

LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK



JUDUL :
FUNDAMENTAL DART

Disusun oleh:
Diva Zahra Berliani (21102103)

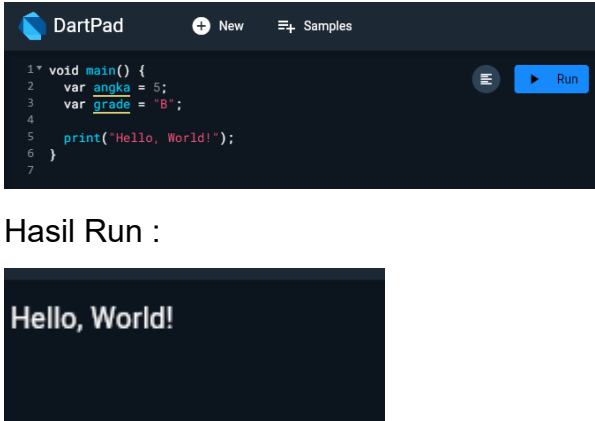
TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
BANYUMAS, JAWA TENGAH
2024

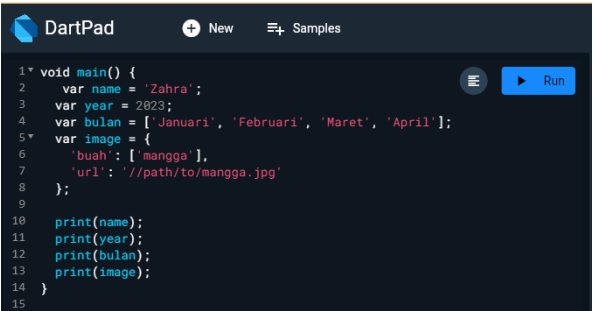
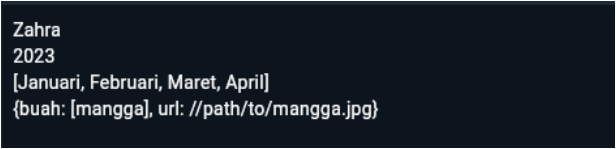
Pembahasan

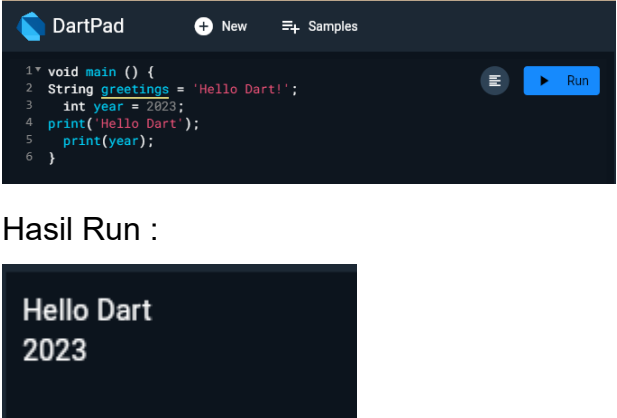
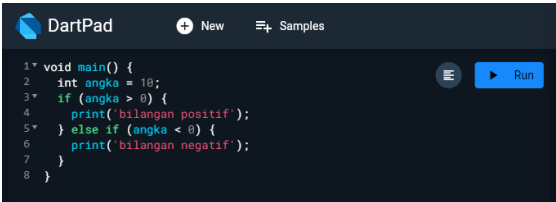
Mata kuliah Pemrograman Perangkat Bergerak (PPB) disini mengajarkan kami agar dapat memahami pemrograman perangkat bergerak, mampu membuat aplikasi berbasis perangkat bergerak, memahami dan mengimplementasikan konsep integrasi program perangkat bergerak dengan layanan berbasis daring.

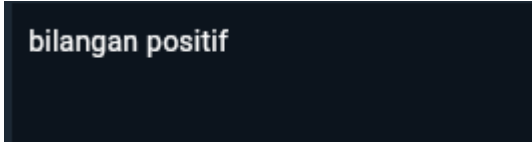
Pada pertemuan 2 ini, mata kuliah PPB membahas tentang pengenalan Dart. Dart sendiri merupakan bahasa pemrograman yang fleksibel, open source dan general purpose yang dimana dart sendiri dikembangkan oleh Google. Dart bisa berjalan di mana pun baik itu Android, IOS, maupun web. Dan disini kami melakukan sebuah implementasi / praktikum tentang modul Fundamental Dart.

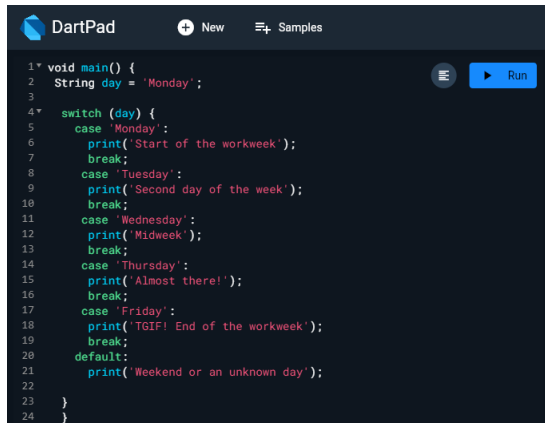
Langkah-Langkah Praktikum

Langkah Praktikum	Pembahasan
<p>Gambar Langkah 1</p>  <p>The screenshot shows the DartPad web interface. At the top, there's a header with the DartPad logo, a '+ New' button, and a 'Samples' menu. Below the header is a code editor with the following Dart code:</p> <pre>1 void main() { 2 var angka = 5; 3 var grade = "B"; 4 5 print("Hello, World!"); 6 } 7</pre> <p>There is a 'Run' button to the right of the code editor. Below the code editor, the output of the program is displayed in a dark box: 'Hello, World!'.</p>	<p>Pembahasan dari Gambar Langkah 1</p> <p>var angka = 5;; Deklarasi variabel angka dengan nilai 5. Dart akan secara otomatis menentukan tipe data variabel berdasarkan nilai yang diberikan, dalam hal ini, tipe data variabel angka akan menjadi int karena nilainya adalah bilangan bulat. var grade = "B"; Deklarasi variabel grade dengan nilai "B". Dart akan menentukan tipe data variabel grade sebagai String</p>

	<p>karena nilainya adalah sebuah string. <code>print("Hello, World!");</code> Mencetak string "Hello, World!" ke konsol. Ini adalah contoh penggunaan fungsi <code>print()</code> untuk menampilkan output pada konsol.</p>
<p>Gambar Langkah 2</p>  <p>Hasil Run:</p> 	<p>Pembahasan dari Gambar Langkah 2 :</p> <p><code>var name = 'Zahra';</code> Deklarasi variabel <code>name</code> dengan nilai string <code>'Zahra'</code>. <code>var year = 2023;</code> Deklarasi variabel <code>year</code> dengan nilai numerik <code>2023</code>. <code>var bulan = ['Januari', 'Februari', 'Maret', 'April'];</code> Deklarasi variabel <code>bulan</code> dengan nilai array yang berisi empat string: <code>'Januari'</code>, <code>'Februari'</code>, <code>'Maret'</code>, dan <code>'April'</code>. <code>var image = {'buah': ['mangga'], 'url': '//path/to/mangga.jpg'};</code> Deklarasi variabel <code>image</code> dengan nilai objek map yang memiliki dua properti, yaitu <code>'buah'</code> yang memiliki nilai array <code>['mangga']</code>, dan <code>'url'</code> yang memiliki nilai string <code>'//path/to/mangga.jpg'</code>. <code>print(name);</code> Mencetak nilai</p>

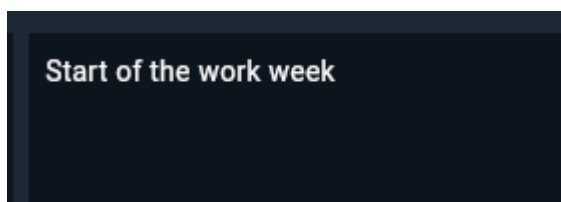
	<p>variabel name ke konsol. print(year);: Mencetak nilai variabel year ke konsol. print(bulan);: Mencetak nilai variabel bulan (array) ke konsol. print(image);: Mencetak nilai variabel image (objek map) ke konsol.</p>
<p>Gambar Langkah 3</p>  <p>Hasil Run :</p>	<p>Pembahasan dari Gambar Langkah 3</p> <p>String greetings = 'Hello Dart!'; Deklarasi variabel greetings sebagai string dengan nilai 'Hello Dart!'. Variabel ini tidak digunakan dalam program. int year = 2023;; Deklarasi variabel year sebagai integer dengan nilai 2023. print('Hello Dart');: Mencetak string 'Hello Dart' ke konsol. Ini adalah contoh penggunaan fungsi print() untuk menampilkan output pada konsol. print(year);: Mencetak nilai variabel year ke konsol.</p>
<p>Gambar a</p> 	<p>Pembahasan dari gambar a, menggunakan control flow dengan if dan else.</p> <ul style="list-style-type: none"> • Pertama mendeklarasikan

<p>Hasil run :</p> 	<p>nilai 10 ke dalam variabel angka (tipe integer) terlebih dahulu. Kemudian pada baris <code>if (angka > 0)</code> mengevaluasi apakah nilai variabel angka lebih besar dari 0. Jika kondisi tersebut benar (nilai angka positif)</p> <ul style="list-style-type: none"> • Jika kondisi tersebut salah (nilai angka negatif), maka program akan mengevaluasi kondisi pada blok <code>else if</code>. Pada blok <code>else if (angka < 0)</code> mengevaluasi apakah nilai variabel angka kurang dari 0. • Jika kondisi ini benar (nilai angka negatif), maka program akan mencetak teks "bilangan negatif". Pada gambar a menghasilkan bilangan positif karena nilai angka = 10.
<p>Gambar b</p>	<p>Pada gambar b, menggunakan <code>switch case</code>. Pertama, mendeklarasikan variabel <code>day</code> dengan</p>

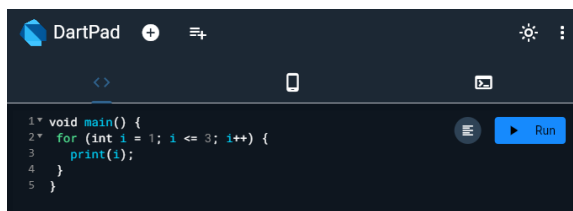


```
1* void main() {
2   String day = 'Monday';
3
4*  switch (day) {
5     case 'Monday':
6       print('Start of the workweek');
7       break;
8     case 'Tuesday':
9       print('Second day of the week');
10      break;
11     case 'Wednesday':
12       print('Midweek');
13       break;
14     case 'Thursday':
15       print('Almost there!');
16       break;
17     case 'Friday':
18       print('TGIF! End of the workweek');
19       break;
20     default:
21       print('Weekend or an unknown day');
22   }
23 }
24 }
```

Hasil run :



Gambar c

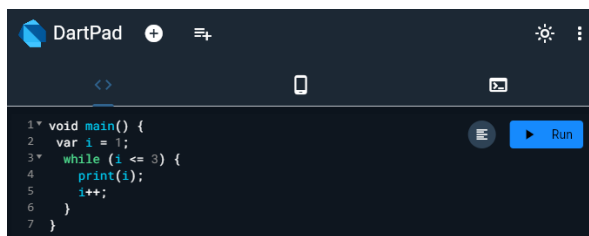


```
1* void main() {
2*  for (int i = 1; i <= 3; i++) {
3    print(i);
4  }
5 }
```

Hasil run :



Gambar d (while)



```
1* void main() {
2   var i = 1;
3*  while (i <= 3) {
4    print(i);
5    i++;
6  }
7 }
```

nilai "Monday". Kemudian, menggunakan switch statement untuk mengevaluasi nilai day. Switch (day) {...}: blok switch statement yang mengevaluasi nilai day. Setiap case adalah pola (pattern) yang akan dibandingkan dengan nilai day. Jika nilai day cocok dengan pola pada case, maka blok kode yang sesuai akan dieksekusi. Pada gambar ini, mencetak "Start of the work week".

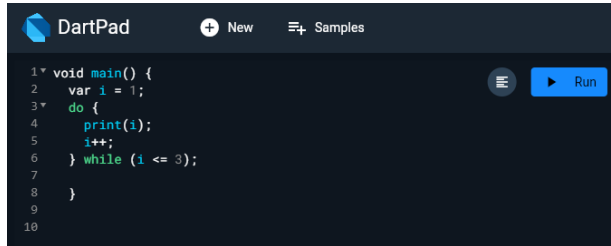
Pada gambar c, menggunakan control flow dengan for loops. Pertama, Variabel i dideklarasikan sebagai integer dengan nilai awal 1. Kemudian perulangan for akan berjalan selama i kurang dari atau sama dengan 3. Dan Setiap iterasi, nilai i akan bertambah 1 (i++).

Pada gambar d, menggunakan control flow dengan while dan do-while. Hasil eksekusi dari kedua program ini akan sama, yaitu

Hasil run :

```
1
2
3
```

Gambar d (do-while)



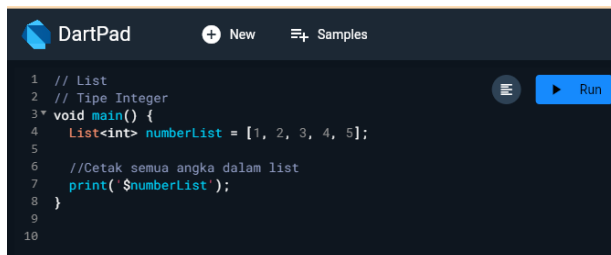
```
1 void main() {
2   var i = 1;
3   do {
4     print(i);
5     i++;
6   } while (i <= 3);
7 }
8
9
10
```

Hasil run :

```
1
2
3
```

mencetak angka dari 1 hingga 10. Perbedaannya terletak pada urutan evaluasi kondisi: pada while, kondisi dievaluasi sebelum perulangan, sedangkan pada do-while, kondisi dievaluasi setelah minimal satu iterasi perulangan.

Gambar 4

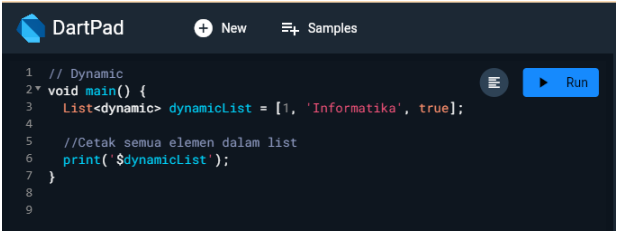
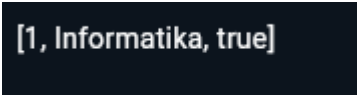
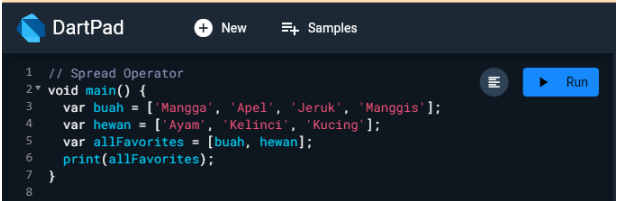
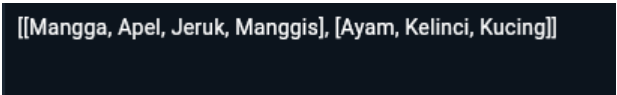
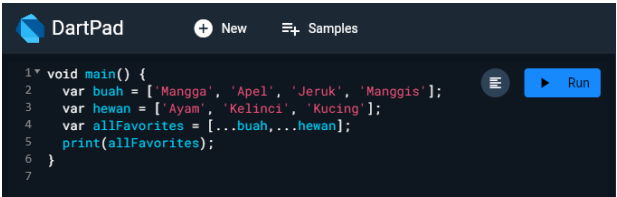


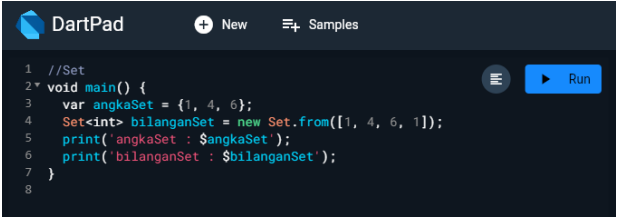
```
1 // List
2 // Tipe Integer
3 void main() {
4   List<int> numberList = [1, 2, 3, 4, 5];
5
6   //Cetak semua angka dalam list
7   print('$numberList');
8 }
9
10
```

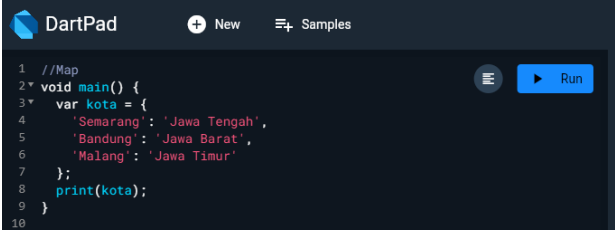
Hasil run :

