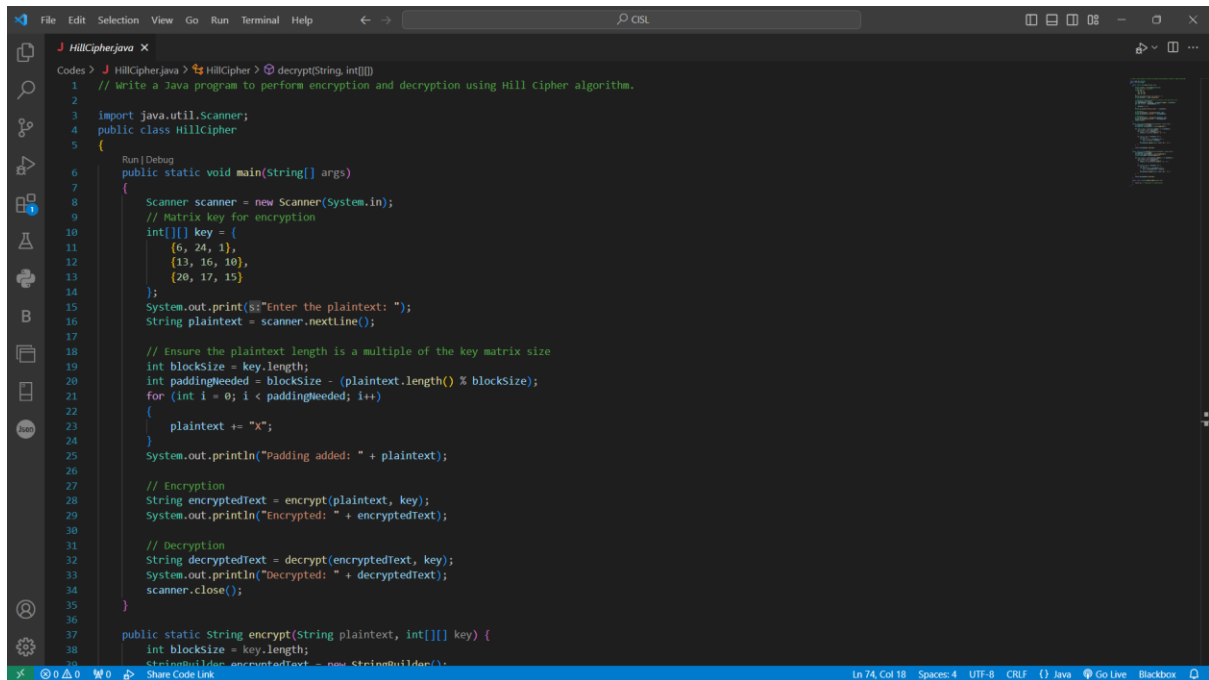
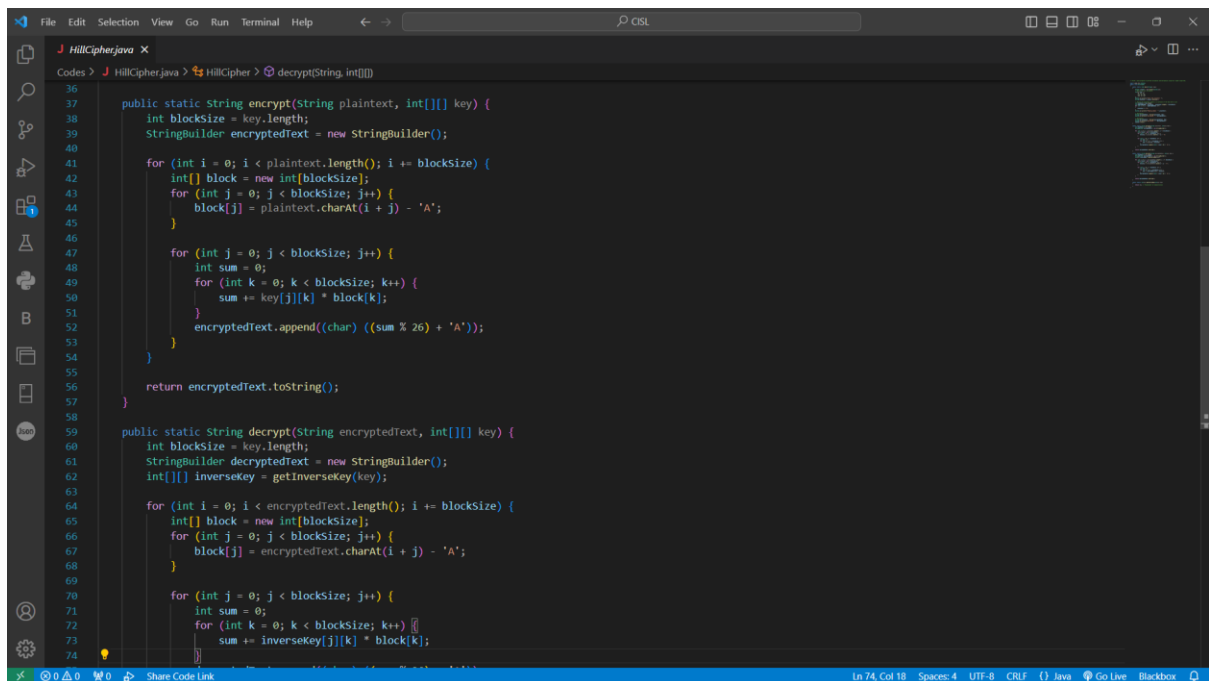


5. Write a Java program to perform encryption and decryption using Hill Cipher algorithm.

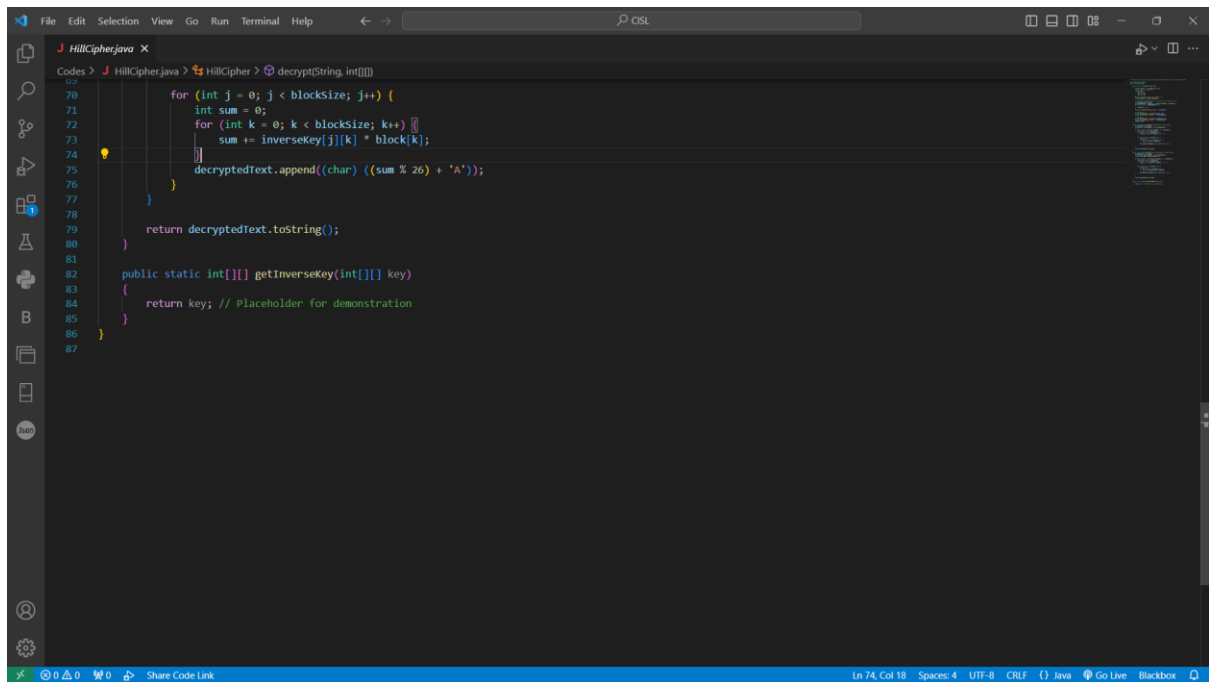
Program:



```
File Edit Selection View Go Run Terminal Help
HillCipher.java
Codes > J HillCipher.java > HillCipher > decrypt(String, int[][])
1 // Write a Java program to perform encryption and decryption using Hill Cipher algorithm.
2
3 import java.util.Scanner;
4 public class HillCipher
5 {
6     public static void main(String[] args)
7     {
8         Scanner scanner = new Scanner(System.in);
9         // Matrix key for encryption
10        int[][] key = {
11            {6, 24, 1},
12            {13, 16, 10},
13            {20, 17, 15}
14        };
15        System.out.print("Enter the plaintext: ");
16        String plaintext = scanner.nextLine();
17
18        // Ensure the plaintext length is a multiple of the key matrix size
19        int blockSize = key.length;
20        int paddingNeeded = blockSize - (plaintext.length() % blockSize);
21        for (int i = 0; i < paddingNeeded; i++)
22        {
23            plaintext += "X";
24        }
25        System.out.println("Padding added: " + plaintext);
26
27        // Encryption
28        String encryptedText = encrypt(plaintext, key);
29        System.out.println("Encrypted: " + encryptedText);
30
31        // Decryption
32        String decryptedText = decrypt(encryptedText, key);
33        System.out.println("Decrypted: " + decryptedText);
34        scanner.close();
35    }
36
37    public static String encrypt(String plaintext, int[][] key) {
38        int blockSize = key.length;
39        StringBuilder encryptedText = new StringBuilder();
40
41        for (int i = 0; i < plaintext.length(); i += blockSize) {
42            int[] block = new int[blockSize];
43            for (int j = 0; j < blockSize; j++) {
44                block[j] = plaintext.charAt(i + j) - 'A';
45            }
46
47            for (int j = 0; j < blockSize; j++) {
48                int sum = 0;
49                for (int k = 0; k < blockSize; k++) {
50                    sum += key[j][k] * block[k];
51                }
52                encryptedText.append((char) ((sum % 26) + 'A'));
53            }
54        }
55
56        return encryptedText.toString();
57    }
58
59    public static String decrypt(String encryptedText, int[][] key) {
60        int blockSize = key.length;
61        StringBuilder decryptedText = new StringBuilder();
62        int[][] inverseKey = getInverseKey(key);
63
64        for (int i = 0; i < encryptedText.length(); i += blockSize) {
65            int[] block = new int[blockSize];
66            for (int j = 0; j < blockSize; j++) {
67                block[j] = encryptedText.charAt(i + j) - 'A';
68            }
69
70            for (int j = 0; j < blockSize; j++) {
71                int sum = 0;
72                for (int k = 0; k < blockSize; k++) {
73                    sum += inverseKey[j][k] * block[k];
74                }
75            }
76
77            decryptedText.append((char) ((sum % 26) + 'A'));
78        }
79
80        return decryptedText.toString();
81    }
82
83    private static int[][] getInverseKey(int[][] key) {
84        int n = key.length;
85        int[][] inverse = new int[n][n];
86        int[][] adj = new int[n][n];
87        for (int i = 0; i < n; i++) {
88            for (int j = 0; j < n; j++) {
89                adj[j][i] = key[i][j];
90            }
91        }
92        int det = determinant(adj);
93        for (int i = 0; i < n; i++) {
94            for (int j = 0; j < n; j++) {
95                inverse[j][i] = adj[i][j] * det;
96            }
97        }
98        for (int i = 0; i < n; i++) {
99            for (int j = 0; j < n; j++) {
100                inverse[j][i] = inverse[j][i] % n;
101            }
102        }
103        return inverse;
104    }
105
106    private static int determinant(int[][] matrix) {
107        int n = matrix.length;
108        if (n == 1) return matrix[0][0];
109        if (n == 2) return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
110        int det = 0;
111        for (int i = 0; i < n; i++) {
112            int sign = 1;
113            for (int j = 0; j < n; j++) {
114                if (j % 2 == 0) sign = 1;
115                else sign = -1;
116            }
117            det += sign * matrix[0][i] * determinant(removeRowAndCol(matrix, 0, i));
118        }
119        return det;
120    }
121
122    private static int[][] removeRowAndCol(int[][] matrix, int row, int col) {
123        int n = matrix.length;
124        int[][] result = new int[n-1][n-1];
125        for (int i = 0; i < n-1; i++) {
126            for (int j = 0; j < n-1; j++) {
127                if (i < row) result[i][j] = matrix[i][j];
128                else result[i][j] = matrix[i+1][j];
129            }
130        }
131        return result;
132    }
133}
```

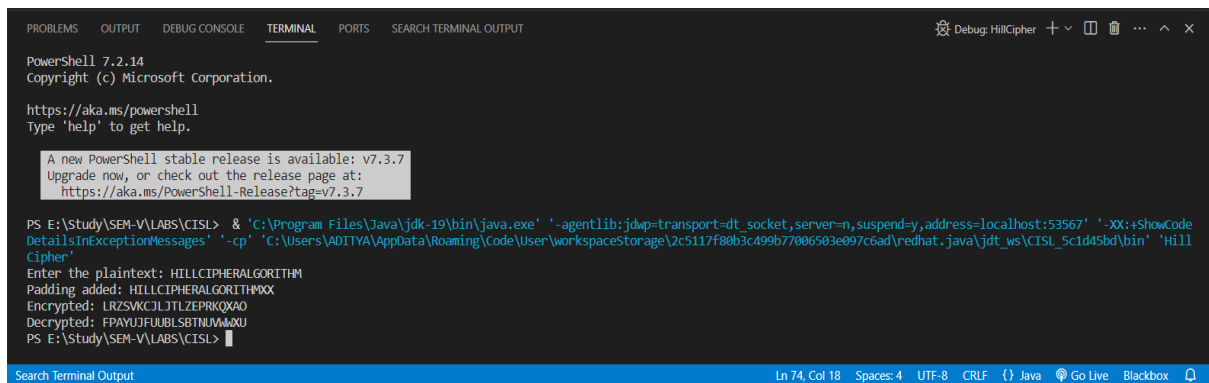


```
File Edit Selection View Go Run Terminal Help
HillCipher.java
Codes > J HillCipher.java > HillCipher > decrypt(String, int[][])
36
37    public static String encrypt(String plaintext, int[][] key) {
38        int blockSize = key.length;
39        StringBuilder encryptedText = new StringBuilder();
40
41        for (int i = 0; i < plaintext.length(); i += blockSize) {
42            int[] block = new int[blockSize];
43            for (int j = 0; j < blockSize; j++) {
44                block[j] = plaintext.charAt(i + j) - 'A';
45            }
46
47            for (int j = 0; j < blockSize; j++) {
48                int sum = 0;
49                for (int k = 0; k < blockSize; k++) {
50                    sum += key[j][k] * block[k];
51                }
52                encryptedText.append((char) ((sum % 26) + 'A'));
53            }
54        }
55
56        return encryptedText.toString();
57    }
58
59    public static String decrypt(String encryptedText, int[][] key) {
60        int blockSize = key.length;
61        StringBuilder decryptedText = new StringBuilder();
62        int[][] inverseKey = getInverseKey(key);
63
64        for (int i = 0; i < encryptedText.length(); i += blockSize) {
65            int[] block = new int[blockSize];
66            for (int j = 0; j < blockSize; j++) {
67                block[j] = encryptedText.charAt(i + j) - 'A';
68            }
69
70            for (int j = 0; j < blockSize; j++) {
71                int sum = 0;
72                for (int k = 0; k < blockSize; k++) {
73                    sum += inverseKey[j][k] * block[k];
74                }
75            }
76
77            decryptedText.append((char) ((sum % 26) + 'A'));
78        }
79
80        return decryptedText.toString();
81    }
82
83    private static int[][] getInverseKey(int[][] key) {
84        int n = key.length;
85        int[][] inverse = new int[n][n];
86        int[][] adj = new int[n][n];
87        for (int i = 0; i < n; i++) {
88            for (int j = 0; j < n; j++) {
89                adj[j][i] = key[i][j];
90            }
91        }
92        int det = determinant(adj);
93        for (int i = 0; i < n; i++) {
94            for (int j = 0; j < n; j++) {
95                inverse[j][i] = adj[i][j] * det;
96            }
97        }
98        for (int i = 0; i < n; i++) {
99            for (int j = 0; j < n; j++) {
100                inverse[j][i] = inverse[j][i] % n;
101            }
102        }
103        return inverse;
104    }
105
106    private static int determinant(int[][] matrix) {
107        int n = matrix.length;
108        if (n == 1) return matrix[0][0];
109        if (n == 2) return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
110        int det = 0;
111        for (int i = 0; i < n; i++) {
112            int sign = 1;
113            for (int j = 0; j < n; j++) {
114                if (j % 2 == 0) sign = 1;
115                else sign = -1;
116            }
117            det += sign * matrix[0][i] * determinant(removeRowAndCol(matrix, 0, i));
118        }
119        return det;
120    }
121
122    private static int[][] removeRowAndCol(int[][] matrix, int row, int col) {
123        int n = matrix.length;
124        int[][] result = new int[n-1][n-1];
125        for (int i = 0; i < n-1; i++) {
126            for (int j = 0; j < n-1; j++) {
127                if (i < row) result[i][j] = matrix[i][j];
128                else result[i][j] = matrix[i+1][j];
129            }
130        }
131        return result;
132    }
133}
```



```
69
70     for (int j = 0; j < blockSize; j++) {
71         int sum = 0;
72         for (int k = 0; k < blockSize; k++) {
73             sum += inverseKey[j][k] * block[k];
74         }
75         decryptedText.append((char) ((sum % 26) + 'A'));
76     }
77 }
78
79 return decryptedText.toString();
80 }
81
82 public static int[][] getInverseKey(int[][] key)
83 {
84     return key; // Placeholder for demonstration
85 }
86
87 }
```

Output:



```
PowerShell 7.2.14
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

A new PowerShell stable release is available: v7.3.7
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.3.7

PS E:\Study\SEM-V\LABS\CISL> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:53567' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ADITYA\AppData\Roaming\Code\User\workspaceStorage\2c5117f80b3c499b77006503e097c6ad\redhat.java\jdt_ws\CISL_5c1d45bd\bin' 'HillCipher'
Enter the plaintext: HILLCIPHERALGORITHM
Padding added: HILLCIPHERALGORITHMXX
Encrypted: LRZSVKCDJTLZEPKQXAO
Decrypted: FPAYUJFUUBLSBNTNUMXQXU
PS E:\Study\SEM-V\LABS\CISL> 
```