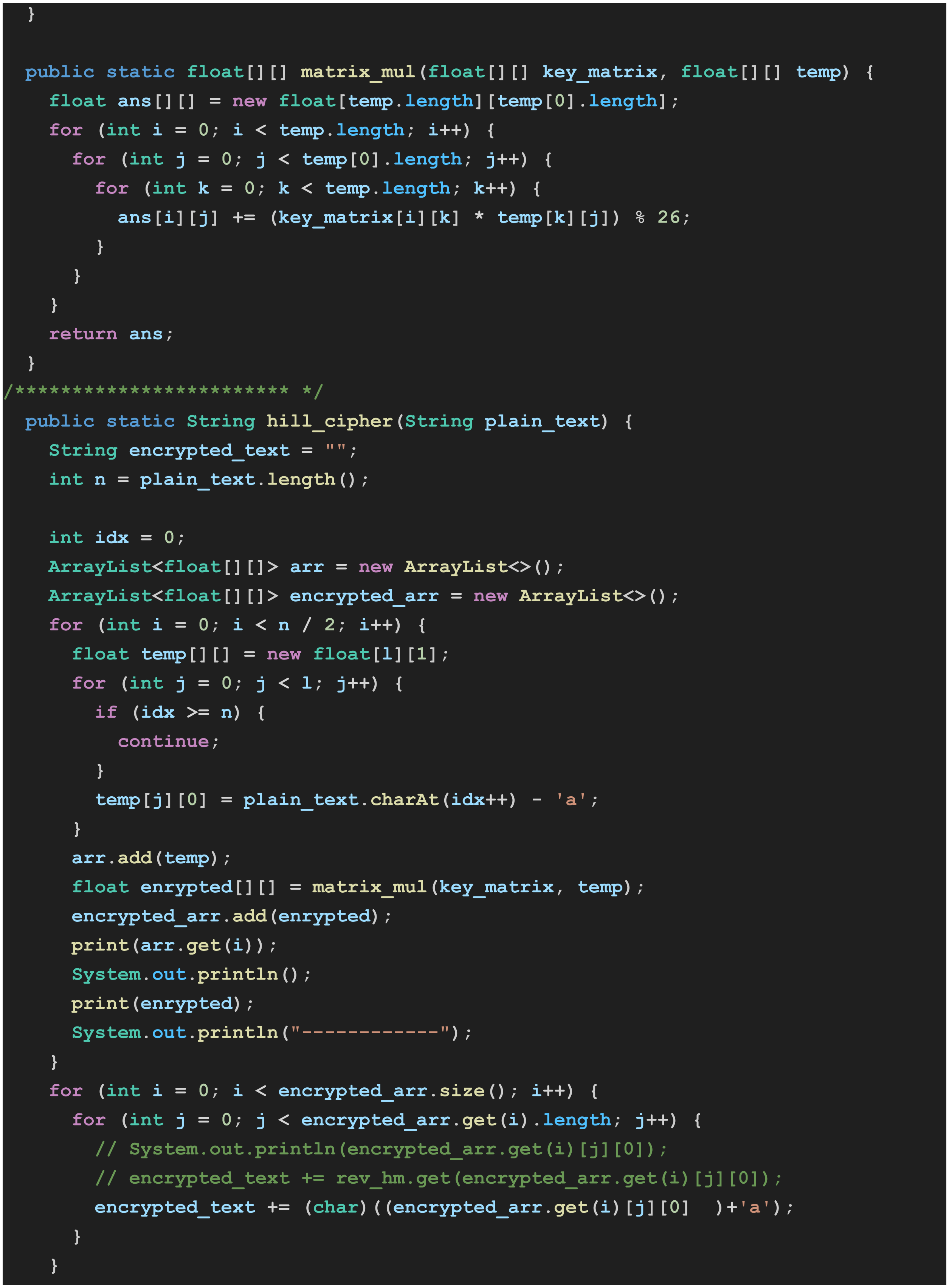
ISC ASSIGNMENT -05

ROLL NO: U21CS052
NAME : PANCHAL GUNGUN PARESH

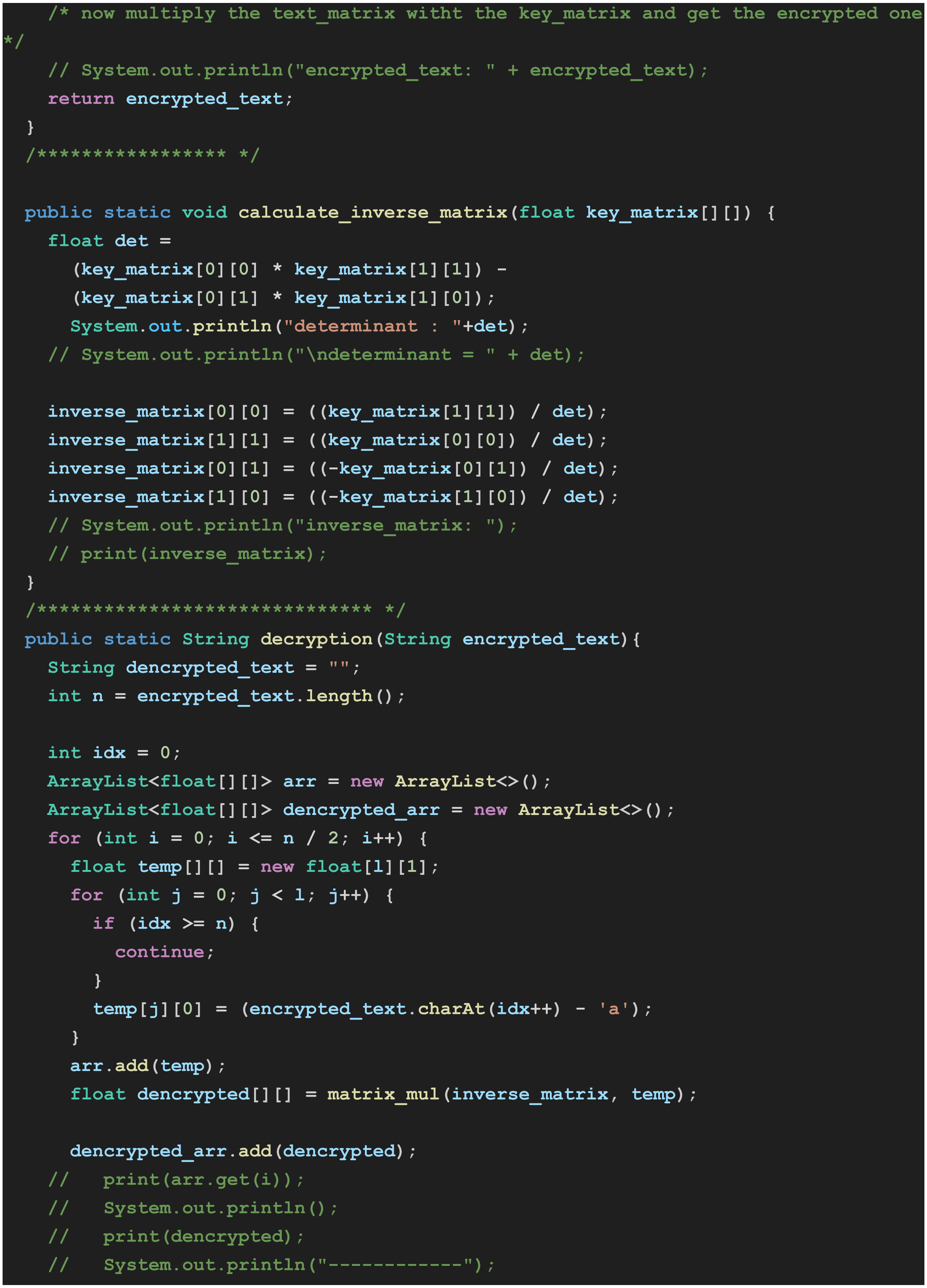
HILL CIPHER

QUE1. Design and implement a program to perform encryption and decryption using the Hill Cipher with 2x2 matrix.
Consider the following inputs for the program.
For Encryption :
a) Input1 – Plaintext, Key matrix, and Inverse key matrix
b) Output1 – Ciphertext
For Decryption :
a) Input2- Ciphertext
b) Output2- Plaintext

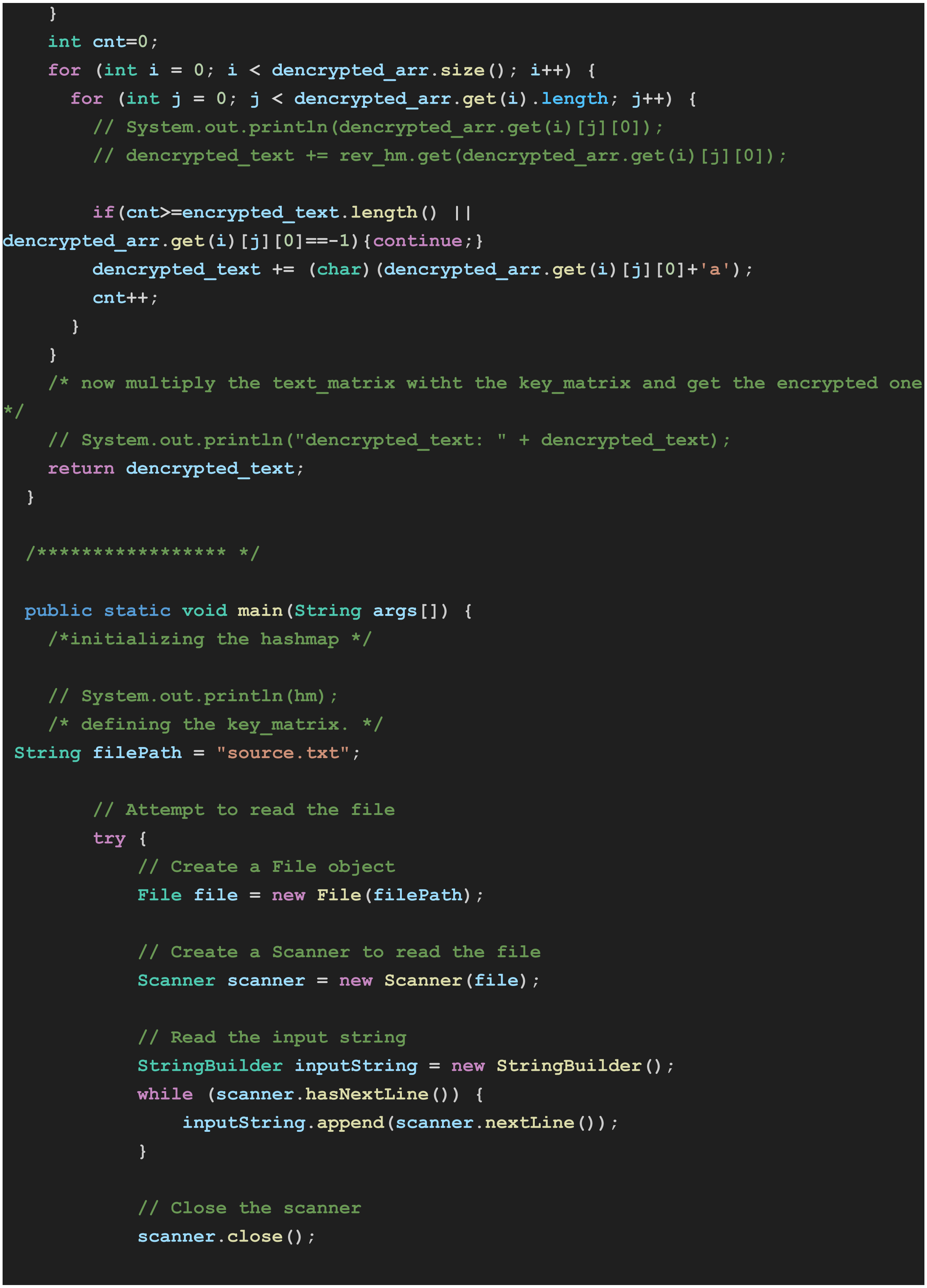
```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class que1_2x2 {

  public static int l = 2;
  public static float key_matrix[][] = { { 3, 0 }, { 0, 2 } };
  public static float inverse_matrix[][] = new
float[key_matrix.length][key_matrix[0].length];



  public static void print(float text_matrix[][]) {
    for (int i = 0; i < text_matrix.length; i++) {
      for (int j = 0; j < text_matrix[0].length; j++) {
        System.out.print(text_matrix[i][j] + " ");
      }
      System.out.println();
    }
```



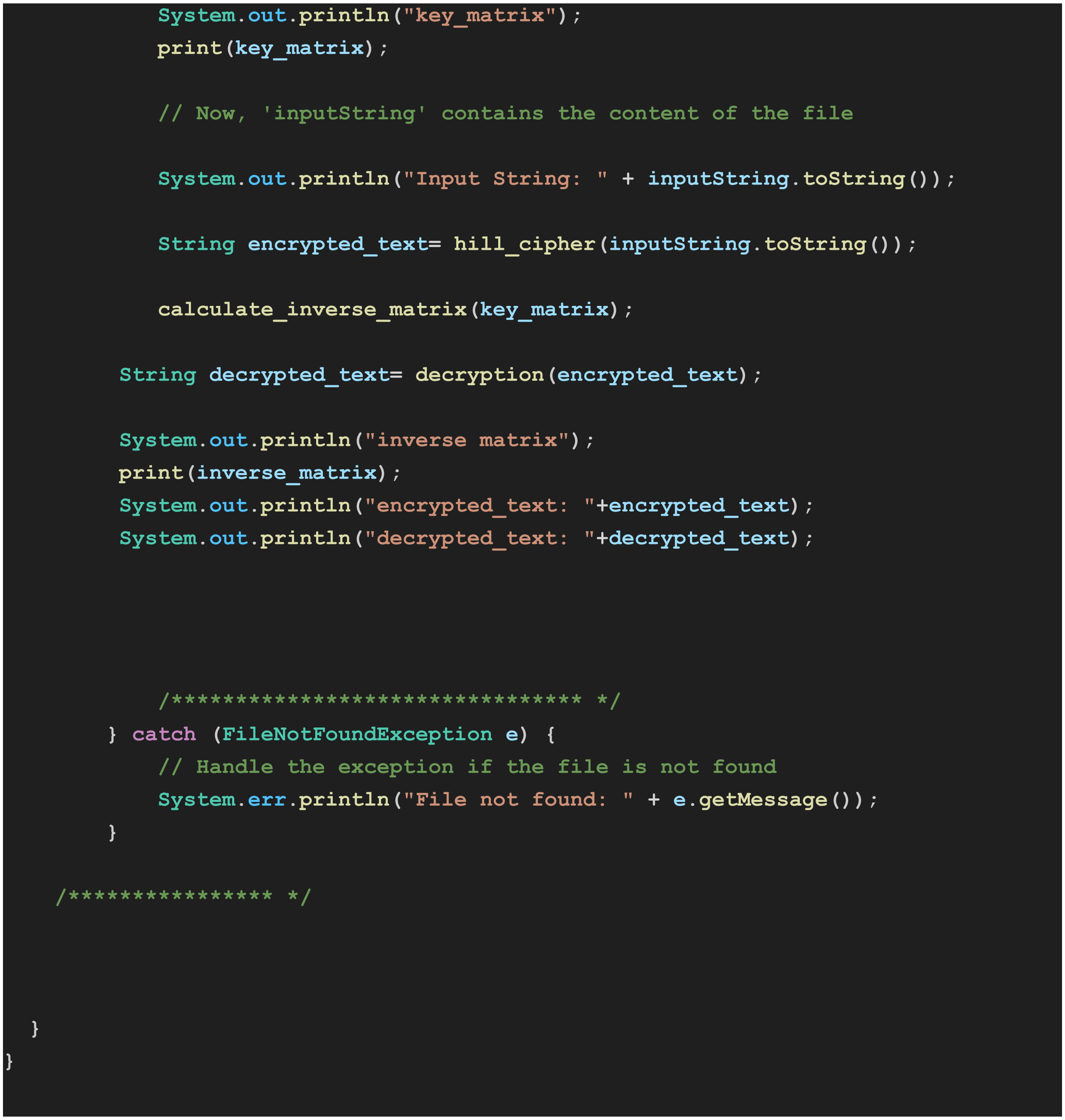
```
  }

  public static float[][] matrix_mul(float[][] key_matrix, float[][] temp) {
    float ans[][] = new float[temp.length][temp[0].length];
    for (int i = 0; i < temp.length; i++) {
      for (int j = 0; j < temp[0].length; j++) {
        for (int k = 0; k < temp.length; k++) {
          ans[i][j] += (key_matrix[i][k] * temp[k][j]) % 26;
        }
      }
    }
    return ans;
  }
/************************ */
  public static String hill_cipher(String plain_text) {
    String encrypted_text = "";
    int n = plain_text.length();

    int idx = 0;
    ArrayList<float[][]> arr = new ArrayList<>();
    ArrayList<float[][]> encrypted_arr = new ArrayList<>();
    for (int i = 0; i < n / 2; i++) {
      float temp[][] = new float[l][1];
      for (int j = 0; j < l; j++) {
        if (idx >= n) {
          continue;
        }
        temp[j][0] = plain_text.charAt(idx++) - 'a';
      }
      arr.add(temp);
      float enrypted[][] = matrix_mul(key_matrix, temp);
      encrypted_arr.add(enrypted);
      print(arr.get(i));
      System.out.println();
      print(enrypted);
      System.out.println("------------");
    }
    for (int i = 0; i < encrypted_arr.size(); i++) {
      for (int j = 0; j < encrypted_arr.get(i).length; j++) {
        // System.out.println(encrypted_arr.get(i)[j][0]);
        // encrypted_text += rev_hm.get(encrypted_arr.get(i)[j][0]);
        encrypted_text += (char)((encrypted_arr.get(i)[j][0]  )+'a');
      }
    }
```