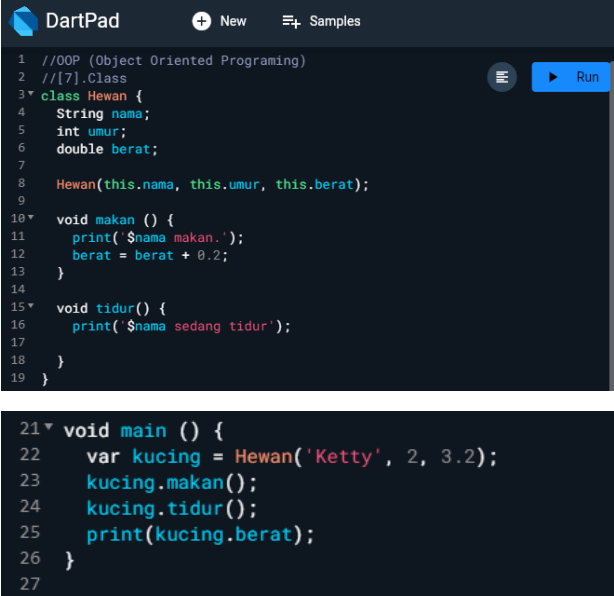
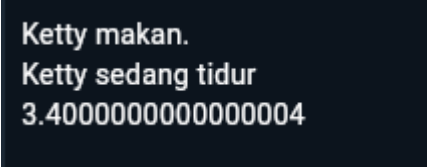
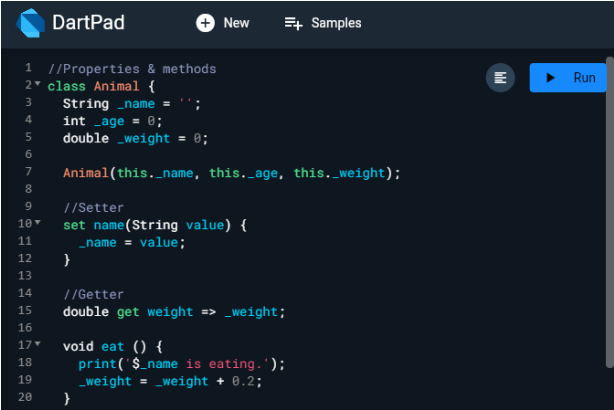
```
[1, 2, 3, 4, 5]
```

Gambar 4 dan 5, menggunakan konsep list. List sendiri adalah tipe data yang digunakan untuk merepresentasikan koleksi nilai yang terurut. List dapat berisi elemen-elemen dengan tipe data yang sama atau berbeda. List dapat diinisialisasi dengan nilai-nilai yang diberikan, atau dapat dibuat kosong dan kemudian

	<p>diisi kemudian. Sebagai contoh: Pada gambar 4, merupakan contoh satu objek List yang berisi kumpulan data bertipe integer.</p>
<p>Gambar 5</p>  <pre> 1 // Dynamic 2 void main() { 3 List<dynamic> dynamicList = [1, 'Informatika', true]; 4 5 //Cetak semua elemen dalam list 6 print('\$dynamicList'); 7 } 8 9 </pre> <p>Hasil run :</p> 	<p>Pada gambar 5, Jika kita tidak mendefinisikan nilai secara eksplisit ke dalam List, maka List akan menyimpan tipe dynamic.</p>
<p>Gambar 6</p>  <pre> 1 // Spread Operator 2 void main() { 3 var buah = ['Mangga', 'Apel', 'Jeruk', 'Manggis']; 4 var hewan = ['Ayam', 'Kelinci', 'Kucing']; 5 var allFavorites = [...buah, ...hewan]; 6 print(allFavorites); 7 } 8 </pre> <p>Hasil run :</p> 	<p>Pada gambar 6, program tidak menggunakan spread operator. Nilai list tidak tergabung, variabel allFavorites menjadi List yang menampung 2 list di dalamnya.</p>
<p>Gambar 7</p>  <pre> 1 void main() { 2 var buah = ['Mangga', 'Apel', 'Jeruk', 'Manggis']; 3 var hewan = ['Ayam', 'Kelinci', 'Kucing']; 4 var allFavorites = [...buah, ...hewan]; 5 print(allFavorites); 6 } 7 </pre>	<p>Pada gambar 7, kita menggunakan spread operator untuk menyisipkan elemen dari buah dan hewan ke allFavorites. Dengan spread operator(...) variabel hewan</p>

<p>Hasil run :</p> <pre>[Mangga, Apel, Jeruk, Manggis, Ayam, Kelinci, Kucing]</pre>	<p>dan buah dapat menjadi 1 List</p>
<p>Gambar 8</p>  <p>Hasil run :</p> <pre>angkaSet : {1, 4, 6} bilanganSet : {1, 4, 6}</pre>	<p>Pada gambar 8, menggunakan set (sebuah collection yang hanya dapat menyimpan nilai yang unik dan tidak boleh ada nilai duplikat). Pada contoh, Meskipun kita memasukkan elemen 4 dan 1 dua kali, Set hanya menyimpan satu salinan dari setiap elemen. Ini menunjukkan sifat unik dari Set.</p> <ul style="list-style-type: none"> • kita membuat dua Set: angkaSet dan bilanganSet, yang berisi beberapa bilangan bulat. Kita bisa menggunakan kurung kurawal ({}) atau konstruktor Set.from() untuk membuat Set dari daftar atau iterable lainnya. • Ketika mencetak angkaSet, berarti angkaSet adalah Set yang berisi tiga elemen: 1, 4, dan 6. Dan ketika mencetak bilanganSet,

	<p>berarti <code>bilanganSet</code> adalah Set yang berisi tiga elemen: 1, 4, dan 6.</p>
<p>Gambar 9</p>  <pre> 1 //Map 2 void main() { 3 var kota = { 4 'Semarang': 'Jawa Tengah', 5 'Bandung': 'Jawa Barat', 6 'Malang': 'Jawa Timur' 7 }; 8 print(kota); 9 } 10 </pre> <p>Hasil run :</p> <pre>{Semarang: Jawa Tengah, Bandung: Jawa Barat, Malang: Jawa Timur}</pre>	<p>Pada gambar 9, menggunakan map yang dimana map sendiri merupakan sebuah collection yang dapat menyampaikan data dengan format key-value.</p> <ul style="list-style-type: none"> • Pertama, membuat sebuah Map bernama kota, yang berisi beberapa kota dan provinsinya. Kita bisa menggunakan kurung kurawal (<code>{}</code>) atau konstruktor <code>Map()</code> untuk membuat Map dari daftar atau iterable lainnya. • Berarti, kota adalah Map yang berisi tiga pasangan kunci dan nilai: Semarang dan Jawa Tengah, Bandung dan Jawa Barat, Malang dan Jawa Timur.
<p>Gambar 10</p>	<p>Pada gambar 10, menggunakan class yang dimana class sendiri merupakan sebuah blueprint untuk membuat objek. Di</p>

 <pre> 1 //OOP (Object Oriented Programing) 2 //[[7]].Class 3 class Hewan { 4 String nama; 5 int umur; 6 double berat; 7 8 Hewan(this.nama, this.umur, this.berat); 9 10 void makan () { 11 print('\$nama makan. '); 12 berat = berat + 0.2; 13 } 14 15 void tidur() { 16 print('\$nama sedang tidur '); 17 } 18 } 19 21 void main () { 22 var kucing = Hewan('Ketty', 2, 3.2); 23 kucing.makan(); 24 kucing.tidur(); 25 print(kucing.berat); 26 } 27 </pre> <p>Hasil run :</p>  <pre> Ketty makan. Ketty sedang tidur 3.4000000000000004 </pre>	<p>dalam class kita perlu mendefinisikan sifat (attribute) dan perilaku (behaviour) dari objek yang akan dibuat.</p> <ul style="list-style-type: none"> • Pertama, disini kita memiliki kelas bernama Hewan dan atribut kelas Hewan meliputi nama, umur, dan berat. Metode kelas Hewan adalah makan() dan tidur(). • Membuat objek kucing dengan atribut nama: 'Ketty', umur: 2, dan berat: 2.0. nantinya Metode makan() menambahkan berat kucing dan metode tidur() mencetak pesan bahwa kucing sedang tidur.
<p>Gambar 11</p>  <pre> 1 //Properties & methods 2 class Animal { 3 String _name = ''; 4 int _age = 0; 5 double _weight = 0; 6 7 Animal(this._name, this._age, this._weight); 8 9 //Setter 10 set name(String value) { 11 _name = value; 12 } 13 14 //Getter 15 double get weight => _weight; 16 17 void eat () { 18 print('\$_name is eating. '); 19 _weight = _weight + 0.2; 20 } 21 </pre>	<p>Pada gambar 11, menggambarkan konsep OOP (properties & methods).</p> <ul style="list-style-type: none"> • Pertama, pada kelas ini memiliki tiga atribut: <code>_name</code>, <code>_age</code>, dan <code>_weight</code>. Atribut ini diinisialisasi melalui konstruktor <code>Animal</code>. Lalu terdapat setter untuk mengatur

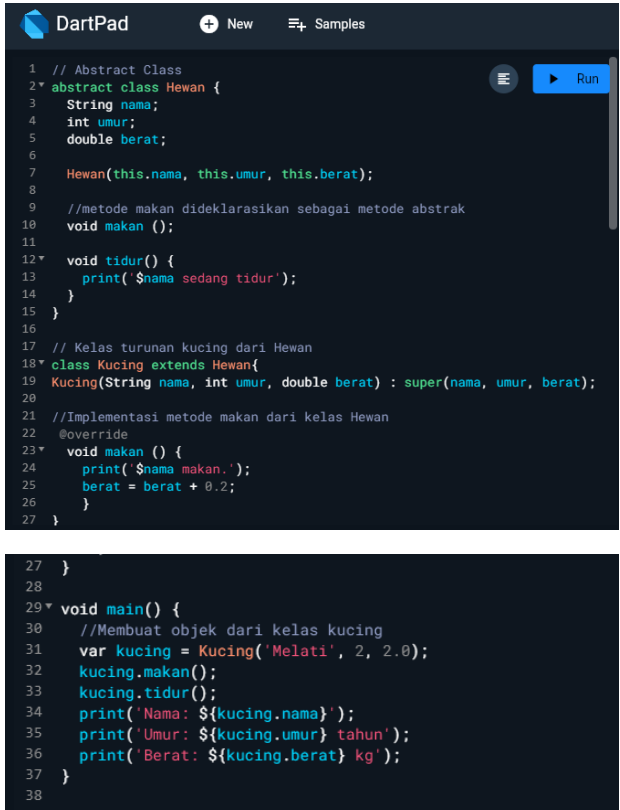
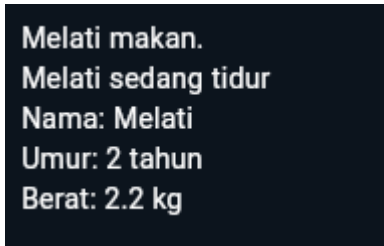
<pre> 23 void sleep () { 24 print('\$_name is sleeping.');</pre> <pre> 25 } 26 void poop () { 27 print('\$_name is pooping.');</pre> <pre> 28 _weight = _weight - 0.1; 29 } 30 } 31 32 void main () { 33 var myPet = Animal('Fluffy', 2, 2.5); 34 myPet.eat(); 35 myPet.sleep(); 36 myPet.poop(); 37 38 print('Name: \${myPet._name}'); 39 print('Age: \${myPet._age} years'); 40 print('Weight: \${myPet._weight}kg'); 41 }</pre> <p>Hasil run :</p> <pre> Fluffy is eating. Fluffy is sleeping. Fluffy is pooping. Name: Fluffy Age: 2 years Weight: 2.6kg</pre>	<p>nilai <code>_name</code> dan getter untuk mengakses nilai <code>_weight</code>. Metode <code>eat()</code> menambahkan berat hewan setelah makan, <code>sleep()</code> mencetak pesan bahwa hewan sedang tidur, <code>poop()</code> mencetak pesan bahwa hewan sedang buang air besar dan mengurangi beratnya.</p> <ul style="list-style-type: none"> Kemudian ada objek <code>myPet</code>, objek ini dibuat dari kelas <code>Animal</code> dengan nama <code>Fluffy</code>, umur 2 tahun, dan berat 2.5 kg. Metode <code>eat()</code>, <code>sleep()</code>, dan <code>poop()</code> dipanggil pada objek ini.
<p>Gambar 12</p> <pre> 1 // [9].Inheritance 2 import 'hewan.dart'; 3 4 class Meong extends Hewan { 5 String warnaBulu; 6 7 Meong(8 String nama, 9 int umur, 10 double berat, 11 this.warnaBulu, 12) : super(nama, umur, berat); 13 14 void jalan() { 15 print('\$nama berjalan');</pre> <pre> 16 } 17 18 void warna() { 19 print('\$nama berwarna \$warnaBulu');</pre> <pre> 20 } 21 }</pre> <pre> 23 void main() { 24 var kucing = Meong('Ketty', 2, 3.2, 'Putih');</pre> <pre> 25 kucing.jalan(); 26 kucing.makan(); 27 kucing.warna(); 28 print(kucing.berat); 29 }</pre>	<p>Pada gambar 12, menggunakan konsep inheritance (kemampuan suatu program untuk membuat kelas baru dari kelas yang ada). Di dalam OOP kelas yang menurunkan sifat disebut sebagai kelas induk (parent class/superclass) sementara kelas yang mewarisi kelas induknya disebut sebagai</p>

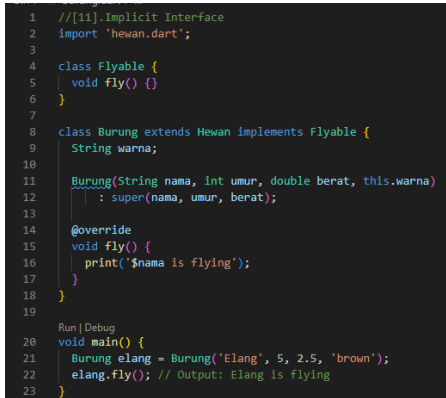
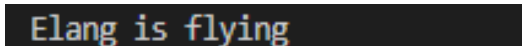
Hasil run :

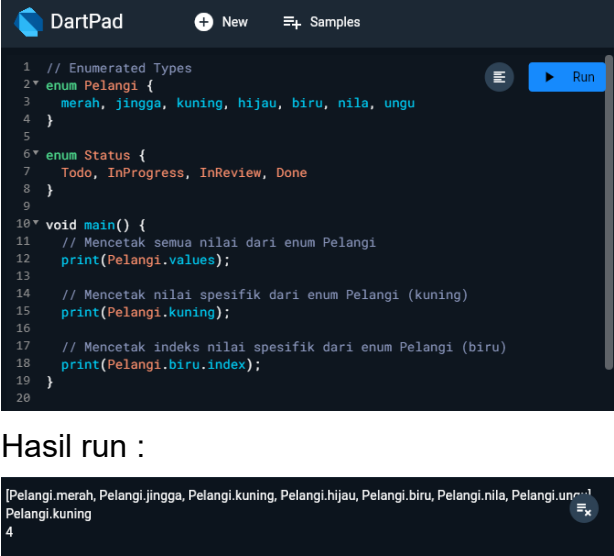
```
Ketty berjalan  
Ketty makan.  
Ketty berwarna Putih  
Nama: Ketty  
Umur: 2 tahun  
Berat: 2.2 kg
```

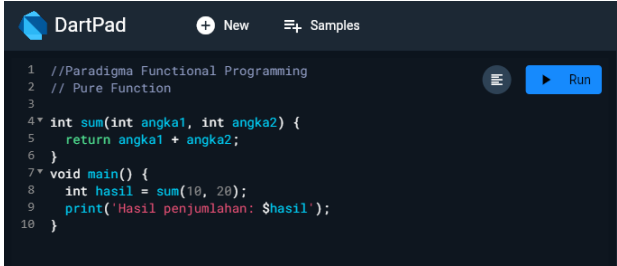
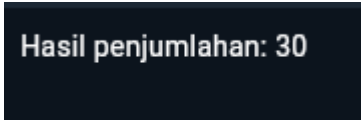
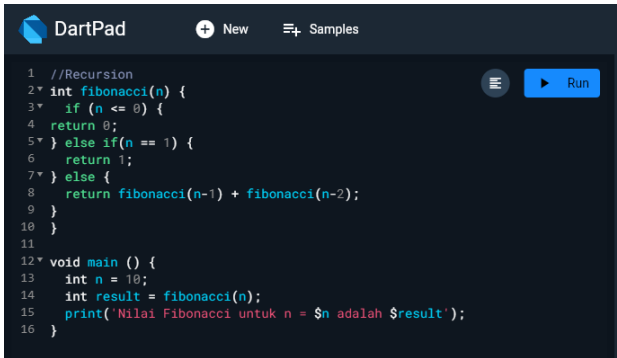
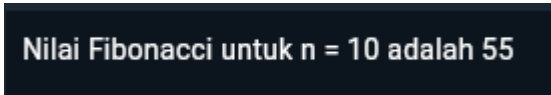
kelas anak (child/subclass).

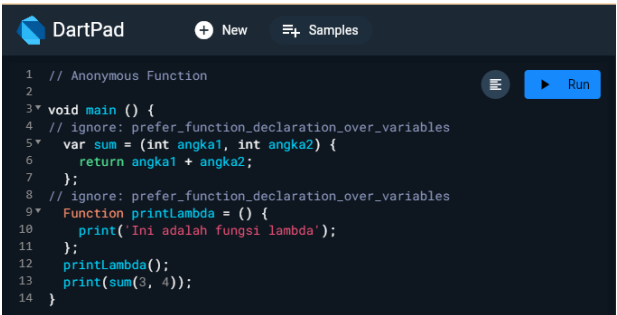
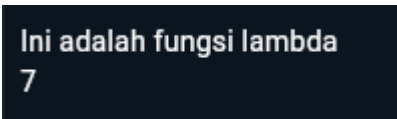
- Pertama, mengimpor file `hewan.dart` yang mungkin berisi definisi kelas `Hewan`. Kemudian mendeklarasikan Kelas `Meong` yang merupakan turunan dari kelas `Hewan`. Ini dinyatakan dengan menggunakan kata kunci `extends`. Kelas `Meong` memiliki beberapa atribut, yaitu `nama`, `umur`, `berat`, dan `warnaBulu`.
- Konstruktor `Meong` menerima beberapa parameter (`nama`, `umur`, `berat`, dan `warna bulu`) dan memanggil konstruktor kelas induk (`super(nama, umur, berat)`).
- Pada metode `jalan()`: mencetak pesan bahwa kucing dengan nama tertentu sedang berjalan, `warna()`: Metode ini mencetak pesan tentang warna bulu kucing dengan nama tertentu dan `main()`: fungsi ini memanggil


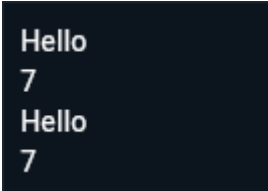
	<p>metode jalan(),makan(), dan warna() pada objek kucing, serta mencetak berat kucing.</p>
<p>Gambar 13</p>  <pre> 1 // Abstract Class 2 abstract class Hewan { 3 String nama; 4 int umur; 5 double berat; 6 7 Hewan(this.nama, this.umur, this.berat); 8 9 //metode makan dideklarasikan sebagai metode abstrak 10 void makan (); 11 12 void tidur() { 13 print('\$nama sedang tidur'); 14 } 15 } 16 17 // Kelas turunan kucing dari Hewan 18 class Kucing extends Hewan{ 19 Kucing(String nama, int umur, double berat) : super(nama, umur, berat); 20 21 //Implementasi metode makan dari kelas Hewan 22 @override 23 void makan () { 24 print('\$nama makan.'); 25 berat = berat + 0.2; 26 } 27 } 28 29 void main() { 30 //Membuat objek dari kelas kucing 31 var kucing = Kucing('Melati', 2, 2.0); 32 kucing.makan(); 33 kucing.tidur(); 34 print('Nama: \${kucing.nama}'); 35 print('Umur: \${kucing.umur} tahun'); 36 print('Berat: \${kucing.berat} kg'); 37 } 38 </pre> <p>Hasil run :</p>  <pre> Melati makan. Melati sedang tidur Nama: Melati Umur: 2 tahun Berat: 2.2 kg </pre>	<p>Pada gambar 13, menggunakan konsep abstract. Sebelumnya Kita telah membuat class Hewan, untuk menjadikan sebuah kelas menjadi abstract hanya perlu menambahkan keyword abstract sebelum penulisan kelas : main.dart seperti pada gambar program disamping.</p> <ul style="list-style-type: none"> • Kelas Kucing merupakan turunan dari kelas abstrak Hewan. Konstruktor Kucing memanggil konstruktor kelas induk (super(nama, umur, berat)). Lalu metode makan() di-override dari kelas Hewan. Ketika kucing makan, beratnya bertambah sebesar 0.2. • Jadi, secara keseluruhan, program ini menggambarkan pewarisan antara kelas abstrak Hewan dan kelas turunan Kucing,

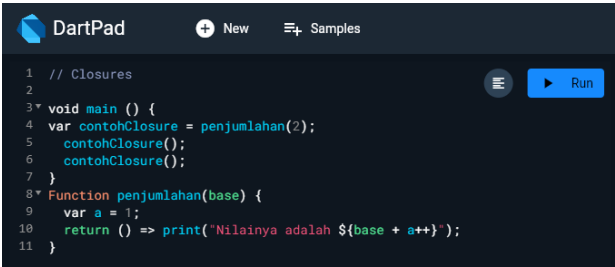
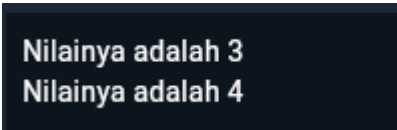
	<p>di mana Kucing mengimplementasikan metode abstrak makan()).</p>
<p>Gambar 14</p>  <pre> 1 //([11].Implicit Interface 2 import 'hewan.dart'; 3 4 class Flyable { 5 void fly() {} 6 } 7 8 class Burung extends Hewan implements Flyable { 9 String warna; 10 11 Burung(String nama, int umur, double berat, this.warna) 12 : super(nama, umur, berat); 13 14 @override 15 void fly() { 16 print('\$nama is flying'); 17 } 18 } 19 20 Run Debug 21 void main() { 22 Burung elang = Burung('Elang', 5, 2.5, 'brown'); 23 elang.fly(); // Output: Elang is flying </pre> <p>Hasil run :</p> 	<p>Pada gambar 14, menggunakan konsep implicit interface. Dart tidak memiliki keyword atau syntax untuk mendeklarasikan interface seperti bahasa pemrograman OOP lainnya. Setiap class di dalam Dart dapat bertindak sebagai interface. Oleh karena itu interface pada dart dikenal sebagai implicit interface.</p> <ul style="list-style-type: none"> • Untuk mengimplementasikan interface, perlu menggunakan keyword implements. Beberapa interface dapat diimplementasikan sekaligus pada satu kelas. • <code>@override</code> disini menunjukan fungsi tersebut mengesampingkan fungsi yang ada di interface atau kelas induknya, lalu menggunakan fungsi yang ada dalam kelas itu sendiri

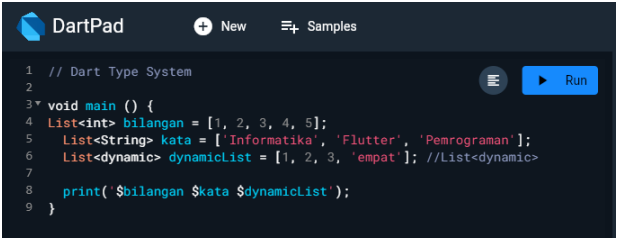
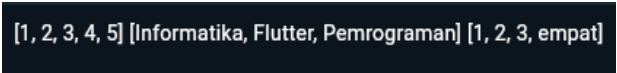
	sebagai gantinya.
<p>Gambar 15</p>  <p>The screenshot shows the DartPad interface. The code defines two enums: <code>Pelangi</code> with values <code>merah, jingga, kuning, hijau, biru, nila, ungu</code> and <code>Status</code> with values <code>Todo, InProgress, InReview, Done</code>. The <code>main</code> function prints the values of <code>Pelangi</code>, a specific value <code>Pelangi.kuning</code>, and the index of <code>Pelangi.biru</code>.</p> <p>Hasil run :</p> <pre>[Pelangi.merah, Pelangi.jingga, Pelangi.kuning, Pelangi.hijau, Pelangi.biru, Pelangi.nila, Pelangi.ungu] Pelangi.kuning 4</pre>	<p>Pada gambar 15, menggunakan enumerated types, yang dimana enum mewakili Kumpulan konstan yang membuat kode lebih jelas dan mudah dibaca.</p> <ul style="list-style-type: none"> • Pertama, mendeklarasikan enum <code>Pelangi</code> (dimana enum <code>Pelangi</code> memiliki beberapa nilai konstan yang mewakili warna-warna <code>Pelangi</code>, seperti <code>merah,jingga,kuning,hijau, biru,nila dan ungu</code>). • Mendeklarasikan enum <code>status</code> (memiliki beberapa nilai konstan yang mewakili status tugas, seperti “<code>Todo</code>”, “<code>In_Progress</code>”, “<code>In_Review</code>”, dan “<code>Done</code>”. • Memanggil <code>Pelangi.values</code> untuk menghasilkan daftar semua nilai dalam enum <code>Pelangi</code>, <code>Pelangi.kuning</code> untuk mengakses nilai konstan “<code>kuning</code>” dari enum <code>Pelangi</code> dan <code>Pelangi.biru.index</code> untuk

	<p>mengakses indeks (urutan) nilai konstan “biru” dalam enum Pelangi.</p>
<p>Gambar 16</p>  <pre> 1 //Paradigma Functional Programming 2 // Pure Function 3 4* int sum(int angka1, int angka2) { 5 return angka1 + angka2; 6 } 7* void main() { 8 int hasil = sum(10, 20); 9 print('Hasil penjumlahan: \$hasil'); 10 } </pre> <p>Hasil run :</p> 	<p>Pada gambar 16, menggunakan konsep pure function (sebuah fungsi yang bergantung dengan argumen atau parameter yang dimasukkan ke dalamnya). Pada sum, fungsi ini mengambil dua argumen (angka1 dan angka2) dan mengembalikan hasil penjumlahan keduanya. Kemudian main() merupakan fungsi utama untuk memanggil fungsi sum(10, 20) dan mencetak hasilnya.</p>
<p>Gambar 17</p>  <pre> 1 //Recursion 2* int fibonacci(n) { 3* if (n <= 0) { 4 return 0; 5* } else if(n == 1) { 6 return 1; 7* } else { 8 return fibonacci(n-1) + fibonacci(n-2); 9 } 10 } 11 12* void main () { 13 int n = 10; 14 int result = fibonacci(n); 15 print('Nilai Fibonacci untuk n = \$n adalah \$result'); 16 } </pre> <p>Hasil run :</p> 	<p>Pada gambar 17, menggunakan konsep recursion. Pertama, pada fungsi fibonacci(n) menghitung nilai bilangan Fibonacci ke-n. Bilangan Fibonacci adalah deret angka di mana setiap angka adalah hasil penjumlahan dua angka sebelumnya. Fungsi ini memiliki tiga kondisi:</p>

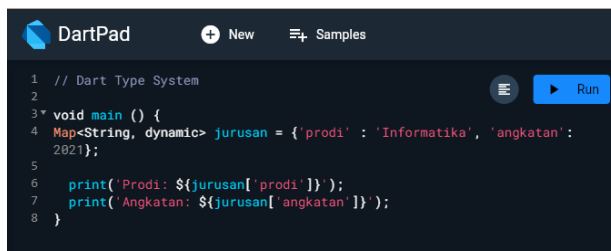
	<ol style="list-style-type: none"> 1. Jika n kurang dari atau sama dengan 0, kembalikan 0. 2. Jika n sama dengan 1, kembalikan 1. 3. Jika tidak, kembalikan hasil penjumlahan dari dua panggilan rekursif: <code>fibonacci(n - 1)</code> dan <code>fibonacci(n - 2)</code>. <p>Kemudian pada <code>main()</code> yang merupakan fungsi utama akan menghitung nilai Fibonacci untuk $n = 10$ dan mencetak hasilnya.</p>
<p>Gambar 18</p>  <p>Hasil run :</p> 	<p>Pada gambar 18, menggunakan anonymous functions yang dimana Anonymous function sendiri juga dikenal dengan nama lambda. Untuk membuatnya kita cukup menuliskan tanda kurung untuk menerima parameter dan body function-nya. Dan selain itu juga dapat menggunakan expression</p>

	<p>untuk membuat kode fungsi menjadi lebih ringkas dengan fat arrow (=>).</p>
<p>Gambar 19</p>  <p>Hasil run :</p> 	<p>Pada gambar 19, menggunakan higher order functions (Fungsi yang menggunakan fungsi lainnya sebagai parameter, menjadi tipe kembalian, atau keduanya). Pada contohHigherOrderFunction: Fungsi ini mengambil dua parameter:</p> <ol style="list-style-type: none"> 1. message: Sebuah pesan yang akan dicetak. 2. myFunction: Sebuah fungsi yang akan diterima sebagai argumen. <ul style="list-style-type: none"> • Opsi 1, Kita mendefinisikan variabel sum sebagai fungsi anonim dengan notasi panah (=>). Fungsi ini mengambil dua argumen (num1 dan num2) dan mengembalikan hasil penjumlahan keduanya. • Opsi 2, Kita langsung

	<p>memanggil fungsi contohHigherOrderFunction dengan menggunakan fungsi anonim sebagai argumen. Fungsi anonim ini juga mengambil dua argumen (num1 dan num2) dan mengembalikan hasil penjumlahan.</p>
<p>Gambar 20</p>  <pre> 1 // Closures 2 3* void main () { 4 var contohClosure = penjumlahan(2); 5 contohClosure(); 6 contohClosure(); 7 } 8* Function penjumlahan(base) { 9 var a = 1; 10 return () => print("Nilainya adalah \${base + a++}"); 11 } </pre> <p>Hasil run :</p>  <pre> Nilainya adalah 3 Nilainya adalah 4 </pre>	<p>Pada gambar 20, menggunakan closures. closures memungkinkan kita untuk mengakses variabel lokal di luar lingkup fungsi, bahkan setelah fungsi tersebut selesai dieksekusi.</p> <ul style="list-style-type: none"> • Fungsi penjumlahan(base): mengambil satu parameter (base) dan mengembalikan sebuah fungsi anonim (closure). Fungsi anonim ini memiliki akses ke variabel lokal a dan parameter base. Setiap kali fungsi anonim dipanggil, nilai a akan bertambah satu.

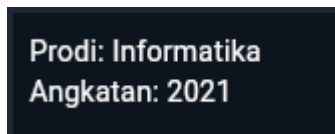
	<ul style="list-style-type: none"> • Variabel <code>contohClosure</code>: menyimpan hasil dari pemanggilan <code>penjumlahan(2)</code>. Ini berarti kita memiliki sebuah closure yang memiliki akses ke <code>base = 2</code> dan variabel lokal <code>a</code>. • Pemanggilan <code>contohClosure()</code>: Kita memanggil closure dua kali. Setiap kali closure dipanggil, nilai <code>a</code> bertambah satu, dan hasil penjumlahan dicetak.
<p>Gambar e</p>  <p>Hasil run gambar e :</p> 	<p>Pada gambar e, menggunakan konsep generic (Untuk membuat tipe data yang lebih spesifik dan memastikan keamanan tipe data pada saat kompilasi). Pertama, kita mendeklarasikan List dengan Tipe Data:</p> <ol style="list-style-type: none"> 1. Variabel <code>bilangan</code> adalah sebuah list yang hanya dapat berisi nilai-nilai

Gambar f



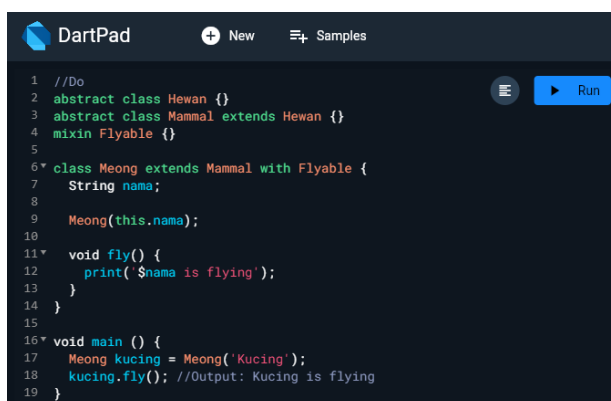
```
1 // Dart Type System
2
3 void main () {
4   Map<String, dynamic> jurusan = {'prodi' : 'Informatika', 'angkatan':
5     2021};
6   print('Prodi: ${jurusan['prodi']}');
7   print('Angkatan: ${jurusan['angkatan']}');
8 }
```

Hasil run gambar f :



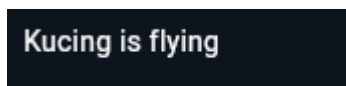
Prodi: Informatika
Angkatan: 2021

Gambar g



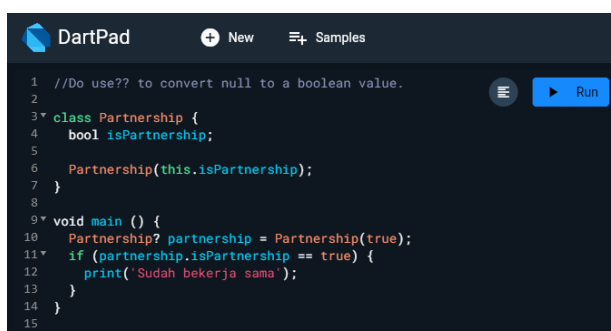
```
1 //Do
2 abstract class Hewan {}
3 abstract class Mammal extends Hewan {}
4 mixin Flyable {}
5
6 class Meong extends Mammal with Flyable {
7   String nama;
8
9   Meong(this.nama);
10
11 void fly() {
12   print('$nama is flying');
13 }
14 }
15
16 void main () {
17   Meong kucing = Meong('Kucing');
18   kucing.fly(); //Output: Kucing is flying
19 }
```

Hasil run gambar g :



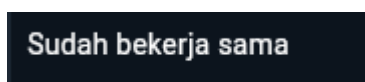
Kucing is flying

Gambar h



```
1 //Do use?? to convert null to a boolean value.
2
3 class Partnership {
4   bool isPartnership;
5
6   Partnership(this.isPartnership);
7 }
8
9 void main () {
10   Partnership? partnership = Partnership(true);
11   if (partnership.isPartnership == true) {
12     print('Sudah bekerja sama');
13   }
14 }
15
```

Hasil run :



Sudah bekerja sama

bertipe data int.

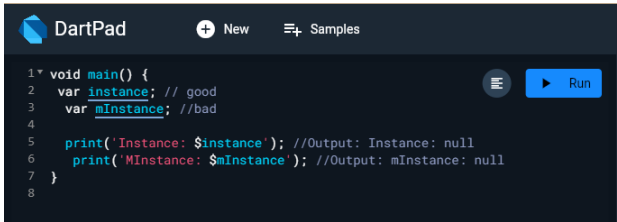
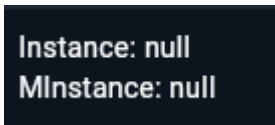
2. Variabel kata adalah sebuah list yang hanya dapat berisi nilai-nilai bertipe data String.
3. Variabel `dynamicList` adalah sebuah list yang dapat berisi nilai-nilai dari berbagai tipe data, termasuk int dan String.

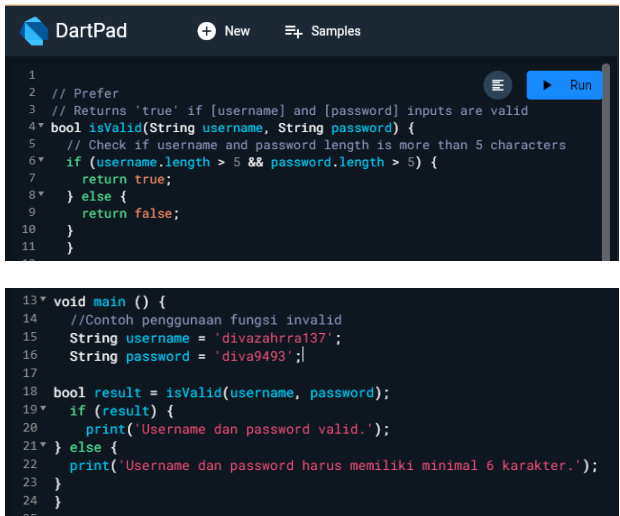
Pada gambar f, menggunakan Type Inference. Dart memiliki analyzer yang dapat menentukan tipe untuk field, method, variabel lokal, dan beberapa tipe argumen generic.

- Pertama, mendeklarasikan variabel `jurusan` dengan tipe data `Map`. `Map` ini memiliki kunci berupa `String` (misalnya `'prodi'`) dan nilai berupa `dynamic` (dalam hal ini, `'Informatika'` dan `2021`).
- `print('Prodi: ${jurusan['prodi']}');`
Mengakses nilai dengan kunci `'prodi'` dari `Map`

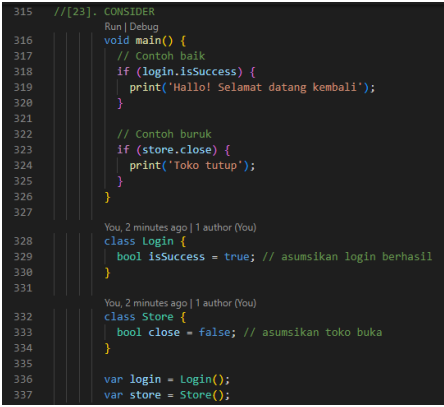

	<p>jurusan dan mencetaknya.</p> <ul style="list-style-type: none"> • <code>print('Angkatan: \${jurusan['angkatan']}');:</code> Mengakses nilai dengan kunci 'angkatan' dari Map jurusan dan mencetaknya. <p>Pada gambar g, menerapkan DO name type using UpperCamelCase (Class, enum, typedef, dan type parameter harus menggunakan huruf kapital pada huruf pertama dari setiap kata termasuk kata pertama.).</p> <ul style="list-style-type: none"> • Pertama, mendeklarasikan abstract class dengan nama Hewan, abstract class dengan nama Mammal yang mengextends (mewarisi) dari Hewan, • Mendeklarasikan mixin dengan nama Flyable. Mixin adalah cara untuk mendefinisikan kode yang dapat digunakan kembali dalam beberapa hierarki kelas. Flyable dapat digunakan dengan
--	--

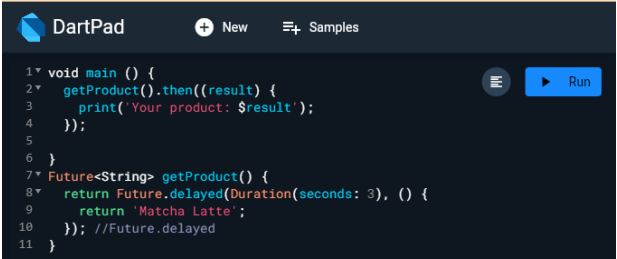
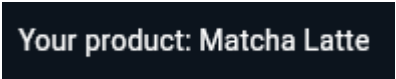
	<p>menggunakan kata kunci with di kelas lain.</p> <ul style="list-style-type: none"> • Mendeklarasikan kelas Meong yang meng-extends dari Mammal dan menggunakan mixin Flyable. Kelas Meong memiliki sifat-sifat dari Mammal dan juga memiliki kemampuan yang didefinisikan dalam Flyable. <p>Pada gambar h, kita mendeklarasikan Kelas Partnership: Kelas ini memiliki satu atribut isPartnership dengan tipe data bool. Kemudian konstruktor menerima satu parameter (isPartnership) dan menginisialisasi atribut isPartnership dengan nilai yang diberikan.</p> <ul style="list-style-type: none"> • Variabel partnership: Variabel ini menyimpan objek dari kelas Partnership dengan nilai true untuk atribut isPartnership.
--	---

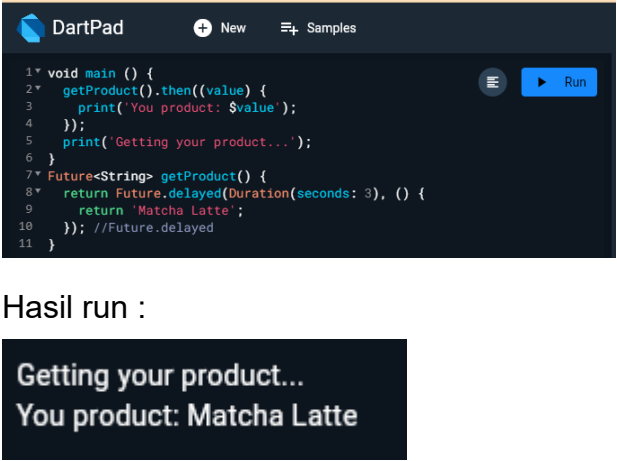
	<ul style="list-style-type: none"> Kita disini menggunakan null-aware operator (?.) pada baris if (partnership.isPartnership == true) untuk memastikan bahwa kita hanya mengakses isPartnership jika objek partnership tidak null. Jadi, Jika isPartnership bernilai true, kita mencetak pesan “Sudah bekerja sama”.
<p>Gambar 21</p>  <pre> 1 void main() { 2 var instance; // good 3 var mInstance; //bad 4 5 print('Instance: \$instance'); //Output: Instance: null 6 print('MInstance: \$mInstance'); //Output: mInstance: null 7 } 8 </pre> <p>Hasil run :</p>  <pre> Instance: null MInstance: null </pre>	<p>Pada gambar 21, menerapkan aturan don't. yang dimana Aturan yang diawali dengan DON'T tidak baik untuk diterapkan. Contohnya : DON'T use prefix letters.</p> <ul style="list-style-type: none"> Pertama, mendeklarasikan variabel instance menggunakan kata kunci var. Penggunaan var memungkinkan Dart untuk menentukan tipe data variabel berdasarkan nilai yang diberikan. Dalam hal ini, karena kita belum memberikan nilai, instance akan memiliki nilai null.

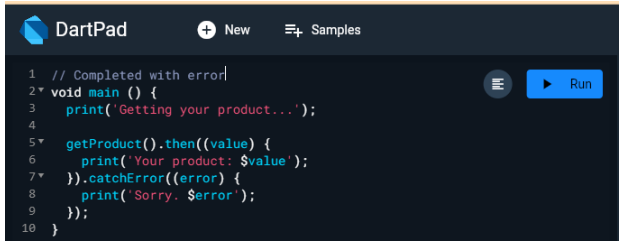
	<ul style="list-style-type: none"> • Mendeklarasikan variabel <code>mInstance</code> menggunakan kata kunci <code>var</code>. Namun, masalahnya adalah nama variabel ini tidak deskriptif. Nama <code>mInstance</code> tidak memberikan informasi yang jelas tentang tujuan variabel tersebut. Sebagai praktik yang baik, kita sebaiknya menggunakan nama variabel yang lebih bermakna dan mudah dipahami.
<p>Gambar 22</p>  <pre> 1 // Prefer 2 // Returns 'true' if [username] and [password] inputs are valid 3 4 bool isValid(String username, String password) { 5 // Check if username and password length is more than 5 characters 6 if (username.length > 5 && password.length > 5) { 7 return true; 8 } else { 9 return false; 10 } 11 } 12 13 void main () { 14 //Contoh penggunaan fungsi invalid 15 String username = 'divazahra137'; 16 String password = 'diva9493'; 17 18 bool result = isValid(username, password); 19 if (result) { 20 print('Username dan password valid.');</pre>	<p>Pada gambar 22, menggunakan konsep prefer (Praktik yang harus diikuti). Namun, ada kemungkinan keadaan yang lebih masuk akal untuk melakukan sebaliknya.</p> <ul style="list-style-type: none"> • Pertama, fungsi <code>isValid</code> disini memeriksa apakah panjang username dan password lebih dari 5

<p>Hasil run :</p> <div data-bbox="255 259 705 331" data-label="Text"> <pre>Username dan password valid.</pre> </div>	<p>karakter.</p> <ul style="list-style-type: none"> • Jika ya, fungsi mengembalikan true, yang menandakan bahwa input valid. Jika tidak, fungsi mengembalikan false. • Contoh penggunaan fungsi isValid: username = 'divazahra137' password = 'diva9493' Karena keduanya memiliki lebih dari 5 karakter, hasilnya adalah "Username dan password valid."
<p>Gambar 23</p> <div data-bbox="255 1133 871 1328" data-label="Image"> <pre> 1* void main () { 2 String nama = 'Diva Zahra'; 3 int birthYear = 2003; 4 int thisYear = DateTime.now().year; 5 6 print('Hallo, \$nama, berumur \${thisYear - birthYear} tahun.');</pre> </div> <p>Hasil run :</p> <div data-bbox="255 1400 769 1476" data-label="Text"> <pre>Hallo, Diva Zahra, berumur 21 tahun.</pre> </div>	<p>Pada gambar 23, menggunakan konsep avoid. Avoid sendiri adalah kebalikan dari PREFER yang menjelaskan hal-hal yang tidak boleh dilakukan, namun kemungkinan ada alasan bagus untuk melakukannya pada beberapa kejadian. Dalam kode di samping, kita mendefinisikan variabel nama dengan nilai 'Diva Zahra', birthYear dengan nilai 2003, dan thisYear dengan nilai tahun saat ini menggunakan</p>

	DateTime.now().year.
<p>Gambar 24</p>  <pre> 315 //[[29]]. CONSIDER 316 Run Debug 317 void main() { 318 // Contoh baik 319 if (login.isSuccess) { 320 print('Hallo! Selamat datang kembali'); 321 } 322 // Contoh buruk 323 if (store.close) { 324 print('Toko tutup'); 325 } 326 } 327 328 You, 2 minutes ago 1 author (You) 329 class Login { 330 bool isSuccess = true; // asumsikan login berhasil 331 } 332 333 You, 2 minutes ago 1 author (You) 334 class Store { 335 bool close = false; // asumsikan toko buka 336 } 337 338 var login = Login(); 339 var store = Store(); </pre> <p>Hasil run :</p>  <pre> Hallo! Selamat datang kembali </pre>	<p>Pada gambar 24, menggunakan konsep consider (Bisa diikuti atau tidak diikuti, tergantung pada keadaan dan preferensi). penamaan dalam kode baik itu nama variabel, fungsi, maupun lainnya adalah hal yang sangat penting namun juga tidak mudah. Solusinya kita membuat seolah-olah sedang membuat kalimat.</p> <ul style="list-style-type: none"> • Terdapat dua kelas, yaitu Login dan Store. Kelas Login memiliki properti isSuccess yang menandakan status keberhasilan login, dan kelas Store memiliki properti close yang menandakan apakah toko tutup atau tidak. • Dalam fungsi main, terdapat dua blok kondisi if. Blok pertama memeriksa apakah login berhasil (login.isSuccess) dan jika ya, maka akan mencetak

	<p>pesan selamat datang. Blok kedua memeriksa apakah toko tutup (store.close) dan jika ya, maka akan mencetak pesan bahwa toko tutup.</p>
<p>Gambar 25</p>  <p>Hasil run :</p> 	<p>Pada gambar 25, menggunakan konsep Future yang dimana memungkinkan kita untuk menangani operasi asinkron dengan cara yang lebih terstruktur dan mudah dibaca.</p> <ul style="list-style-type: none"> • Fungsi getProduct() mengembalikan sebuah Future. Fungsi ini akan menghasilkan nilai bertipe String di masa depan setelah operasi asinkron selesai. Dalam contoh ini, kita menggunakan Future.delayed untuk mensimulasikan operasi yang memakan waktu selama 3 detik. Setelah 3 detik, fungsi ini mengembalikan nilai 'Matcha Latte'. • Kita menggunakan

	<p>metode <code>.then()</code> pada hasil dari pemanggilan <code>getProduct()</code>. Metode ini akan mengeksekusi fungsi yang diberikan ketika Future selesai. Dalam contoh ini, kita mencetak pesan “Your product: Matcha Latte” setelah 3 detik.</p>
<p>Gambar 26</p>  <p>The image shows a screenshot of the DartPad web interface. The top bar has the DartPad logo, a 'New' button, and a 'Samples' menu. The code editor contains the following Dart code:</p> <pre> 1 void main () { 2 getProduct().then((value) { 3 print('You product: \$value'); 4 }); 5 print('Getting your product...'); 6 } 7 Future<String> getProduct() { 8 return Future.delayed(Duration(seconds: 3), () { 9 return 'Matcha Latte'; 10 }); //Future.delayed 11 } </pre> <p>Below the code editor, the output is displayed in a dark box with white text:</p> <pre> Getting your product... You product: Matcha Latte </pre>	<p>Pada gambar 26, Setelah Future dijalankan, perlu adanya handler untuk menangani status completed with data. Caranya dengan menggunakan method <code>.then()</code> dari objek Future.</p> <ul style="list-style-type: none"> • <code>Future<String></code> <code>getProduct()</code>: sebuah fungsi yang mengembalikan objek Future dengan tipe data String. Fungsi ini menggunakan <code>Future.delayed</code> untuk menunda eksekusi selama 3 detik sebelum mengembalikan nilai. • Metode <code>Future.delayed</code> yang menunda eksekusi

	<p>selama 3 detik. Setelah waktu tertunda, fungsi anonim yang diberikan akan dijalankan.</p> <ul style="list-style-type: none"> • <code>return 'Matcha Latte';</code>: Fungsi anonim mengembalikan string “Matcha Latte” setelah waktu tertunda. • <code>main()</code> memanggil <code>getProduct()</code>. Kemudian menggunakan <code>.then()</code> untuk menangani hasil dari <code>getProduct()</code>. Jika berhasil, mencetak pesan “You product: ...” dengan nilai yang diterima. Jika terjadi kesalahan, menangkapnya dengan <code>.catchError()</code> dan mencetak pesan kesalahan. Akhirnya, mencetak “Thank you”.
<p>Gambar 27</p>  <pre> 1 // Completed with error 2 void main () { 3 print('Getting your product...'); 4 5 getProduct().then((value) { 6 print('Your product: \$value'); 7 }).catchError((error) { 8 print('Sorry. \$error'); 9 }); 10 } </pre>	<p>Pada gambar 27, menggunakan <code>completed with error</code> yang dimana fungsi ini untuk mengatasi eror atau exception di dalam Future, kita dapat menambahkan method <code>.catchError()</code> setelah</p>

```

12 Future<String> getProduct() {
13   return Future.delayed(Duration(seconds: 3), () {
14     var isProductAvailable = false;
15     //ignore: dead_code
16     if (isProductAvailable) {
17       return 'Coffee Boba';
18     } else {
19       throw 'Our stock is not enough.';
20     }
21   }); // Future.delayed
22 }

```

Hasil run :

```

Getting your product...
Sorry. Our stock is not enough.

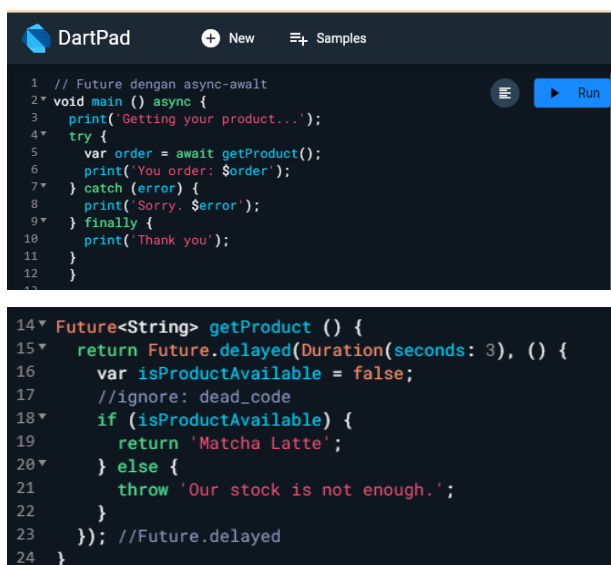
```

then.

Program ini sama seperti pada program sebelumnya, namun disini terdapat fungsi:

- var isProductAvailable = false;; Variabel isProductAvailable diberi nilai false. Ini menunjukkan bahwa stok produk tidak cukup.
- throw 'Our stock is not enough.';: Jika stok produk tidak mencukupi, fungsi akan melempar kesalahan dengan pesan "Our stock is not enough."

Gambar 28



Pada gambar 28, menggunakan konsep Future dengan async-await. Dart mempunyai keyword async dan await yang merupakan alternatif untuk dapat menuliskan proses asynchronous layaknya proses synchronous. • main(): Fungsi utama yang menggunakan async dan await. Ini

Hasil run :

```
Getting your product...  
Sorry. Our stock is not enough.  
Thank you
```

memungkinkan kita menunggu hasil dari `getProduct()` tanpa menghentikan eksekusi program. Pertama, mencetak pesan "Getting your product..." untuk memberi tahu pengguna bahwa produk sedang diambil. Kemudian menggunakan `await` untuk menunggu hasil dari `getProduct()`. Jika berhasil, mencetak pesan "Your product: ..." dengan nilai yang diterima. Jika terjadi kesalahan, menangkapnya dengan `catchError()` dan mencetak pesan kesalahan. Akhirnya, mencetak "Thank you".

- `Future<String>`

`getProduct()`: fungsi yang mengembalikan objek `Future` dengan tipe data `String`. Fungsi ini menggunakan `Future.delayed` untuk menunda eksekusi selama 3 detik sebelum mengembalikan nilai.

- Lalu metode

	<p>Future.delayed menunda eksekusi selama 3 detik. Setelah waktu tertunda, fungsi anonim yang diberikan akan dijalankan.</p> <ul style="list-style-type: none"> • return 'Matcha Latte';: Fungsi anonim mengembalikan string "Matcha Latte" setelah waktu tertunda. • Variabel isProductAvailable diberi nilai false. Ini menunjukkan bahwa stok produk tidak cukup. • throw 'Our stock is not enough.':; Jika stok produk tidak mencukupi, fungsi akan melempar kesalahan dengan pesan "Our stock is not enough."
--	---