

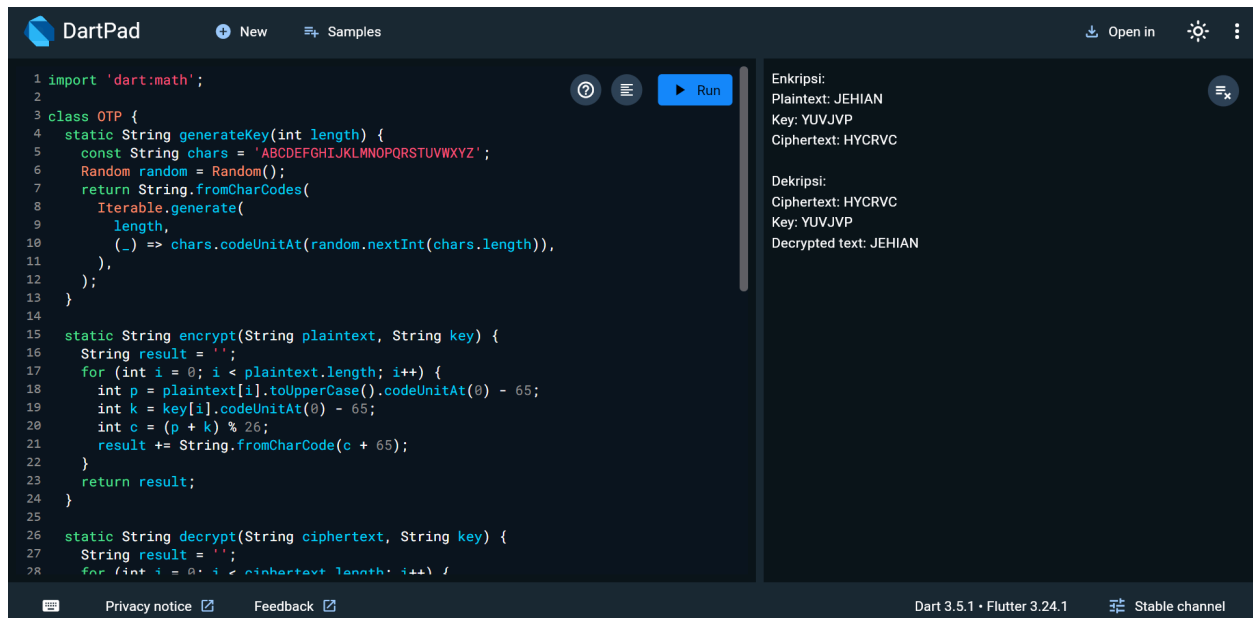
NAMA : JEHIAN ATHAYA TSANI AZ ZUHRY

NIM : H1D022006

SHIFT ASLI : C

SHIFT BARU : D

## PENJELASAN TUGAS PERTEMUAN 1 PRAKTIKUM MOBILE



```
1 import 'dart:math';
2
3 class OTP {
4   static String generateKey(int length) {
5     const String chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
6     Random random = Random();
7     return String.fromCharCode(
8       Iterable.generate(
9         length,
10        (_) => chars.codeUnitAt(random.nextInt(chars.length)),
11      ),
12    );
13  }
14
15  static String encrypt(String plaintext, String key) {
16    String result = '';
17    for (int i = 0; i < plaintext.length; i++) {
18      int p = plaintext[i].toUpperCase().codeUnitAt(0) - 65;
19      int k = key[i].codeUnitAt(0) - 65;
20      int c = (p + k) % 26;
21      result += String.fromCharCode(c + 65);
22    }
23    return result;
24  }
25
26  static String decrypt(String ciphertext, String key) {
27    String result = '';
28    for (int i = 0; i < ciphertext.length; i++) {
```

Enkripsi:  
Plaintext: JEHIAN  
Key: YUVJVP  
Ciphertext: HYCRVC

Dekripsi:  
Ciphertext: HYCRVC  
Key: YUVJVP  
Decrypted text: JEHIAN

Dart 3.5.1 • Flutter 3.24.1 Stable channel

Program dart yang saya buat yaitu **implementasi** dari algoritma enkripsi One-Time Pad (OTP) menggunakan bahasa pemrograman Dart.

