CODE SUMMARY

This Java program implements the columnar transposition cipher, a classical encryption technique. It allows users to encrypt and decrypt messages using a numeric key that determines the rearrangement order of characters in columns. The program takes user input, processes it according to the cipher rules, displays intermediate tables for visualization, and offers the option to repeat the process for multiple messages.

In the main method, the user is prompted to enter a plaintext message and a key. The plaintext must contain only alphabetic characters, and the key must be a string of digits. The code validates both inputs: it ensures the message is alphabetic using isAlphabetic() and the key is numeric using isNumeric(). It also checks that the key contains no repeated numbers and only digits within the valid range (i.e., all digits from 1 to the length of the key).

In the encrypt() method, the plaintext is first converted to uppercase and loaded into a 2D character array (table), filling it row by row. If the last row has empty cells, they're filled with a dash (-) as a placeholder. For better visualization, the table is printed with random letters replacing the dashes when displaying the encryption table. To produce the ciphertext, the columns are read in the order specified by the key, skipping any placeholder characters. The result is a scrambled string representing the encrypted version of the input.

If the user chooses to decrypt the encrypted message, the program uses the decrypt() method. It first calculates how many characters each column should have based on the original key and message length. Then, it fills the columns in the correct order with characters from the ciphertext. After reconstructing the table, the program reads it row by row to retrieve the original message. Like during encryption, it also prints a visual table to show how the message is reconstructed.

The program runs inside a loop that allows repeated use. After each encryption and optional decryption, the user is asked whether they want to encrypt another message. If the user chooses not to continue, the program exits gracefully with a goodbye message. This interactive loop makes the tool user-friendly for trying multiple encryptions and decryptions in one session.