TUGAS PRAKTIK PERTEMUAN 2

PRAKTIK FLUTTER DI VS CODE

PEMROGRAMAN MOBILIE

REVALINA KRISTANTI PUTRI

244107060093

KELAS 2D

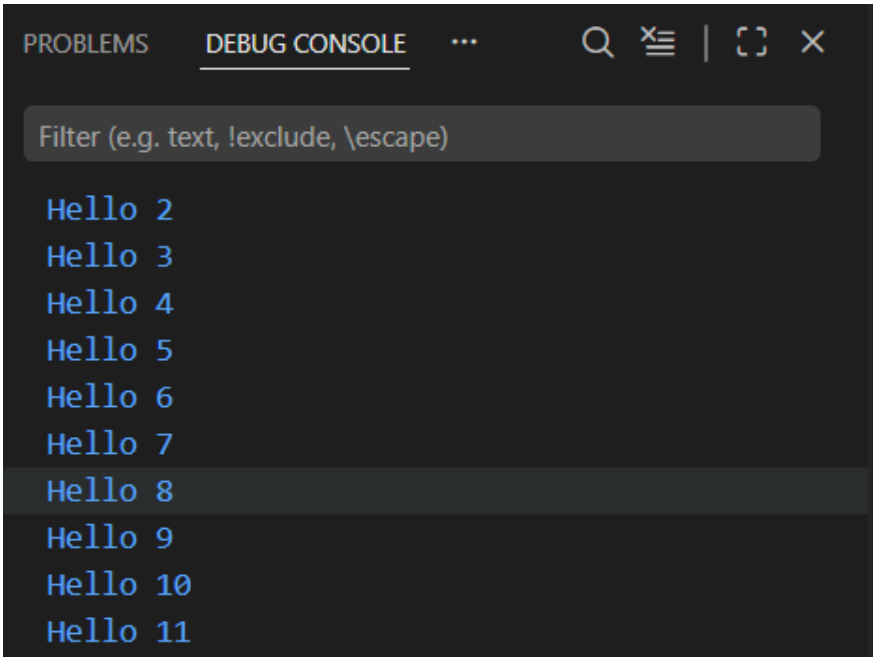
PROGRAM STUDI SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

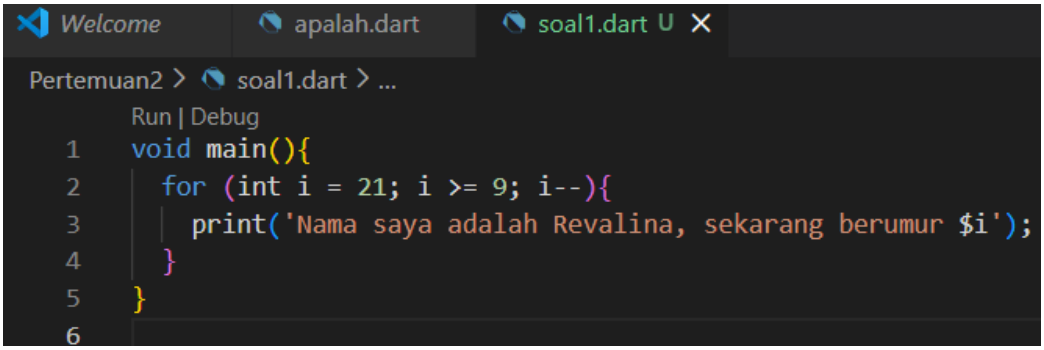
POLITEKNIK NEGERI MALANG

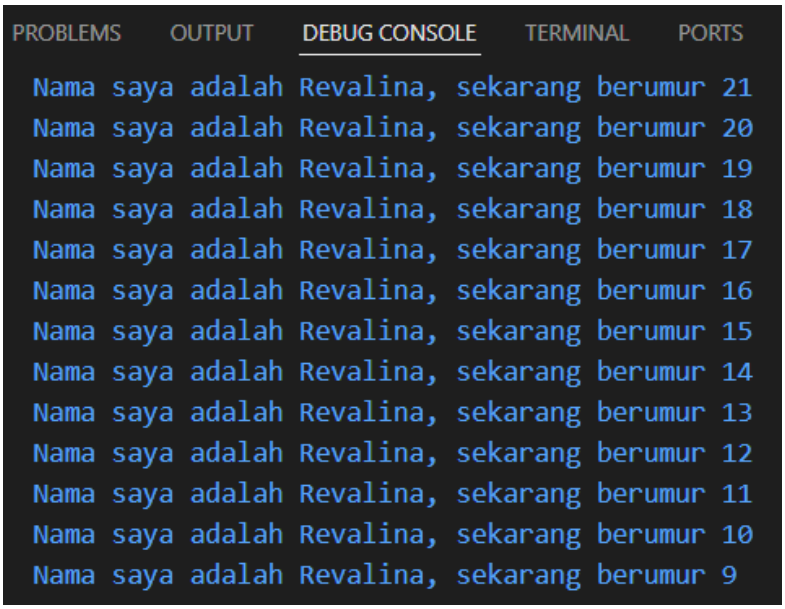
2026

1. Modifikasilah kode pada baris 3 di VS Code atau Editor Code favorit Anda berikut ini agar mendapatkan keluaran (*output*) sesuai yang diminta!