**Tujuan** program ini adalah untuk mendemonstrasikan cara kerja enkripsi dan dekripsi menggunakan metode One-Time Pad. OTP adalah teknik enkripsi yang aman jika digunakan dengan benar, di mana setiap karakter plaintext dienkripsi menggunakan kunci yang berbeda dan hanya digunakan sekali.

**Penerapan** dari modul 1:

1. Penggunaan kelas dan objek (class OTP)
2. Implementasi fungsi (generateKey, encrypt, decrypt)
3. Penggunaan tipe data (String, int)
4. Operasi matematika dan logika (misalnya, operasi modulo)

5. Struktur kontrol (perulangan for)
6. Penggunaan fungsi main()

Hal yang saya **pelajari sendiri**:

1. Penggunaan library dart:math untuk menghasilkan angka acak
2. Manipulasi karakter dan kode ASCII
3. Penggunaan metode-metode String seperti replaceAll(), toUpperCase(), dll.
4. Implementasi algoritma kriptografi

**Penjelasan kode:**

```
class OTP {  
  static String generateKey(int length) {  
    const String chars = 'ABCDEFGHJKLMNOPQRSTUVWXYZ';  
    Random random = Random();  
    return String.fromCharCode(  
      Iterable.generate(  
        length,  
        (_) => chars.codeUnitAt(random.nextInt(chars.length)),  
      ),  
    );  
  }  
}
```

Metode generateKey menghasilkan kunci acak dengan panjang tertentu. Ini menggunakan karakter A-Z dan Random untuk memilih karakter secara acak.

```
static String encrypt(String plaintext, String key) {  
  String result = '';  
  for (int i = 0; i < plaintext.length; i++) {  
    int p = plaintext[i].toUpperCase().codeUnitAt(0) - 65;  
    int k = key[i].codeUnitAt(0) - 65;  
    int c = (p + k) % 26;  
    result += String.fromCharCode(c + 65);  
  }  
  return result;  
}
```

```
}
```

Metode encrypt mengenkripsi plaintext menggunakan kunci. Setiap karakter plaintext diubah menjadi angka (A=0, B=1, dst), ditambahkan dengan karakter kunci yang sesuai, lalu diambil modulo 26 untuk mendapatkan karakter ciphertext.

```
static String decrypt(String ciphertext, String key) {  
    String result = '';  
    for (int i = 0; i < ciphertext.length; i++) {  
        int c = ciphertext[i].codeUnitAt(0) - 65;  
        int k = key[i].codeUnitAt(0) - 65;  
        int p = (c - k + 26) % 26;  
        result += String.fromCharCode(p + 65);  
    }  
    return result;  
}  
}
```

Metode decrypt melakukan proses sebaliknya dari encrypt untuk mendapatkan plaintext kembali dari ciphertext.

```
void main() {  
    String plaintext = 'JEHIAN';  
    plaintext = plaintext.replaceAll(' ', '').toUpperCase();  
    String key = OTP.generateKey(plaintext.length);  
  
    String ciphertext = OTP.encrypt(plaintext, key);  
    print('Enkripsi:');  
    print('Plaintext: $plaintext');  
    print('Key: $key');  
    print('Ciphertext: $ciphertext');  
  
    String decrypted = OTP.decrypt(ciphertext, key);  
    print('\nDekripsi:');  
    print('Ciphertext: $ciphertext');  
    print('Key: $key');
```

```
    print('Decrypted text: $decrypted');  
}
```

Fungsi main mendemonstrasikan penggunaan kelas OTP:

1. Menetapkan plaintext
2. Membersihkan dan memformat plaintext
3. Menghasilkan kunci
4. Melakukan enkripsi
5. Mencetak hasil enkripsi
6. Melakukan dekripsi
7. Mencetak hasil dekripsi

**Kesimpulan:**

Program ini menunjukkan bagaimana OTP bekerja dengan mengenkripsi pesan "JEHIAN" menggunakan kunci acak, lalu mendekripsinya kembali untuk memverifikasi bahwa proses tersebut berhasil.