Nama : Febrian Dani Ritonga

Kelas : 3H

NIM: 2141720070

Soal 1

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

```
1 void main() {
2   for (int i = 0; i < 10; i++) {
3     print('hello ${i + 2}');
4   }
5 }
```

Output yang diminta:

```
Console

  Nama saya adalah Fulan, sekarang berumur 18
  Nama saya adalah Fulan, sekarang berumur 17
  Nama saya adalah Fulan, sekarang berumur 16
  Nama saya adalah Fulan, sekarang berumur 15
  Nama saya adalah Fulan, sekarang berumur 14
  Nama saya adalah Fulan, sekarang berumur 13
  Nama saya adalah Fulan, sekarang berumur 12
  Nama saya adalah Fulan, sekarang berumur 11
  Nama saya adalah Fulan, sekarang berumur 10
  Nama saya adalah Fulan, sekarang berumur 9
```

Jawab :

Kode :

```
void main(){
  for (int i = 0; i<20; i++){
  print('Nama saya dani, sekarang saya berumur ${20-i}');
  }
}
```

Hasil run :

```
[Running] dart "c:\Semester 5\2141720070-mo
Nama saya dani, sekarang saya berumur 20
Nama saya dani, sekarang saya berumur 19
Nama saya dani, sekarang saya berumur 18
Nama saya dani, sekarang saya berumur 17
Nama saya dani, sekarang saya berumur 16
Nama saya dani, sekarang saya berumur 15
Nama saya dani, sekarang saya berumur 14
Nama saya dani, sekarang saya berumur 13
Nama saya dani, sekarang saya berumur 12
Nama saya dani, sekarang saya berumur 11
Nama saya dani, sekarang saya berumur 10
Nama saya dani, sekarang saya berumur 9
Nama saya dani, sekarang saya berumur 8
Nama saya dani, sekarang saya berumur 7
Nama saya dani, sekarang saya berumur 6
Nama saya dani, sekarang saya berumur 5
Nama saya dani, sekarang saya berumur 4
Nama saya dani, sekarang saya berumur 3
Nama saya dani, sekarang saya berumur 2
Nama saya dani, sekarang saya berumur 1

[Done] exited with code=0 in 0.87 seconds
```

Soal 2

Jawab :

Berikut beberapa alasan mengapa pemahaman Dart penting sebelum mulai menggunakan Flutter:

- Bahasa Dasar: Dart adalah bahasa pemrograman dasar yang digunakan dalam pengembangan Flutter. Anda perlu memahami sintaksis, struktur, dan konsep dasar Dart untuk memahami bagaimana mengembangkan aplikasi Flutter.
- Pemrograman Reaktif: Flutter adalah framework pemrograman reaktif yang mengharuskan Anda memahami konsep pemrograman reaktif, yang melibatkan perubahan state dan pembaruan antarmuka pengguna. Untuk memahami bagaimana mengimplementasikan pemrograman reaktif dalam Flutter, Anda perlu memahami konsep seperti setState, Stream, dan Future, yang merupakan bagian integral dari Dart.
- Pustaka Dart: Flutter menggunakan banyak pustaka (library) Dart yang sudah ada untuk mengakses berbagai fitur dan sumber daya perangkat. Anda perlu memahami bagaimana menggunakan pustaka-pustaka ini dalam Dart.
- Membuat Fungsi dan Kelas Khusus: Dalam Flutter, Anda akan sering membuat fungsi dan kelas khusus untuk mengorganisasi kode Anda. Ini melibatkan pemahaman konsep seperti pewarisan, enkapsulasi, dan polimorfisme, yang merupakan bagian dari pemrograman berorientasi objek dalam Dart.
- Pengoptimalan Kode: Memahami Dart akan membantu Anda menulis kode yang lebih efisien dan mudah dibaca. Anda dapat mengoptimalkan kode Anda dengan lebih baik jika Anda mengerti bahasa dasar di mana Flutter dibangun.