Output yang diminta (gantilah Fulan dengan nama Anda):





2. Mengapa sangat penting untuk memahami bahasa pemrograman Dart sebelum kita menggunakan framework Flutter? Jelaskan!

Jawab:

Sangat penting karena Dart adalah dasar dari Flutter atau bisa dibilang bahwa Flutter dibangun menggunakan Bahasa Dart sebagai Bahasa utamanya. Semua logika aplikasi nya, seperti pembuatan variabel, percabangan, perulangan, hingga pengelolaan data ditulis menggunakan sintaks Dart. Selain itu, Flutter juga menerapkan konsep Object-Oriented Programming (OOP) yang terdapat dalam Dart, seperti class, object, dan method, yang digunakan dalam pembuatan widget dan struktur aplikasi. Pemahaman terhadap Dart juga membantu dalam proses debugging ketika terjadi error pada aplikasi, karena sebagian besar kesalahan berasal dari logika kode Dart. Dengan memahi dasar-dasar Dart ini lah yang menjadikan pengembang dapat lebih mudah memahami cara kerja Flutter dan mengembangkan aplikasi secara lebih efektif dan terstruktur.

3. Rangkumlah materi dari codelab ini menjadi poin-poin penting yang dapat Anda gunakan untuk membantu proses pengembangan aplikasi mobile menggunakan framework Flutter.

Jawab:

1. Cara kerja dart (Dart Execution Model)

   Dart dapat dijalankan melalui dua cara, yaitu menggunakan Dart Virtual Machine (VM) dan dikompilasi menjadi JavaScript (darts2js). Dart mendukung dua mode kompilasi, yaitu *Just-In-Time (JIT)* untuk proses pengembangan dan *Ahead-Of-Time* (AOT) untuk performa optimal saat aplikasi dirilis. Pemahaman ini penting dalam Flutter karena berhubungan dengan performa aplikasi dan fitur seperti *hot reload*.
2. Hot Reload

   Fitur hot reload ini memungkinkan pengembangan melihat perubahan kode secara langsung tanpa harus me-restart aplikasi sepenuhnya. Fitur ini mempercepat proses pengembangan dna debugging pada Flutter.
3. Struktur Dasar Bahasa Dart

   Dart memiliki sintaks yang mirip dengan bahasa seperti Java dan JavaScript sehingga relative mudah dipelajari. Dart mendukung tipe data bawaan, control alur program(if-else,loop) serta fungsi yang digunakan untuk membangun logika aplikasi Flutter.
4. Konsep Object -Oriented Programming (OOP)

   Dart dirancang berbasis object-oriented (OO) dimana Bahasa OOP didasarkan pada konsep objek yang menyimpan kedua data (fields) dan kode (methods) dan objek-objek ini dibuat dari cetak biru yang disebut class yang mendefinisikan field dan method yang akan dimiliki oleh sebuah objek, dan dengan sesuai prinsip OO memastikan bahwa Dart memiliki konsep seperti class, object, method, encapsulation, inheritance, abstraction, dan

polymorphism. Dalam Flutter, widget dibuat menggunakan konsep class sehingga pemahaman OOP sangat penting.

5. Operator dalam Dart

   Dart menyidiakan berbagai operator yang sering digunakan dalam pengembangan aplikasi, antara lain:

   - Operator aritmatika (+, -, *, /, %, ~/)

     Dart hadir dengan banyak operator typical yang bekerja seperti banyak bahsa pemrograman lainnya, yaitu sebagai berikut:

     - (+) untuk tambahan.
     - (-) untuk pengurangan.
     - (*) untuk perkalian.
     - (/) untuk pembagian.
     - (-) untuk pembagian bilangan bulat. Di dart, setiap pembagian sederhana dengan / menghasilkan nilai double. Untuk mendapatkan nilai bilangan bulat, perlu membuat semacam transformasi (yaitu, dengan typecast) dalam Bahasa pemrograman lain; namun di dart sudah mendukung untuk operasi ini.
     - (%) untuk operasi modulus (sisa bagi dari bilangan bulat).
     - (-expression) untuk negasi (yang membalikkan suatu nilai)

     Beberapa operator memiliki perilaku yang berbeda tergantung pada jenis operan di sisi kiri nya, misalkan operator + dapat digunakan untuk menjumlahkan variabel dari tipe num, tetapi juga dapat digunakan untuk menggabungkan string. Karena method yang dirujuk diimplementasikan secara berbeda pada kelas yang berbeda.

   - Operator increment dan decrement (++ dan --)

     Operator penambahan dan pengurangan juga merupakan operator umum dan diimplementasikan pada angka, sebagai berikut:

- (++var) atau (var++) untuk menambahkan nila variabel sebesar 1.
- (--var) atau (var--) untuk mengurangi nilai variabel sebesar 1.

Operator dart increment dan decrement berperilaku mirip dengan Bahasa lain. Dan penerapannya sangat baik untuk operasi perhitungan pada perulangan.

- Operator equality dan relational (==, !=, >, >, >=, <=)

  Persamaan operasi dart dijelaskan sebagai berikut:
  - (= =) untuk memeriksa apakah operan sama.
  - (! =) untuk memeriksa apakah operan berbeda.

  Untuk melakukan pengujian relasional, maka gunakan operator sebagai berikut:
  - (>) memeriksa apakah operan kiri lebih besar dari operan kanan.
  - (<) memeriksa apakah operan kiri lebih kecil dari opesan kanan
  - (> =) memeriksa apakah operan kiri lebih besar dari atau sama dengan operan kanan.
  - (< =) memeriksa apakah operan kiri kurang dari atau sama dengan operan kanan.

  Di Dart, tidak seperti Java dan Bahasa lainnya, operator (= =) tidak membadingkan referensi/Alamat memori melainkan isi dari variabel tersebut.

- Operator logical (!, &&, ||)

  Operator logika di Dart adlah operator yang diterapkan pada operan bool; bisa berupa variabel, ekspresi, atau kondisi. Nah selain itu, dapat dikombinasikan dengan ekspresi kompleks dengan menggabungkan nilai ekspresi yang dievaluasi. Operator logika yang disediakan adalah sebagai berikut:

- (!expression) negasi atau kebalikan hasil ekspresi yaitu, true menjasi false, dan false menjadi true
- (||) menerapkan operasi logika OR antara dua ekspresi
- (&&) menerapkan operasi logika AND antara dua ekspresi.

Operator-operator ini digunakan untuk pengelolaan data, validasi kondisi, dan pengambilan Keputusan dalam aplikasi Flutter.

6. Type Safety pada Dart

   Dart memiliki system type safety yang membantu mencegah kesalahan tipe data saat kompilasi. Hal ini mmebuat kode lebih aman, terstruktur, dan mengurangi bug saat pengembangan aplikasi.

7. Runtime System dan garbage Collection

   Dart memiliki runtime system serta garbage collector yang membantu pengelolaan memori secara otomatis. Ini sangat penting dalam aplikasi mobile agar penggunaan memori tetap efisie.

4. Buatlah penjelasan dan contoh eksekusi kode tentang perbedaan *Null Safety* dan *Late variabel*!

Jawab:

Perbedaan Null Safety dan Late Variable pada Dart

a. Null Safety

   Null Safety adalah fitur pada Dart yang bertujuan untuk mencegah kesalahan akibat variabel yang berinalai null secara tidak sengaja. Variabel secara default **tidak boleh bernilai null**, kecuali secara eksplisit ditandai dengan tanda tanya (?). dengan adanya Null Safety ini kesalahan seperti *null reference error* dapat dicegah sejak tahap kompilasi.

   Contoh kode null safety

```
Pertemuan2 > soal4.dart > main
      Run | Debug
1     void main() {
2       String nama = "Revalina";
3       print(nama);
4
5       String? alamat;
6       print(alamat);
7     }
```

Output:

```
PROBLEMS    OUTPUT    DEBUG C

Revalina
null

Exited.
```

Contoh ketika tidak diberi nilai:

```
Pertemuan2 > soal4.dart > main
      Run | Debug
1     void main() {
2       String? nama;
3       print(nama);
4     }
```

Kode tersebut akan menghasilkan error karena variabel non-nullable belum diberi nilai.

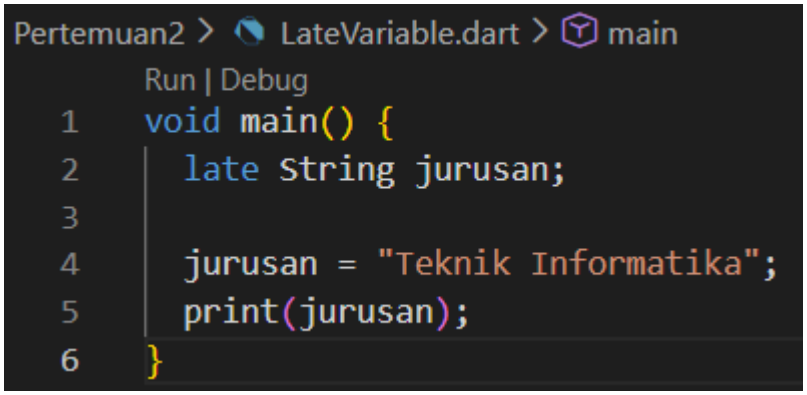
```
PROBLEMS    OUTPUT    DEE

null

Exited.
```

b. Late Variable

Late digunakan ketika kita ingin mendeklarasi variabel non-nullable, tetapinilainya akan diinisialisasi sebelum digunakan, dengan kata lain late ini unutk menunda insialisasi variabel. Fitur ini berguna ketika nilai variabel belum tersedia di awal, tetapinpasti akan siisi sebelum dipakai.



Output: