**Assignment - 2**

**CS0557 - Cryptography Laboratory**

**Name:** Vikas Pundir

**Roll No:** 25203112

**Section:** MTech CSE(IS) First Year

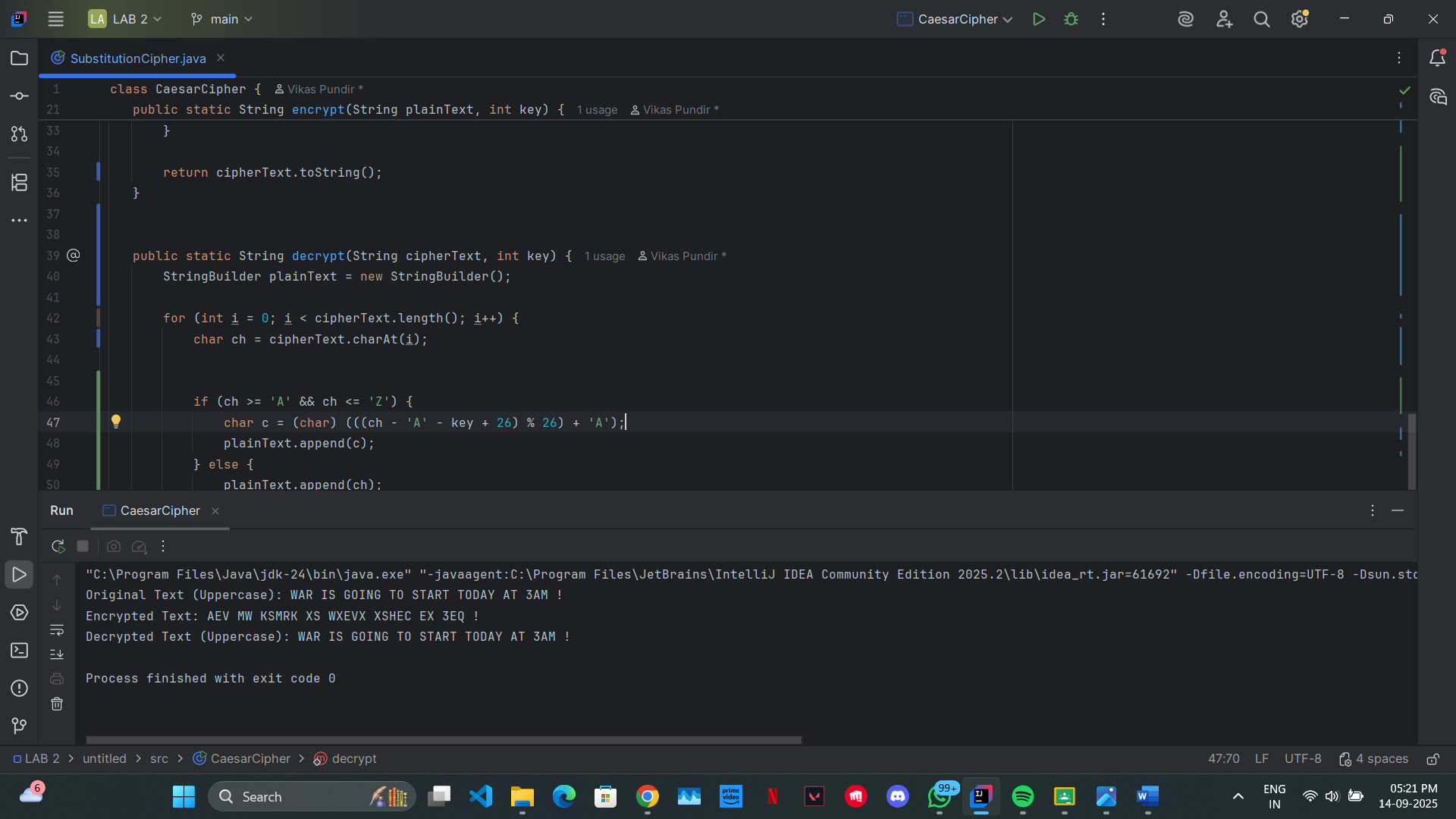
**Date:** 09 September 2025

**Question 1: Implement Caesar substitution cipher.**

**Ans:**

class CaesarCipher {  
  
 public static void main(String[] args) {  
  
 String text = "War is going to start today at 3AM !";  
 int key = 4;  
  
  
 text = text.toUpperCase();  
  
 String encryptedText = *encrypt*(text, key);  
 System.*out*.println("Original Text (Uppercase): " + text);  
 System.*out*.println("Encrypted Text: " + encryptedText);  
  
  
 String decryptedText = *decrypt*(encryptedText, key);  
 System.*out*.println("Decrypted Text (Uppercase): " + decryptedText);  
 }  
  
  
 public static String encrypt(String plainText, int key) {  
 StringBuilder cipherText = new StringBuilder();  
  
 for (int i = 0; i < plainText.length(); i++) {  
 char ch = plainText.charAt(i);  
  
 if (ch >= 'A' && ch <= 'Z') {  
 char c = (char) (((ch - 'A' + key) % 26) + 'A');  
 cipherText.append(c);  
 } else {  
 cipherText.append(ch);  
 }  
 }  
  
 return cipherText.toString();  
 }  
  
  
 public static String decrypt(String cipherText, int key) {  
 StringBuilder plainText = new StringBuilder();  
  
 for (int i = 0; i < cipherText.length(); i++) {  
 char ch = cipherText.charAt(i);  
  
  
 if (ch >= 'A' && ch <= 'Z') {  
 char c = (char) (((ch - 'A' - key + 26) % 26) + 'A');  
 plainText.append(c);  
 } else {  
 plainText.append(ch);  
 }  
 }  
  
 return plainText.toString();  
 }  
}

**Output:**

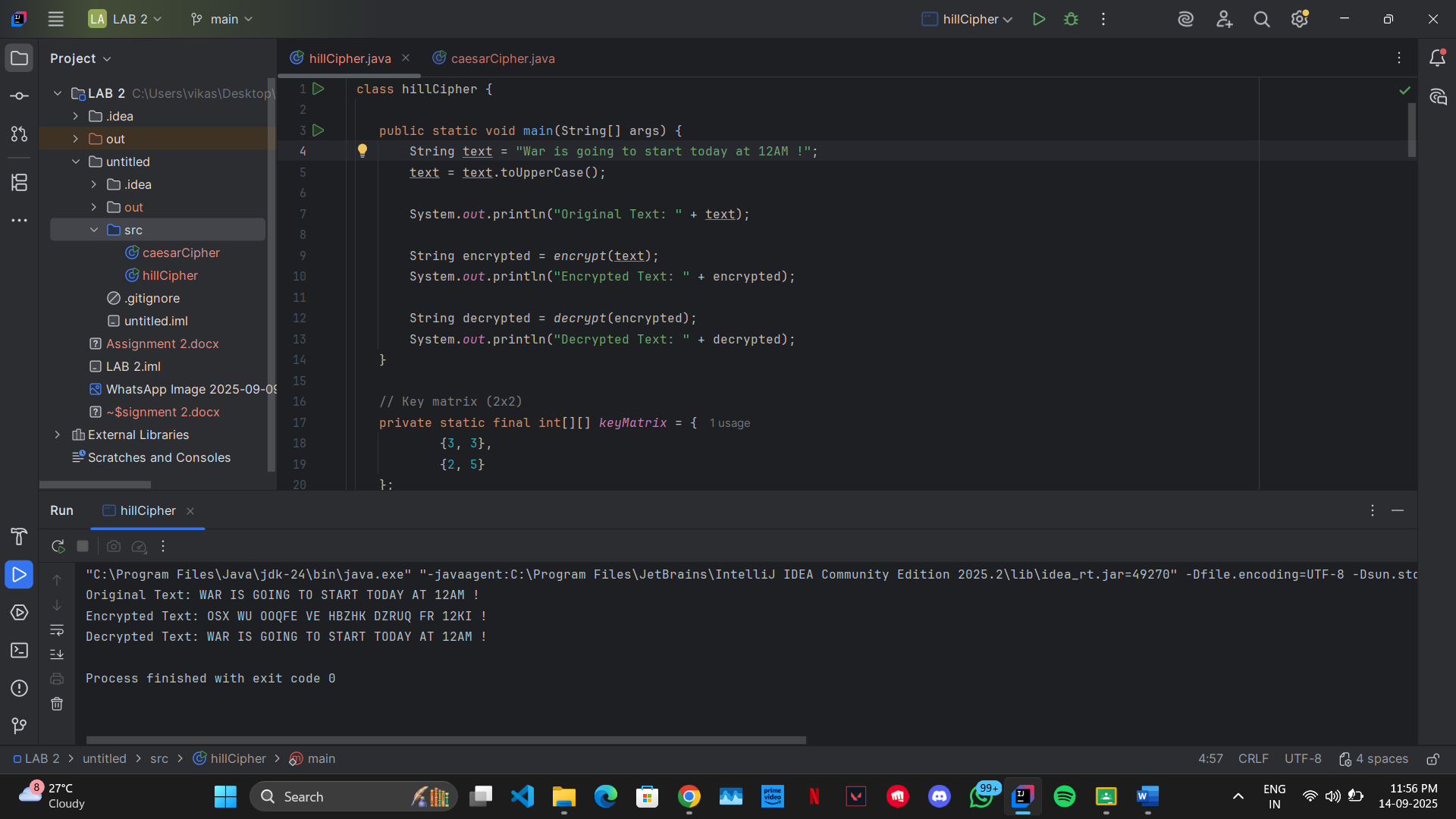


**Question 2: Implement Hill cipher.**

**Ans:**

class hillCipher {  
  
 public static void main(String[] args) {  
 String text = "War is going to start today at 3AM !";  
 text = text.toUpperCase();  
  
 System.*out*.println("Original Text: " + text);  
  
 String encrypted = *encrypt*(text);  
 System.*out*.println("Encrypted Text: " + encrypted);  
  
 String decrypted = *decrypt*(encrypted);  
 System.*out*.println("Decrypted Text: " + decrypted);  
 }  
  
 // Key matrix (2x2)  
 private static final int[][] *keyMatrix* = {  
 {3, 3},  
 {2, 5}  
 };  
  
 // Inverse matrix (2x2) mod 26  
 private static final int[][] *inverseKeyMatrix* = {  
 {15, 17},  
 {20, 9}  
 };  
  
 // Encrypt while preserving non-letter characters  
 public static String encrypt(String text) {  
 StringBuilder cipher = new StringBuilder();  
 StringBuilder letters = new StringBuilder();  
  
 // Collect only letters  
 for (int i = 0; i < text.length(); i++) {  
 char ch = text.charAt(i);  
 if (Character.*isLetter*(ch)) {  
 letters.append(ch);  
 }  
 }  
  
 // Pad if odd  
 if (letters.length() % 2 != 0) {  
 letters.append('X');  
 }  
  
 // Encrypt the letters  
 StringBuilder encryptedLetters = new StringBuilder();  
 for (int i = 0; i < letters.length(); i += 2) {  
 int[] vector = {  
 letters.charAt(i) - 'A',  
 letters.charAt(i + 1) - 'A'  
 };  
  
 int[] result = *multiplyMatrix*(*keyMatrix*, vector);  
  
 encryptedLetters.append((char) (result[0] + 'A'));  
 encryptedLetters.append((char) (result[1] + 'A'));  
 }  
  
 // Reconstruct the final encrypted text  
 int letterIndex = 0;  
 for (int i = 0; i < text.length(); i++) {  
 char ch = text.charAt(i);  
 if (Character.*isLetter*(ch)) {  
 cipher.append(encryptedLetters.charAt(letterIndex));  
 letterIndex++;  
 } else {  
 cipher.append(ch);  
 }  
 }  
  
 return cipher.toString();  
 }  
  
 // Decrypt while preserving non-letter characters  
 public static String decrypt(String text) {  
 StringBuilder plain = new StringBuilder();  
 StringBuilder letters = new StringBuilder();  
  
 // Collect only letters  
 for (int i = 0; i < text.length(); i++) {  
 char ch = text.charAt(i);  
 if (Character.*isLetter*(ch)) {  
 letters.append(ch);  
 }  
 }  
  
 // Pad if odd  
 if (letters.length() % 2 != 0) {  
 letters.append('X');  
 }  
  
 // Decrypt the letters  
 StringBuilder decryptedLetters = new StringBuilder();  
 for (int i = 0; i < letters.length(); i += 2) {  
 int[] vector = {  
 letters.charAt(i) - 'A',  
 letters.charAt(i + 1) - 'A'  
 };  
  
 int[] result = *multiplyMatrix*(*inverseKeyMatrix*, vector);  
  
 decryptedLetters.append((char) (result[0] + 'A'));  
 decryptedLetters.append((char) (result[1] + 'A'));  
 }  
  
 // Reconstruct the final decrypted text  
 int letterIndex = 0;  
 for (int i = 0; i < text.length(); i++) {  
 char ch = text.charAt(i);  
 if (Character.*isLetter*(ch)) {  
 plain.append(decryptedLetters.charAt(letterIndex));  
 letterIndex++;  
 } else {  
 plain.append(ch);  
 }  
 }  
  
 return plain.toString();  
 }  
  
 // Multiply a 2x2 matrix with a 2x1 vector mod 26  
 private static int[] multiplyMatrix(int[][] matrix, int[] vector) {  
 int[] result = new int[2];  
 result[0] = (matrix[0][0] \* vector[0] + matrix[0][1] \* vector[1]) % 26;  
 result[1] = (matrix[1][0] \* vector[0] + matrix[1][1] \* vector[1]) % 26;  
 return result;  
 }  
}

**Output:**



\*\*\*\*\*