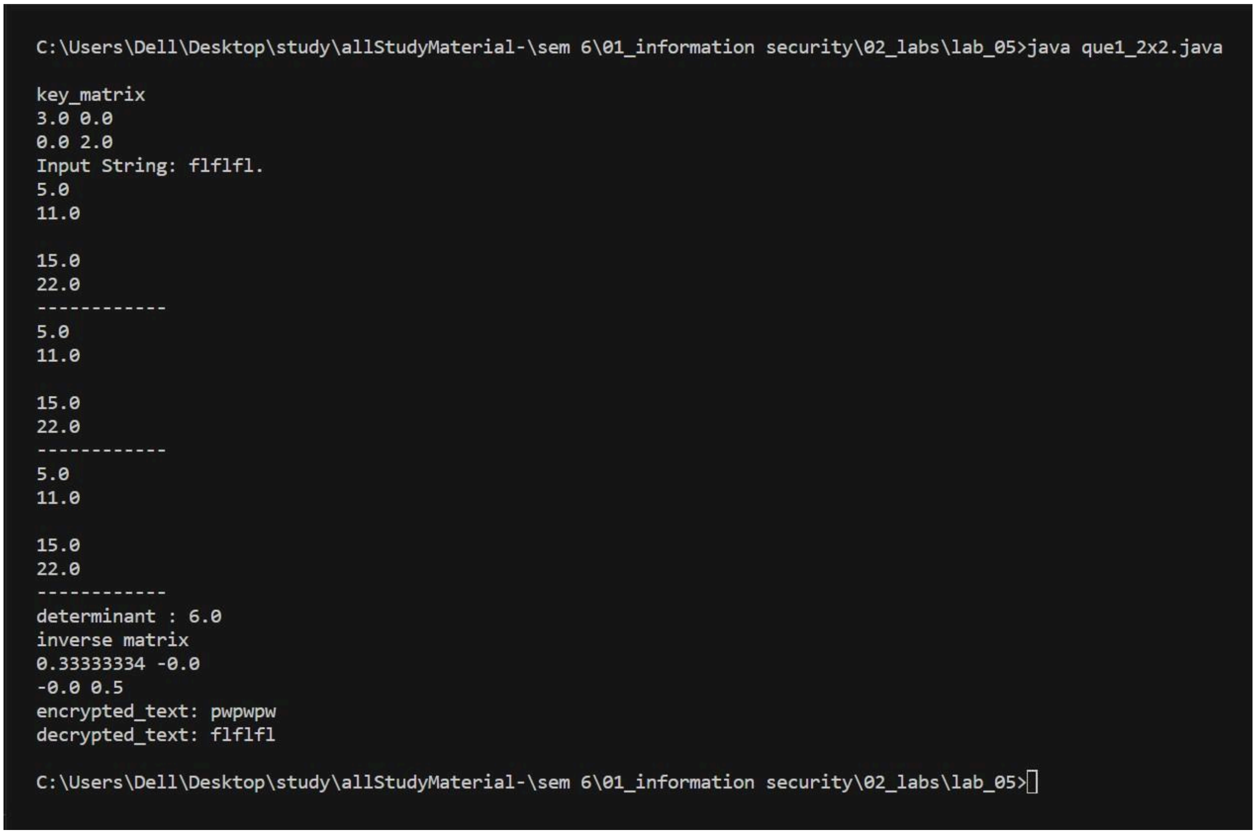
```
    /* now multiply the text_matrix witht the key_matrix and get the encrypted one
*/
    // System.out.println("encrypted_text: " + encrypted_text);
    return encrypted_text;
  }
  /***************** */

  public static void calculate_inverse_matrix(float key_matrix[][]) {
    float det =
      (key_matrix[0][0] * key_matrix[1][1]) -
      (key_matrix[0][1] * key_matrix[1][0]);
      System.out.println("determinant : "+det);
    // System.out.println("\ndeterminant = " + det);

    inverse_matrix[0][0] = ((key_matrix[1][1]) / det);
    inverse_matrix[1][1] = ((key_matrix[0][0]) / det);
    inverse_matrix[0][1] = ((-key_matrix[0][1]) / det);
    inverse_matrix[1][0] = ((-key_matrix[1][0]) / det);
    // System.out.println("inverse_matrix: ");
    // print(inverse_matrix);
  }
  /******************************* */
  public static String decryption(String encrypted_text){
    String dencrypted_text = "";
    int n = encrypted_text.length();

    int idx = 0;
    ArrayList<float[][]> arr = new ArrayList<>();
    ArrayList<float[][]> dencrypted_arr = new ArrayList<>();
    for (int i = 0; i <= n / 2; i++) {
      float temp[][] = new float[l][1];
      for (int j = 0; j < l; j++) {
        if (idx >= n) {
          continue;
        }
        temp[j][0] = (encrypted_text.charAt(idx++) - 'a');
      }
      arr.add(temp);
      float dencrypted[][] = matrix_mul(inverse_matrix, temp);

      dencrypted_arr.add(dencrypted);
    //   print(arr.get(i));
    //   System.out.println();
    //   print(dencrypted);
    //   System.out.println("------------");
```



```
    }
    int cnt=0;
    for (int i = 0; i < dencrypted_arr.size(); i++) {
      for (int j = 0; j < dencrypted_arr.get(i).length; j++) {
        // System.out.println(dencrypted_arr.get(i)[j][0]);
        // dencrypted_text += rev_hm.get(dencrypted_arr.get(i)[j][0]);

        if(cnt>=encrypted_text.length() ||
dencrypted_arr.get(i)[j][0]==-1){continue;}
        dencrypted_text += (char)(dencrypted_arr.get(i)[j][0]+'a');
        cnt++;
      }
    }
    /* now multiply the text_matrix witht the key_matrix and get the encrypted one
*/
    // System.out.println("dencrypted_text: " + dencrypted_text);
    return dencrypted_text;
  }

  /***************** */

  public static void main(String args[]) {
    /*initializing the hashmap */

    // System.out.println(hm);
    /* defining the key_matrix. */
 String filePath = "source.txt";

        // Attempt to read the file
        try {
            // Create a File object
            File file = new File(filePath);

            // Create a Scanner to read the file
            Scanner scanner = new Scanner(file);

            // Read the input string
            StringBuilder inputString = new StringBuilder();
            while (scanner.hasNextLine()) {
                inputString.append(scanner.nextLine());
            }

            // Close the scanner
            scanner.close();
```

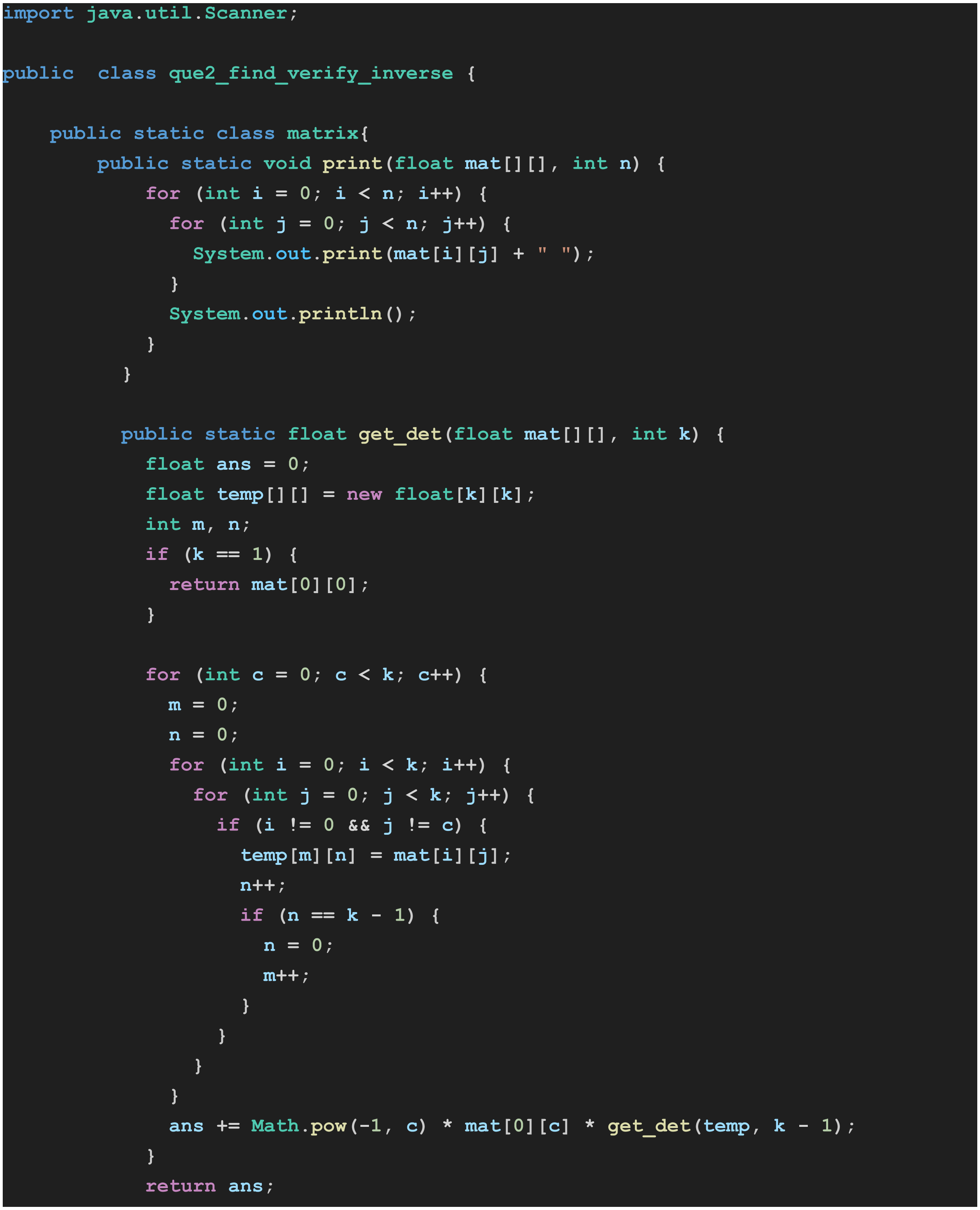


```
            System.out.println("key_matrix");
            print(key_matrix);

            // Now, 'inputString' contains the content of the file

            System.out.println("Input String: " + inputString.toString());

            String encrypted_text= hill_cipher(inputString.toString());

            calculate_inverse_matrix(key_matrix);

         String decrypted_text= decryption(encrypted_text);

         System.out.println("inverse matrix");
         print(inverse_matrix);
         System.out.println("encrypted_text: "+encrypted_text);
         System.out.println("decrypted_text: "+decrypted_text);




            /******************************** */
        } catch (FileNotFoundException e) {
            // Handle the exception if the file is not found
            System.err.println("File not found: " + e.getMessage());
        }

    /**************** */



  }
}
```



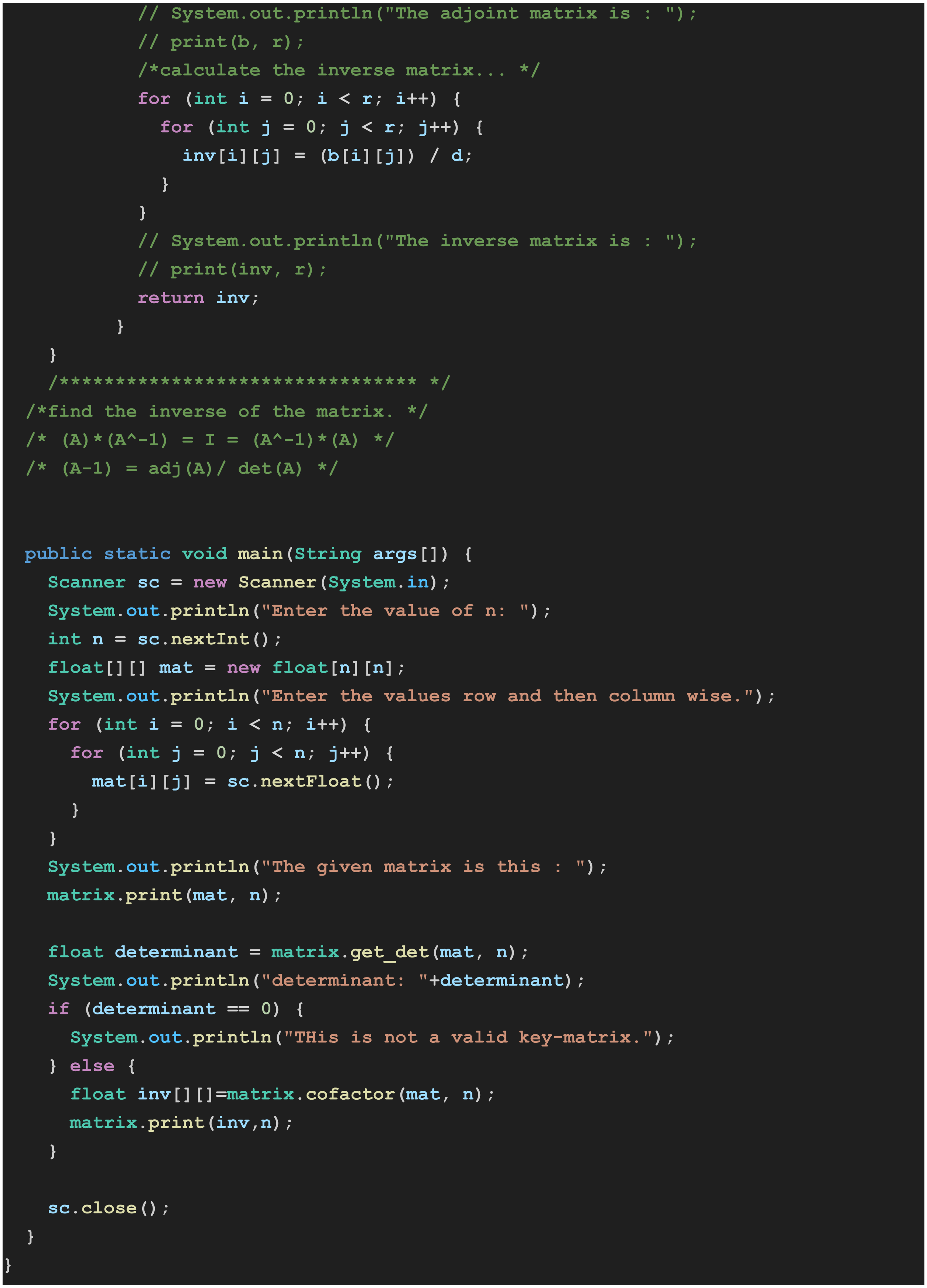
```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_05>java que1_2x2.java

key_matrix
3.0 0.0
0.0 2.0
Input String: flflfl.
5.0
11.0

15.0
22.0
------------
5.0
11.0

15.0
22.0
------------
5.0
11.0

15.0
22.0
------------
determinant : 6.0
inverse matrix
0.33333334 -0.0
-0.0 0.5
encrypted_text: pwpwpw
decrypted_text: flflfl

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_05>
```

2) Implement a program to generate and verify the key matrix for Hill cipher.

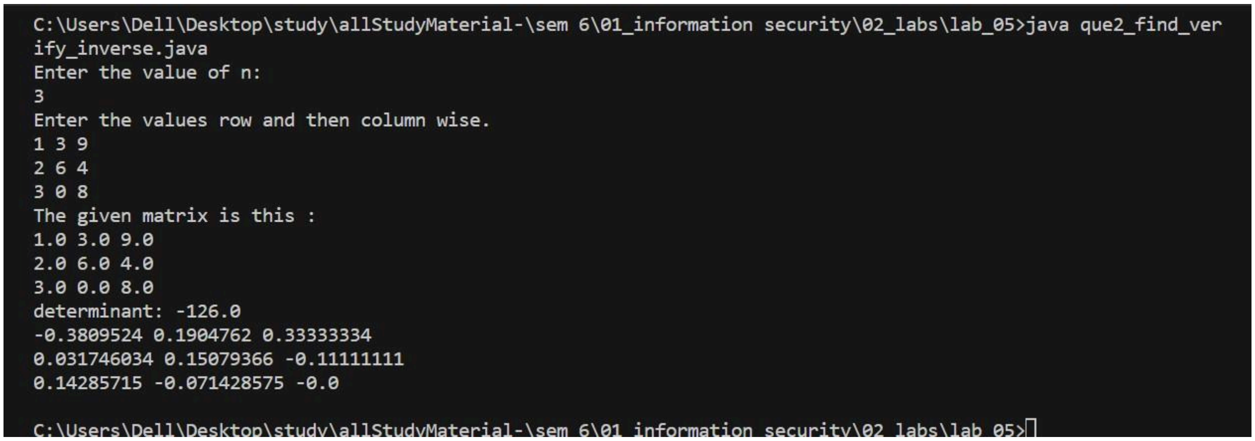


```java
import java.util.Scanner;

public  class que2_find_verify_inverse {

    public static class matrix{
        public static void print(float mat[][], int n) {
            for (int i = 0; i < n; i++) {
              for (int j = 0; j < n; j++) {
                System.out.print(mat[i][j] + " ");
              }
              System.out.println();
            }
          }

          public static float get_det(float mat[][], int k) {
            float ans = 0;
            float temp[][] = new float[k][k];
            int m, n;
            if (k == 1) {
              return mat[0][0];
            }

            for (int c = 0; c < k; c++) {
              m = 0;
              n = 0;
              for (int i = 0; i < k; i++) {
                for (int j = 0; j < k; j++) {
                  if (i != 0 && j != c) {
                    temp[m][n] = mat[i][j];
                    n++;
                    if (n == k - 1) {
                      n = 0;
                      m++;
                    }
                  }
                }
              }
              ans += Math.pow(-1, c) * mat[0][c] * get_det(temp, k - 1);
            }
            return ans;
```

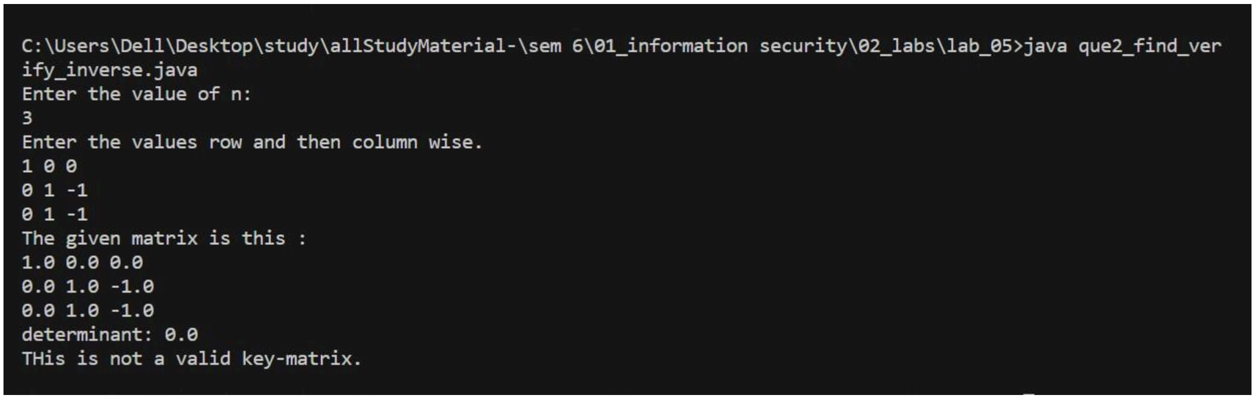
```
  }

  public static float[][] cofactor(float x[][], int k) {
    float b[][] = new float[10][10];
    float y[][] = new float[10][10];
    int p, q, m, n;
    int s = 1;
    for (q = 0; q < k; q++) {
      for (p = 0; p < k; p++) {
        m = 0;
        n = 0;
        for (int i = 0; i < k; i++) {
          for (int j = 0; j < k; j++) {
            if (i != q && j != p) {
              b[m][n] = x[i][j];
              if (n < (k - 2)) {
                n++;
              } else {
                n = 0;
                m++;
              }
            }
          }
        }
        y[q][p] = s * get_det(b, k - 1);
        s = (-1) * s;
      }
    }
    return transpose(x, y, k);
  }

  public static float[][] transpose(float x[][], float y[][], int r) {
    float b[][] = new float[10][10];
    float inv[][] = new float[10][10];
    float d;
    // System.out.println("The cofactor of matrix is : ");
    // print(y, r);
    /* transpose of a co-factor of the matrix is ... */
    for (int i = 0; i < r; i++) {
      for (int j = 0; j < r; j++) {
        b[i][j] = y[j][i];
      }
    }

    d = get_det(x, r);
```



```
            // System.out.println("The adjoint matrix is : ");
            // print(b, r);
            /*calculate the inverse matrix... */
            for (int i = 0; i < r; i++) {
              for (int j = 0; j < r; j++) {
                inv[i][j] = (b[i][j]) / d;
              }
            }
            // System.out.println("The inverse matrix is : ");
            // print(inv, r);
            return inv;
          }
    }
    /********************************* */
  /*find the inverse of the matrix. */
  /* (A)*(A^-1) = I = (A^-1)*(A) */
  /* (A-1) = adj(A)/ det(A) */


  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the value of n: ");
    int n = sc.nextInt();
    float[][] mat = new float[n][n];
    System.out.println("Enter the values row and then column wise.");
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        mat[i][j] = sc.nextFloat();
      }
    }
    System.out.println("The given matrix is this : ");
    matrix.print(mat, n);

    float determinant = matrix.get_det(mat, n);
    System.out.println("determinant: "+determinant);
    if (determinant == 0) {
      System.out.println("THis is not a valid key-matrix.");
    } else {
      float inv[][]=matrix.cofactor(mat, n);
      matrix.print(inv,n);
    }

    sc.close();
  }
}
```



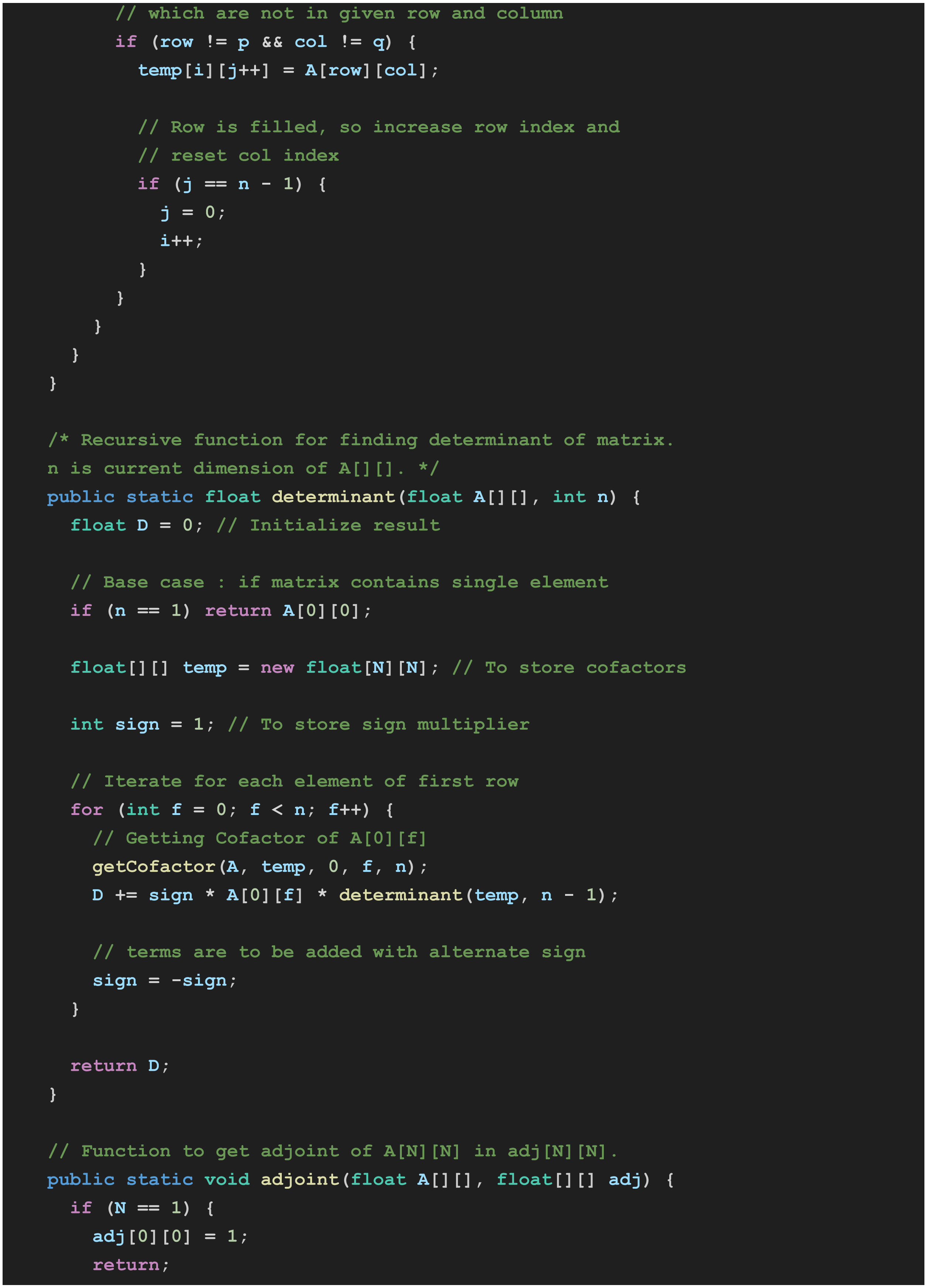
```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_05>java que2_find_ver
ify_inverse.java
Enter the value of n:
3
Enter the values row and then column wise.
1 3 9
2 6 4
3 0 8
The given matrix is this :
1.0 3.0 9.0
2.0 6.0 4.0
3.0 0.0 8.0
determinant: -126.0
-0.3809524 0.1904762 0.33333334
0.031746034 0.15079366 -0.11111111
0.14285715 -0.071428575 -0.0

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_05>
```



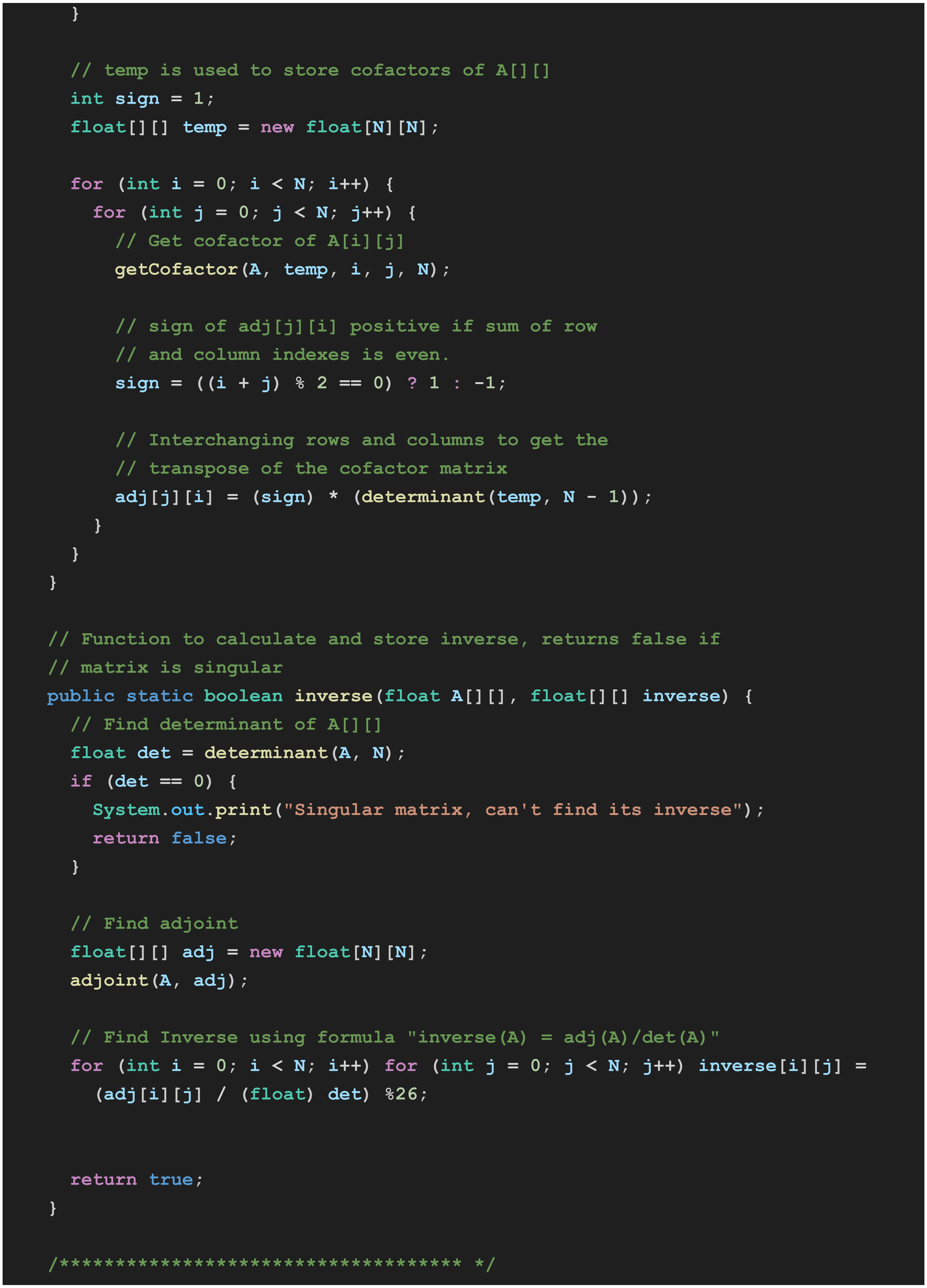
```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_05>java que2_find_ver
ify_inverse.java
Enter the value of n:
3
Enter the values row and then column wise.
1 0 0
0 1 -1
0 1 -1
The given matrix is this :
1.0 0.0 0.0
0.0 1.0 -1.0
0.0 1.0 -1.0
determinant: 0.0
THis is not a valid key-matrix.
```

3) Design and implement a program to perform encryption and decryption using the Hill Cipher for nxn matrix. Where 2<= n <= 5.

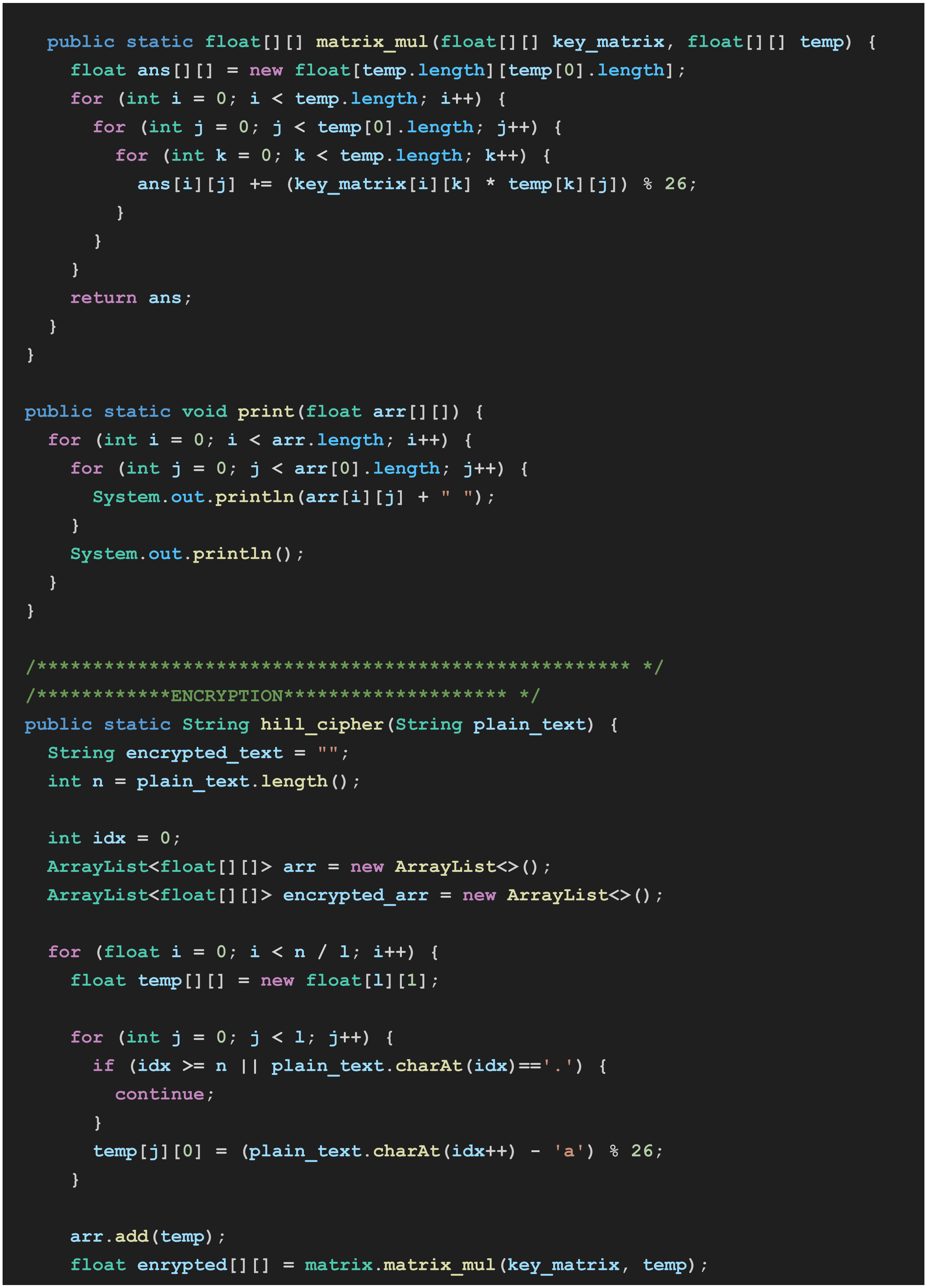
```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class que3_en_de_n {

  public static float key_matrix[][];
  public static float inverse_matrix[][];
  public static float adjoint[][];
  public static int l;
  public static int N;

  /************************* */

  public static class matrix {

    public static void print(float mat[][], int n) {
      for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
          System.out.print(mat[i][j] + " ");
        }
        System.out.println();
      }
    }


    public static void getCofactor(
      float A[][],
      float temp[][],
      int p,
      int q,
      int n
    ) {
      int i = 0, j = 0;

      // Looping for each element of the matrix
      for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
          // Copying into temporary matrix only those element
```



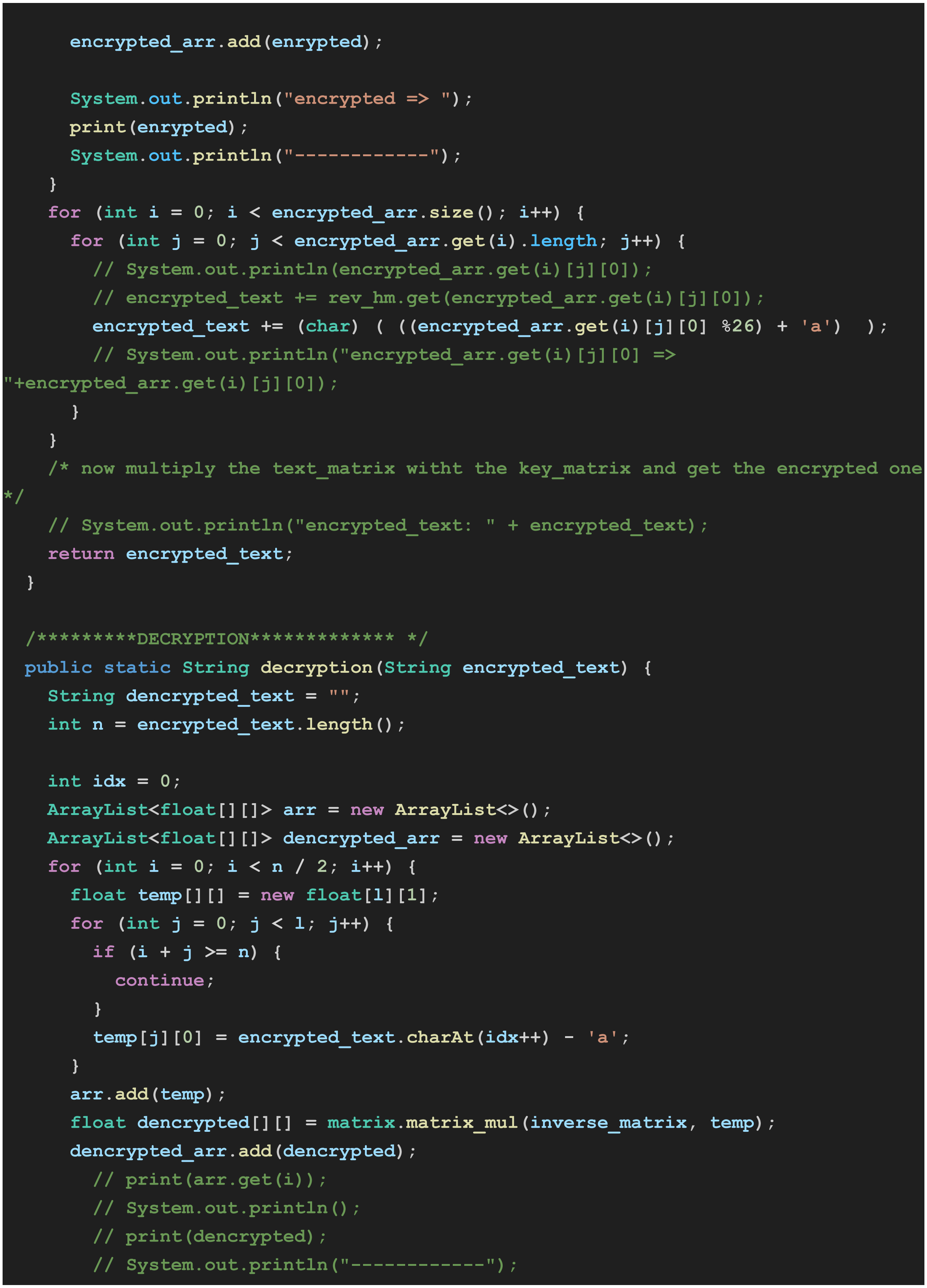
```
      // which are not in given row and column
      if (row != p && col != q) {
        temp[i][j++] = A[row][col];

        // Row is filled, so increase row index and
        // reset col index
        if (j == n - 1) {
          j = 0;
          i++;
        }
      }
    }
  }
}

/* Recursive function for finding determinant of matrix.
n is current dimension of A[][]. */
public static float determinant(float A[][], int n) {
  float D = 0; // Initialize result

  // Base case : if matrix contains single element
  if (n == 1) return A[0][0];

  float[][] temp = new float[N][N]; // To store cofactors

  int sign = 1; // To store sign multiplier

  // Iterate for each element of first row
  for (int f = 0; f < n; f++) {
    // Getting Cofactor of A[0][f]
    getCofactor(A, temp, 0, f, n);
    D += sign * A[0][f] * determinant(temp, n - 1);

    // terms are to be added with alternate sign
    sign = -sign;
  }

  return D;
}

// Function to get adjoint of A[N][N] in adj[N][N].
public static void adjoint(float A[][], float[][] adj) {
  if (N == 1) {
    adj[0][0] = 1;
    return;
```



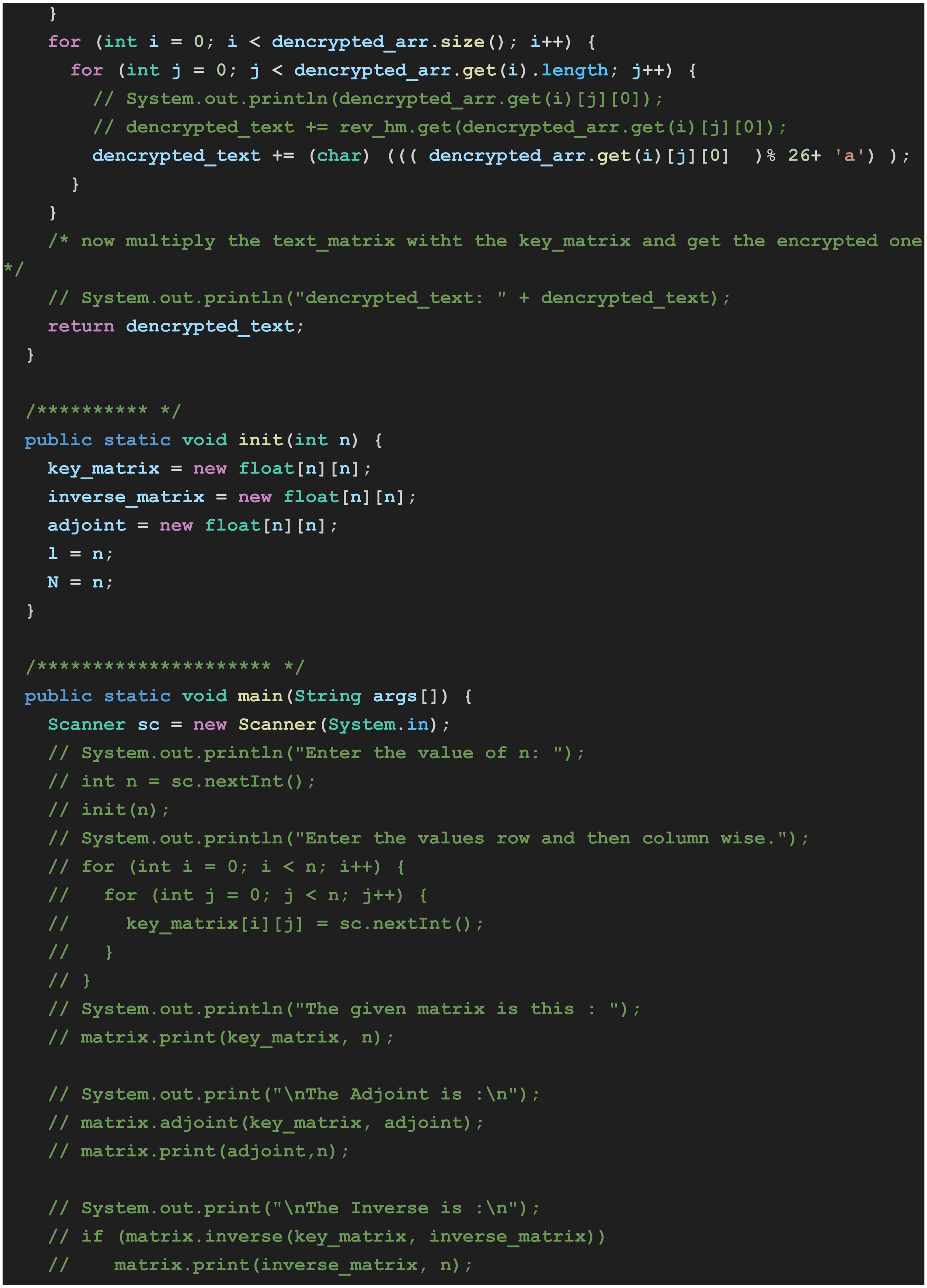
```
  }

  // temp is used to store cofactors of A[][]
  int sign = 1;
  float[][] temp = new float[N][N];

  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      // Get cofactor of A[i][j]
      getCofactor(A, temp, i, j, N);

      // sign of adj[j][i] positive if sum of row
      // and column indexes is even.
      sign = ((i + j) % 2 == 0) ? 1 : -1;

      // Interchanging rows and columns to get the
      // transpose of the cofactor matrix
      adj[j][i] = (sign) * (determinant(temp, N - 1));
    }
  }
}

// Function to calculate and store inverse, returns false if
// matrix is singular
public static boolean inverse(float A[][], float[][] inverse) {
  // Find determinant of A[][]
  float det = determinant(A, N);
  if (det == 0) {
    System.out.print("Singular matrix, can't find its inverse");
    return false;
  }

  // Find adjoint
  float[][] adj = new float[N][N];
  adjoint(A, adj);

  // Find Inverse using formula "inverse(A) = adj(A)/det(A)"
  for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) inverse[i][j] =
    (adj[i][j] / (float) det) %26;


  return true;
}

/*********************************** */
```



```
  public static float[][] matrix_mul(float[][] key_matrix, float[][] temp) {
    float ans[][] = new float[temp.length][temp[0].length];
    for (int i = 0; i < temp.length; i++) {
      for (int j = 0; j < temp[0].length; j++) {
        for (int k = 0; k < temp.length; k++) {
          ans[i][j] += (key_matrix[i][k] * temp[k][j]) % 26;
        }
      }
    }
    return ans;
  }
}

public static void print(float arr[][]) {
  for (int i = 0; i < arr.length; i++) {
    for (int j = 0; j < arr[0].length; j++) {
      System.out.println(arr[i][j] + " ");
    }
    System.out.println();
  }
}

/***************************************************** */
/************ENCRYPTION******************** */
public static String hill_cipher(String plain_text) {
  String encrypted_text = "";
  int n = plain_text.length();

  int idx = 0;
  ArrayList<float[][]> arr = new ArrayList<>();
  ArrayList<float[][]> encrypted_arr = new ArrayList<>();

  for (float i = 0; i < n / l; i++) {
    float temp[][] = new float[l][1];

    for (int j = 0; j < l; j++) {
      if (idx >= n || plain_text.charAt(idx)=='.') {
        continue;
      }
      temp[j][0] = (plain_text.charAt(idx++) - 'a') % 26;
    }

    arr.add(temp);
    float enrypted[][] = matrix.matrix_mul(key_matrix, temp);
```

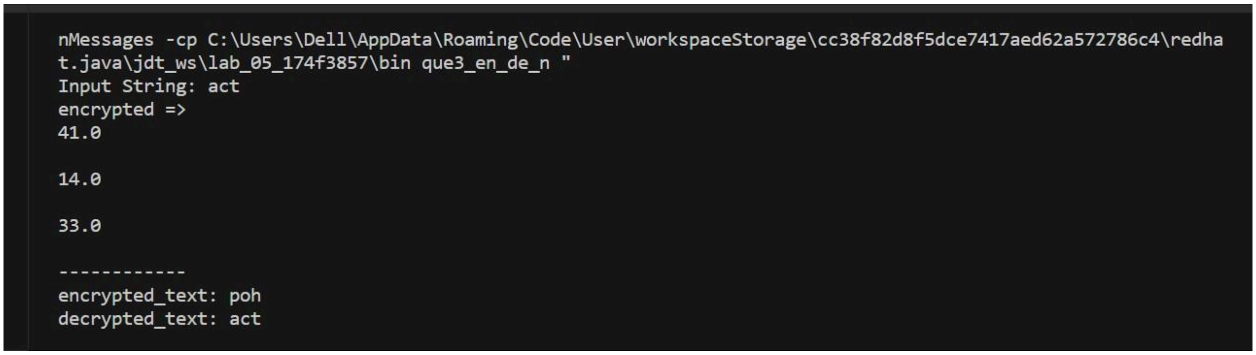


```
      encrypted_arr.add(enrypted);

      System.out.println("encrypted => ");
      print(enrypted);
      System.out.println("------------");
    }
    for (int i = 0; i < encrypted_arr.size(); i++) {
      for (int j = 0; j < encrypted_arr.get(i).length; j++) {
        // System.out.println(encrypted_arr.get(i)[j][0]);
        // encrypted_text += rev_hm.get(encrypted_arr.get(i)[j][0]);
        encrypted_text += (char) ( ((encrypted_arr.get(i)[j][0] %26) + 'a')  );
        // System.out.println("encrypted_arr.get(i)[j][0] =>
"+encrypted_arr.get(i)[j][0]);
      }
    }
    /* now multiply the text_matrix witht the key_matrix and get the encrypted one
*/
    // System.out.println("encrypted_text: " + encrypted_text);
    return encrypted_text;
  }

  /*********DECRYPTION************* */
  public static String decryption(String encrypted_text) {
    String dencrypted_text = "";
    int n = encrypted_text.length();

    int idx = 0;
    ArrayList<float[][]> arr = new ArrayList<>();
    ArrayList<float[][]> dencrypted_arr = new ArrayList<>();
    for (int i = 0; i < n / 2; i++) {
      float temp[][] = new float[l][1];
      for (int j = 0; j < l; j++) {
        if (i + j >= n) {
          continue;
        }
        temp[j][0] = encrypted_text.charAt(idx++) - 'a';
      }
      arr.add(temp);
      float dencrypted[][] = matrix.matrix_mul(inverse_matrix, temp);
      dencrypted_arr.add(dencrypted);
        // print(arr.get(i));
        // System.out.println();
        // print(dencrypted);
        // System.out.println("------------");
```



```
    }
    for (int i = 0; i < dencrypted_arr.size(); i++) {
      for (int j = 0; j < dencrypted_arr.get(i).length; j++) {
        // System.out.println(dencrypted_arr.get(i)[j][0]);
        // dencrypted_text += rev_hm.get(dencrypted_arr.get(i)[j][0]);
        dencrypted_text += (char)  ((( dencrypted_arr.get(i)[j][0]  )% 26+ 'a') );
      }
    }
    /* now multiply the text_matrix witht the key_matrix and get the encrypted one
*/
    // System.out.println("dencrypted_text: " + dencrypted_text);
    return dencrypted_text;
  }

  /********** */
  public static void init(int n) {
    key_matrix = new float[n][n];
    inverse_matrix = new float[n][n];
    adjoint = new float[n][n];
    l = n;
    N = n;
  }

  /******************** */
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    // System.out.println("Enter the value of n: ");
    // int n = sc.nextInt();
    // init(n);
    // System.out.println("Enter the values row and then column wise.");
    // for (int i = 0; i < n; i++) {
    //   for (int j = 0; j < n; j++) {
    //     key_matrix[i][j] = sc.nextInt();
    //   }
    // }
    // System.out.println("The given matrix is this : ");
    // matrix.print(key_matrix, n);

    // System.out.print("\nThe Adjoint is :\n");
    // matrix.adjoint(key_matrix, adjoint);
    // matrix.print(adjoint,n);

    // System.out.print("\nThe Inverse is :\n");
    // if (matrix.inverse(key_matrix, inverse_matrix))
    //    matrix.print(inverse_matrix, n);
```

```
        l =3;
    key_matrix = new float[][] { { 6,24,1 }, { 13,16,10 }, {20,17,15} };
    inverse_matrix = new float[][] { {8,5,10},{21,8,21},{21,12,8} };

    /******************* */

    String filePath = "source.txt";

    // Attempt to read the file
    try {
      // Create a File object
      File file = new File(filePath);

      // Create a Scanner to read the file
      Scanner scanner = new Scanner(file);

      // Read the input string
      StringBuilder inputString = new StringBuilder();
      while (scanner.hasNextLine()) {
        inputString.append(scanner.nextLine());
      }

      // Close the scanner
      scanner.close();

      // Now, 'inputString' contains the content of the file

      System.out.println("Input String: " + inputString.toString());

      String encrypted_text = hill_cipher(inputString.toString());

        String decrypted_text = decryption(encrypted_text);

      System.out.println("encrypted_text: " + encrypted_text);
        System.out.println("decrypted_text: " + decrypted_text);
      /******************************** */
    } catch (FileNotFoundException e) {
      // Handle the exception if the file is not found
      System.err.println("File not found: " + e.getMessage());
    }

    /**************************** */
    sc.close();
  }
}
```



```
nMessages -cp C:\Users\Dell\AppData\Roaming\Code\User\workspaceStorage\cc38f82d8f5dce7417aed62a572786c4\redha
t.java\jdt_ws\lab_05_174f3857\bin que3_en_de_n "
Input String: act
encrypted =>
41.0

14.0

33.0

------------
encrypted_text: poh
decrypted_text: act
```