AIM:- WAP to implement cryptanalysis or decoding using playfair cipher

Program:-

```
import java.util.*;

public class PlayfairDecrypt {

    // Preprocess text: uppercase, remove non-letters, replace J with I
    static String preprocessText(String text) {
        text = text.toUpperCase().replace("J", "I");
        text = text.replaceAll("[^A-Z]", "");
        return text;
    }

    // Generate 5x5 key matrix
    static char[][] generateKeyMatrix(String key) {
        key = preprocessText(key);
        LinkedHashSet<Character> set = new LinkedHashSet<>();

        for (char c : key.toCharArray()) set.add(c);
        for (char c : "ABCDEFGHIKLMNOPQRSTUVWXYZ".toCharArray()) set.add(c);

        char[][] matrix = new char[5][5];
        Iterator<Character> it = set.iterator();
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                matrix[i][j] = it.next();
            }
        }
        return matrix;
    }

    // Find position of a letter in the matrix
    static int[] findPosition(char[][] matrix, char ch) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (matrix[i][j] == ch) return new int[]{i, j};
            }
        }
        return null;
    }

    // Decrypt Playfair
    static String decrypt(String ciphertext, String key) {
        char[][] matrix = generateKeyMatrix(key);
        ciphertext = preprocessText(ciphertext);
```

```java
        StringBuilder plaintext = new StringBuilder();

        for (int i = 0; i < ciphertext.length(); i += 2) {
            char a = ciphertext.charAt(i);
            char b = ciphertext.charAt(i + 1);
            int[] posA = findPosition(matrix, a);
            int[] posB = findPosition(matrix, b);

            if (posA[0] == posB[0]) { // Same row
                plaintext.append(matrix[posA[0]][(posA[1] + 4) % 5]);
                plaintext.append(matrix[posB[0]][(posB[1] + 4) % 5]);
            } else if (posA[1] == posB[1]) { // Same column
                plaintext.append(matrix[(posA[0] + 4) % 5][posA[1]]);
                plaintext.append(matrix[(posB[0] + 4) % 5][posB[1]]);
            } else { // Rectangle swap
                plaintext.append(matrix[posA[0]][posB[1]]);
                plaintext.append(matrix[posB[0]][posA[1]]);
            }
        }
        return plaintext.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter key: ");
        String key = sc.nextLine();
        System.out.print("Enter ciphertext: ");
        String ciphertext = sc.nextLine();

        String plaintext = decrypt(ciphertext, key);
        System.out.println("Decrypted plaintext: " + plaintext);
        sc.close();
    }
}
```

Ciphertext- podrdrpobngeiolido
Key- guidance

Ciphertext- gatlmzclrqtx
Key - monarchy