Soal 3

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

Jawab :

Dart adalah bahasa pemrograman yang dirancang untuk menggabungkan kelebihan-kelebihan dari sebagian besar bahasa tingkat tinggi dengan fitur-fitur bahasa pemrograman terkini. Beberapa fitur kunci Dart meliputi:

- Productive Tooling: Dart menyertakan kakas (tool) yang memudahkan analisis kode, plugin IDE, dan ekosistem paket yang besar, yang semuanya berkontribusi pada produktivitas pengembang.
- Garbage Collection: Dart memiliki mekanisme garbage collection yang mengelola dealokasi memori, membantu menghindari masalah kebocoran memori.
- Type Annotations (Opsional): Dart memungkinkan penggunaan anotasi tipe (type annotations) secara opsional untuk meningkatkan keamanan dan konsistensi dalam mengontrol data dalam aplikasi.
- Statically Typed: Meskipun type annotations bersifat opsional, Dart tetap aman karena menggunakan fitur type-safe dan type inference untuk menganalisis tipe data saat runtime. Ini membantu mendeteksi bug selama kompilasi kode.
- Portability: Dart tidak terbatas pada web, karena dapat diterjemahkan ke JavaScript, dan juga dapat dikompilasi secara native ke kode untuk platform seperti ARM dan x86.

Dalam konteks pengembangan dengan framework Flutter, pemahaman yang kuat tentang Dart menjadi kunci. Semua aspek pengembangan Flutter, termasuk kode aplikasi, kode plugin, dan manajemen dependensi, menggunakan Dart dan fitur-fiturnya. Memahami Dart dengan baik akan membuat pengembang lebih produktif dalam penggunaan Flutter dan merasa nyaman dalam mengembangkan aplikasi Flutter. Dart adalah dasar penting untuk menjadi pengembang Flutter yang sukses.

Object orientation

Seperti kebanyakan bahasa modern, Dart dirancang untuk object-oriented (OO). Secara singkat, Bahasa OOP didasarkan pada konsep objek yang menyimpan kedua data (disebut fields) dan kode (disebut methods). Objek-objek ini dibuat dari cetak biru yang disebut class yang mendefinisikan field dan method yang akan dimiliki oleh sebuah objek.

Sesuai prinsip OOP memastikan bahwa Dart memiliki fitur encapsulation, inheritance, composition, abstraction, dan polymorphism. Kita akan mempelajari kelas Dart lebih banyak lagi di pertemuan dengan topik Class Dart dan Construct, namun sudah cukup jika Anda telah belajar konsep OO dalam bahasa lain seperti Java, maka sebagian besar desain OO pada Dart akan sangat mirip.

Dart operators

Di Dart, operator tidak lebih dari method yang didefinisikan dalam class dengan sintaks khusus.Jadi, ketika Anda menggunakan operator seperti x == y, seolah-olah Anda sedang memanggil x.==(y) metode untuk melakukan perbandingan kesetaraan.Seperti yang mungkin telah Anda catat, kita menggunakan method pada x. Untuk semua tipe data, tidak seperti bahasa

Java yang memiliki data primitif, x selalu berupa turunan dari kelas yang memiliki method. Ini berarti bahwa operator dapat diganti sesuai logika yang Anda inginkan.

Arithmetic operators

Dart hadir dengan banyak operator typical yang bekerja seperti banyak bahasa 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, Anda perlu membuat semacam transformasi (yaitu, typecast) dalam bahasa pemrograman lain; namun 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; Misalnya, 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.Dart juga menyediakan shortcut operator untuk menggabungkan variabel setelah operasi lainnya. Operator aritmatika atau shortcut operator adalah +=, -=, *=, /=, dan ~/=.

Increment and decrement operators

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

++var atau var++ untuk menambah nilai variabel var sebesar 1

--var atau var-- untuk mengurangi nilai variabel var sebesar 1

Operator Dart increment dan decrement berperilaku mirip dengan bahasa lain. Penerapan operator increment dan decrement sangat baik untuk operasi perhitungan pada perulangan.

Equality and relational operators

Persamaan operator 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 operan 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 membandingkan referensi/alamat memori melainkan isi dari variabel tersebut.

Juga, tidak seperti JavaScript, tidak ada operator === yang diperlukan pada Dart karena telah memiliki fitur type safety yang memastikan bahwa operator persamaan == hanya digunakan pada objek dengan tipe yang sama.

Logical operators

Operator logika di Dart adalah operator yang diterapkan pada operan bool; bisa berupa variabel, ekspresi, atau kondisi. 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 menjadi false dan false menjadi true.

|| menerapkan operasi logika OR antara dua ekspresi.

&& menerapkan operasi logika AND antara dua ekspresi.

Soal 4

Buatlah slide yang berisi penjelasan dan contoh eksekusi kode tentang perbedaan Null Safety dan Late variabel ! **(Khusus soal ini kelompok berupa link google slide)**

: https://www.canva.com/design/DAFtczalxJI/DGjJMwe9mOZNWyw-RiMKGQ/edit?utm_content=DAFtczalxJI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton