

LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK



JUDUL :
FUNDAMENTAL DART

Disusun oleh:
Oktavia Ayu Andini (21102081)

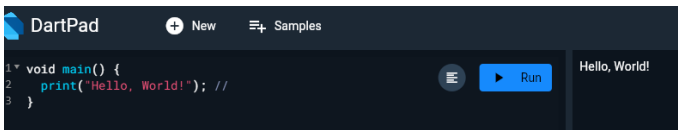
TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
BANYUMAS, JAWA TENGAH
2024

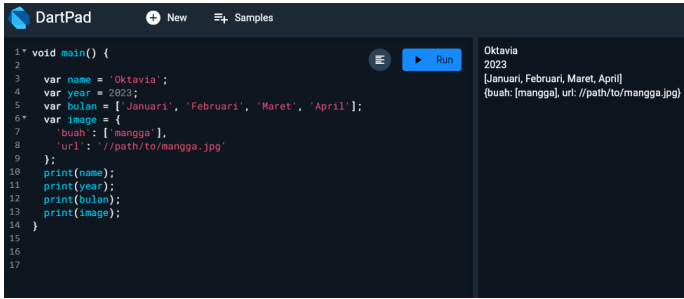
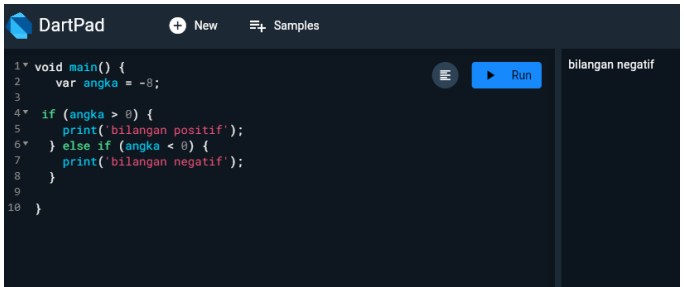
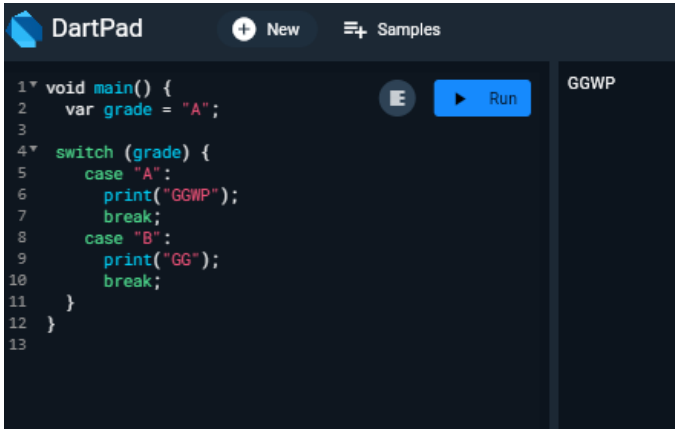
Pembahasan

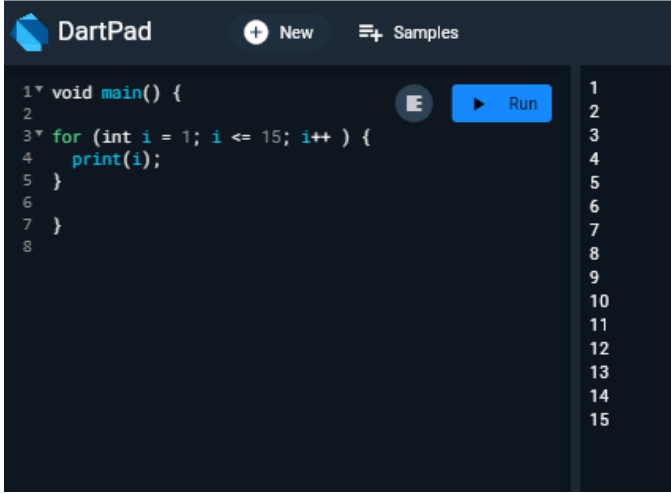
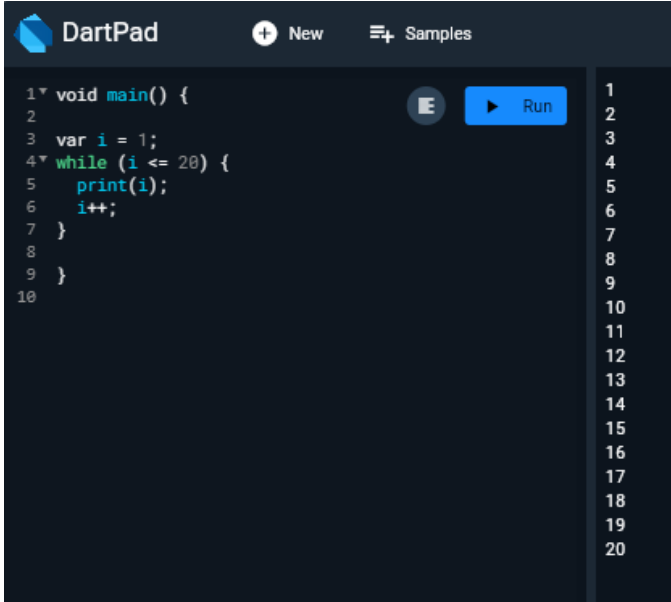
Praktikum Fundamental Dart sangat penting dilakukan karena memberikan pemahaman yang kuat tentang dasar-dasar bahasa pemrograman Dart, yang merupakan fondasi bagi pengembangan aplikasi Flutter yang efektif dan efisien. Dengan melalui praktikum ini, mahasiswa dapat memahami konsep-konsep dasar seperti tipe data, variabel, struktur kontrol, fungsi, dan pemrograman berorientasi objek dalam konteks bahasa Dart. Tujuan utamanya adalah untuk memberikan landasan yang kokoh bagi mahasiswa dalam memahami sintaks dan paradigma yang digunakan dalam Dart, sehingga mereka dapat mengembangkan aplikasi Flutter dengan lebih baik.

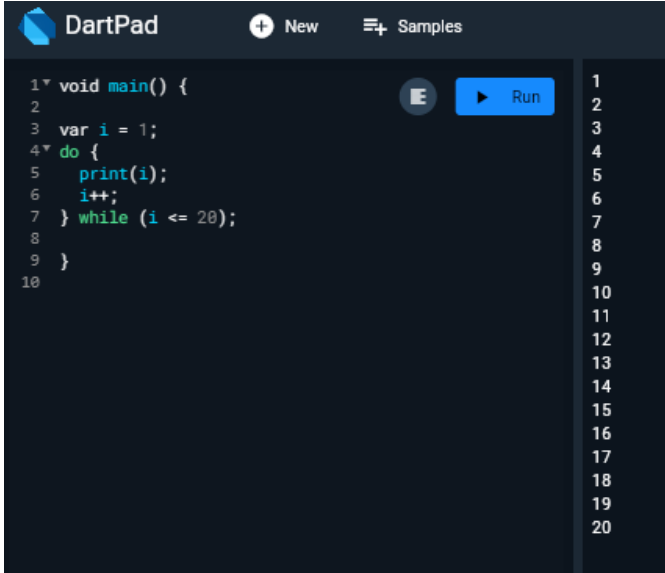
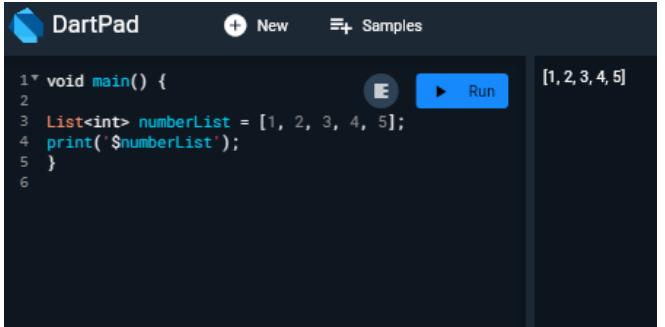
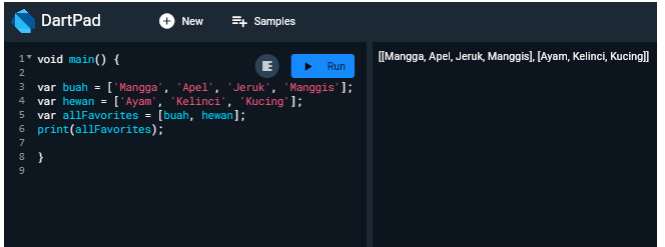
Selain itu, praktikum Fundamental Dart bertujuan untuk melatih kemampuan mahasiswa dalam menulis kode yang bersih, efisien, dan mudah dipahami. Melalui latihan-latihan yang terstruktur, mahasiswa akan diajak untuk mengimplementasikan konsep-konsep Dart dalam skenario-skenario nyata, sehingga mereka dapat memperoleh pengalaman praktis yang berharga dalam menyelesaikan tugas-tugas pemrograman. Dengan demikian, praktikum ini tidak hanya bertujuan untuk memahami konsep-konsep teoritis, tetapi juga untuk membekali mahasiswa dengan keterampilan praktis yang dapat mereka terapkan dalam pengembangan aplikasi sehari-hari menggunakan Dart dan Flutter.

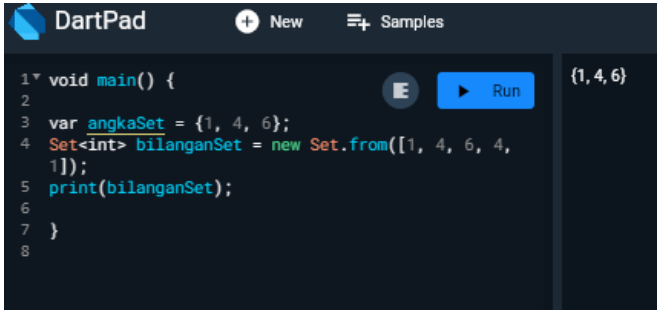
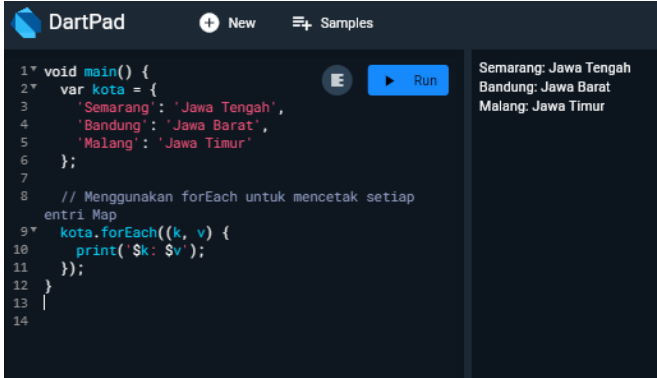
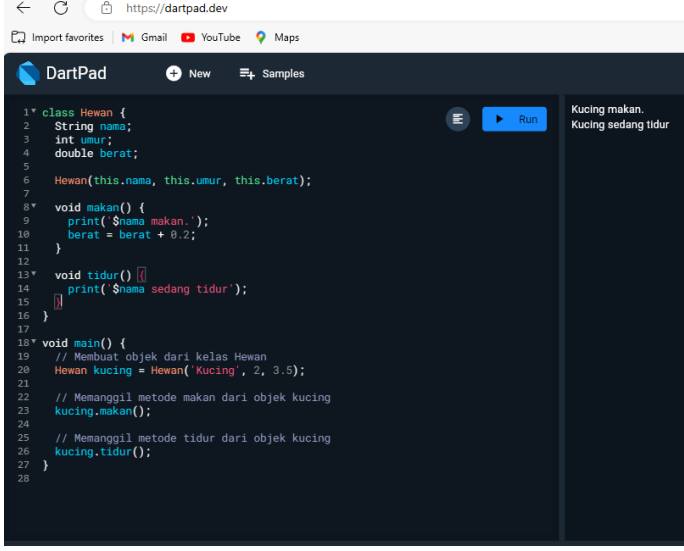
Langkah-Langkah Praktikum

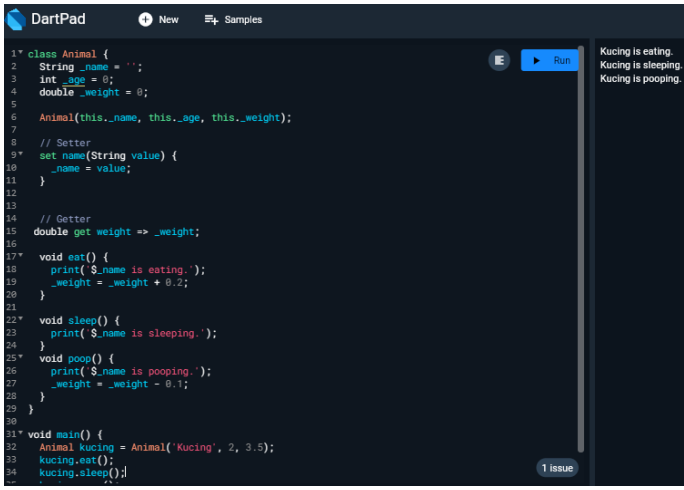
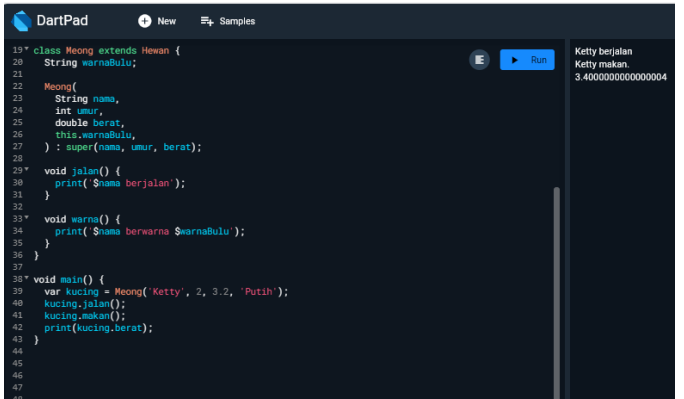
Langkah Praktikum	Pembahasan
<p>Program hello world</p> 	<p>Pada awal praktikum ini kita membuat Hello World, dengan mendefinisikan main, dan melakukan perintah print "Hello World".</p>

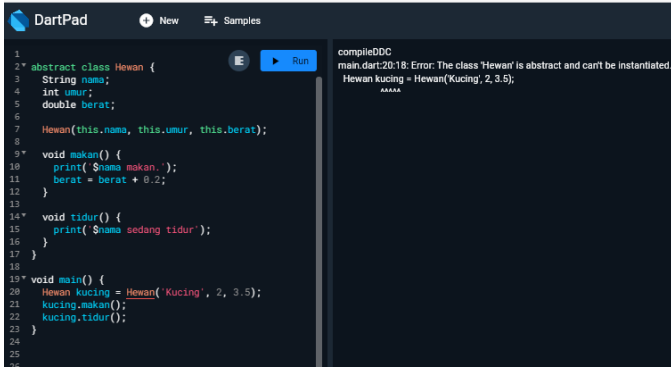
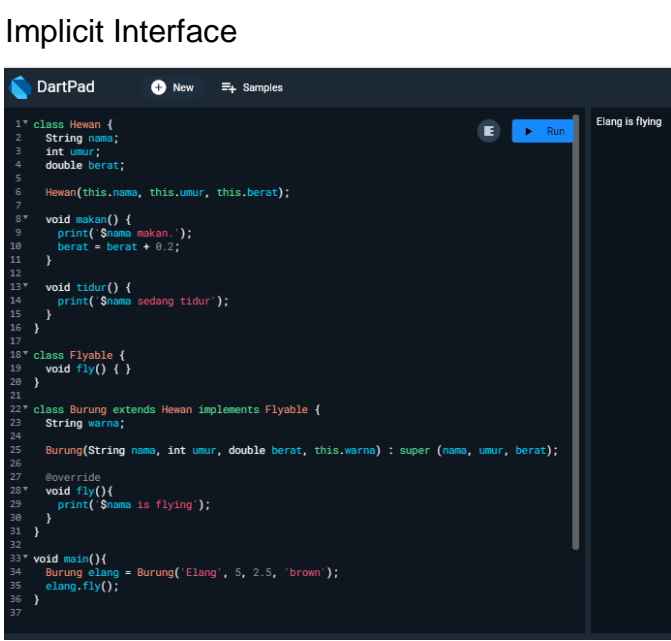
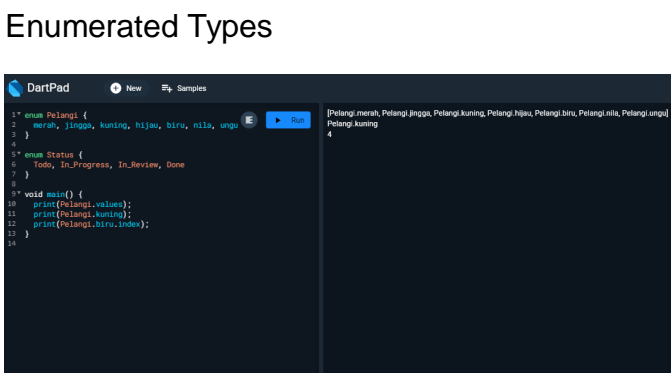
<p>Variabel</p>  <pre> 1* void main() { 2 3 var name = 'Oktavia'; 4 var year = 2023; 5 var bulan = ['Januari', 'Februari', 'Maret', 'April']; 6* var image = { 7 'bush': ['mangga'], 8 'url': '//path/to/mangga.jpg' 9 }; 10 print(name); 11 print(year); 12 print(bulan); 13 print(image); 14 } 15 16 17 </pre>	<p>Untuk mendefinisikan variable, menggunakan keyword “var”.</p> <p>Kemudian untuk memunculkan hasilnya menggunakan perintah print.</p>
<p>Control Flow</p> <p>- If and else</p>  <pre> 1* void main() { 2 var angka = -8; 3 4* if (angka > 0) { 5 print('bilangan positif'); 6* } else if (angka < 0) { 7 print('bilangan negatif'); 8 } 9 10 } </pre>	<p>Control flow merupakan mekanisme yang digunakan untuk mengatur alur eksekusi dari sebuah program.</p> <p>-</p>
<p>- Switch case</p>  <pre> 1* void main() { 2 var grade = "A"; 3 4* switch (grade) { 5 case "A": 6 print("GGWP"); 7 break; 8 case "B": 9 print("GG"); 10 break; 11 } 12 } 13 </pre>	<p>program disamping merupakan program Dart yang menetapkan variabel grade dengan nilai "A" dan kemudian menggunakan struktur kontrol switch untuk memilih tindakan yang sesuai berdasarkan nilai grade. Dalam kasus ini, karena grade sama dengan "A", maka akan mencetak "GGWP" ke konsol.</p>
<p>- For loops</p>	<p>Program disamping menggunakan struktur kontrol for untuk</p>

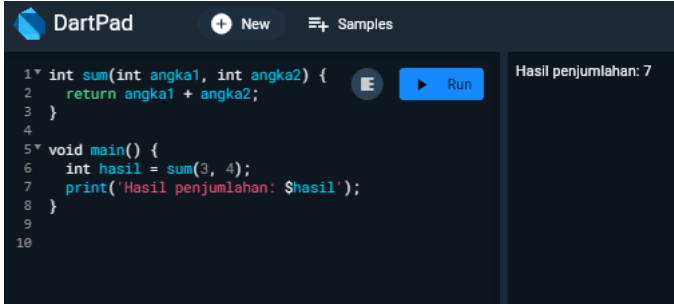
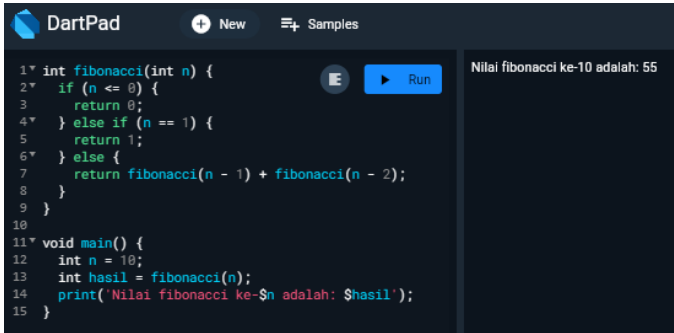
	<p>melakukan iterasi dari 1 hingga 15 dan mencetak nilai i ke konsol pada setiap iterasi. Variabel i dimulai dari 1, kemudian ditingkatkan satu per satu dengan operator ++ setiap kali iterasi dilakukan. Proses iterasi berlangsung selama nilai i kurang dari atau sama dengan 15.</p>
<p>- While and do-while</p> 	<p>while dan do-while merupakan dua struktur kontrol yang digunakan untuk melakukan iterasi berdasarkan kondisi tertentu.</p> <p>While: Struktur kontrol while digunakan untuk melakukan iterasi selama kondisi tertentu bernilai benar. Pertama, kondisi dievaluasi, dan jika kondisi itu benar, blok kode di dalamnya dieksekusi. Setelah itu, kondisi diperiksa lagi. Jika kondisi masih benar, iterasi akan terus berlanjut. Jika</p>

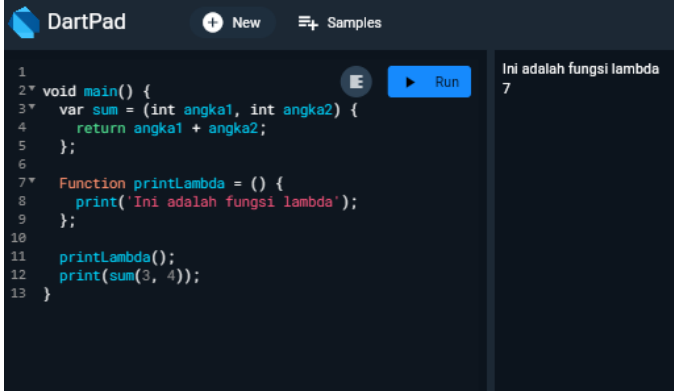


 <pre> 1 void main() { 2 3 var i = 1; 4 do { 5 print(i); 6 i++; 7 } while (i <= 20); 8 9 } 10 </pre>	<p>kondisi salah, iterasi berhenti.</p> <p>Do-While: Struktur kontrol do-while hampir sama dengan while, tetapi di do-while, blok kode di dalamnya akan dieksekusi setidaknya sekali sebelum kondisi dievaluasi. Setelah blok kode dijalankan, kondisi dievaluasi, dan jika benar, iterasi akan berlanjut. Jika kondisi salah, iterasi berhenti.</p>
<p>List</p>  <pre> 1 void main() { 2 3 List<int> numberList = [1, 2, 3, 4, 5]; 4 print('\$numberList'); 5 } 6 </pre>	<p>List digunakan untuk menampilkan koleksi nilai yang terurut.</p> <p>Bisa berisi elemen-elemen dengan tipe data yang sama atau berbeda</p>
<p>Spread Operator</p>  <pre> 1 void main() { 2 3 var buah = ['Mangga', 'Apel', 'Jeruk', 'Manggis']; 4 var hewan = ['Ayam', 'Kelinci', 'Kucing']; 5 var allFavorites = [...buah, ...hewan]; 6 print(allFavorites); 7 8 } 9 </pre>	<p>Spread operator berfungsi untuk menambah banyak nilai kedalam list dengan cara yang singkat.</p> <p>Operator ini dituliskan dengan tanda titik tiga (...).</p>
<p>Set</p>	<p>Set adalah kumpulan nilai unik tanpa urutan tertentu. Karena menggunakan nilai</p>

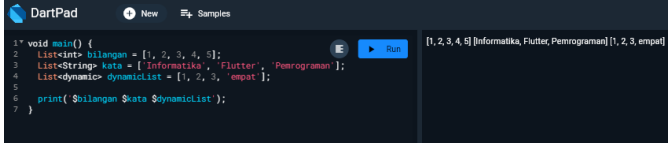
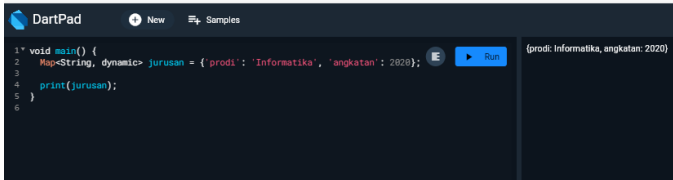
 <pre> 1* void main() { 2 3 var angkaSet = {1, 4, 6}; 4 Set<int> bilanganSet = new Set.from([1, 4, 6, 4, 5 1]); 6 print(bilanganSet); 7 8 } </pre>	<p>unik, maka output yang dihasilkan tidak akan terdapat nilai yang duplikat</p>
<h3>Map</h3>  <pre> 1* void main() { 2* var kota = { 3 'Semarang': 'Jawa Tengah', 4 'Bandung': 'Jawa Barat', 5 'Malang': 'Jawa Timur' 6 }; 7 8 // Menggunakan forEach untuk mencetak setiap 9 // entri Map 10 kota.forEach((k, v) { 11 print('\$k: \$v'); 12 }); 13 14 </pre>	<p>Map adalah sebuah struktur data yang memetakan kunci ke nilai. Setiap kunci dalam map harus unik. Map digunakan untuk memetakan informasi terkait satu sama lain, sehingga memungkinkan pengaksesan data berdasarkan kunci.</p>
<h3>Class</h3>  <pre> 1* class Hewan { 2 String nama; 3 int umur; 4 double berat; 5 6 Hewan(this.nama, this.umur, this.berat); 7 8* void makan() { 9 print('\$nama makan. '); 10 berat = berat + 0.2; 11 } 12 13* void tidur() { 14 print('\$nama sedang tidur'); 15 } 16 17 18* void main() { 19 // Membuat objek dari kelas Hewan 20 Hewan kucing = Hewan('Kucing', 2, 3.5); 21 22 // Memanggil metode makan dari objek kucing 23 kucing.makan(); 24 25 // Memanggil metode tidur dari objek kucing 26 kucing.tidur(); 27 28 } </pre>	<p>Class adalah struktur dasar untuk membuat objek. Class adalah blueprint atau cetak biru untuk objek, yang mendefinisikan atribut (variabel) dan metode (fungsi) yang akan dimiliki oleh objek yang dibuat dari class tersebut</p>
<h3>Properties & method</h3>	<p>Kode ini mendefinisikan sebuah kelas Animal yang memiliki atribut <code>_name</code>,</p>

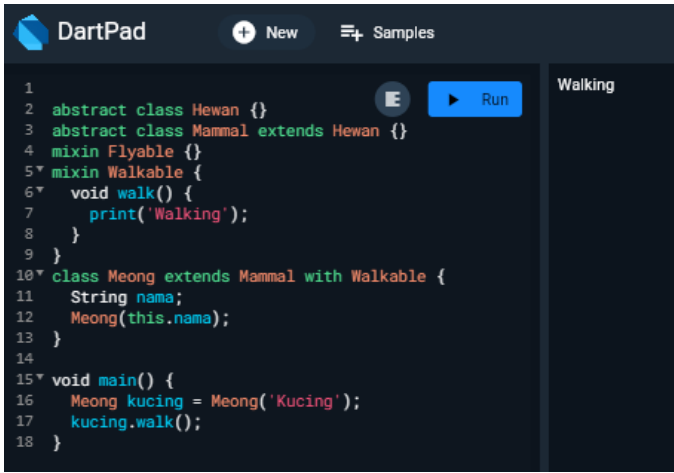
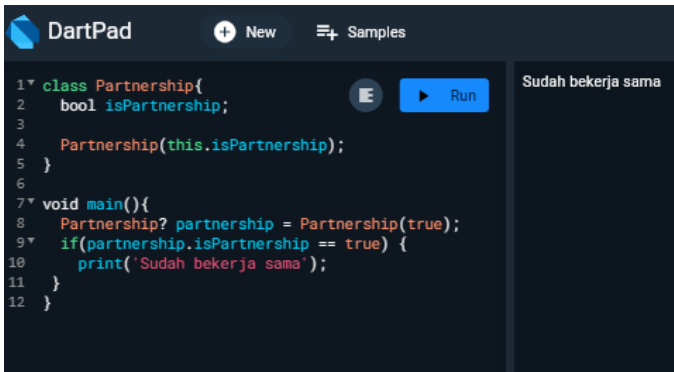
 <pre> 1* class Animal { 2 String _name = ''; 3 int _age = 0; 4 double _weight = 0; 5 6 Animal(this._name, this._age, this._weight); 7 8 // Setter 9* set name(String value) { 10 _name = value; 11 } 12 13 // Getter 14 double get weight => _weight; 15 16 void eat() { 17 print('\$_name is eating. '); 18 _weight = _weight + 0.2; 19 } 20 21 void sleep() { 22 print('\$_name is sleeping. '); 23 } 24 25 void poop() { 26 print('\$_name is pooping. '); 27 _weight = _weight - 0.1; 28 } 29 30 31* void main() { 32 Animal kucing = Animal('Kucing', 2, 3.5); 33 kucing.eat(); 34 kucing.sleep(); 35 } </pre>	<p>_age, dan _weight.</p> <p>Konstruktor Animal digunakan untuk menginisialisasi nilai atribut saat objek dibuat.</p>
<h3>Inheritance</h3>  <pre> 19* class Meong extends Hewan { 20 String warnaBulu; 21 22 Meong(23 String nama, 24 int umur, 25 double berat, 26 this.warnaBulu, 27) : super(nama, umur, berat); 28 29* void jalan() { 30 print('\$_name berjalan'); 31 } 32 33* void warna() { 34 print('\$_name berwarna \$warnaBulu'); 35 } 36 37 38* void main() { 39 var kucing = Meong('Ketty', 2, 3.2, 'Putih'); 40 kucing.jalan(); 41 kucing.makan(); 42 print(kucing.berat); 43 } 44 45 46 47 </pre>	<p>Gambar disamping mendefinisikan sebuah subclass Meong yang merupakan turunan dari superclass Hewan.</p> <p>Subclass ini memiliki tambahan atribut warnaBulu yang merupakan warna bulu dari kucing.</p> <p>Konstruktor Meong menginisialisasi atribut-atribut yang diwarisi dari superclass Hewan serta atribut tambahan warnaBulu.</p>
<h3>Abstract Class</h3>	<p>Kita telah membuat class Hewan sebelumnya, untuk menjadikan sebuah kelas menjadi abstract hanya perlu menambahkan</p>

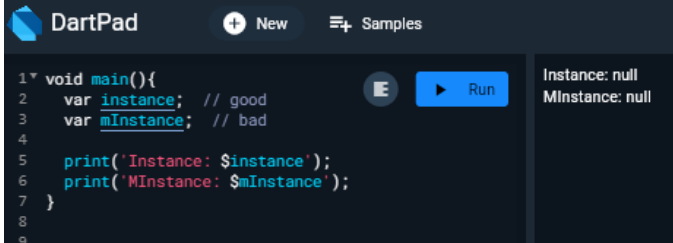
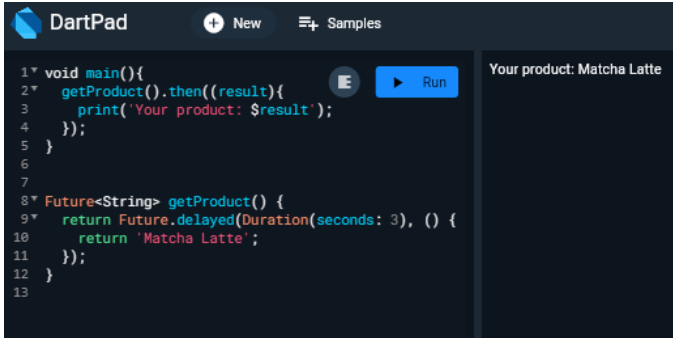
 <pre> 1 2* abstract class Hewan { 3 String nama; 4 int umur; 5 double berat; 6 7 Hewan(this.nama, this.umur, this.berat); 8 9* void makan() { 10 print('\$nama makan. '); 11 berat = berat + 0.2; 12 } 13 14* void tidur() { 15 print('\$nama sedang tidur'); 16 } 17 18 19* void main() { 20 Hewan kucing = Hewan('Kucing', 2, 3.5); 21 kucing.makan(); 22 kucing.tidur(); 23 } 24 25 26 </pre>	<p>keyword <code>abstract</code> sebelum penulisan kelas. Maka akan menghasilkan output error, karena kelas <code>hewan</code> sudah tidak bisa diinisiasikan lagi menjadi sebuah objek.</p>
<h3>Implicit Interface</h3>  <pre> 1* class Hewan { 2 String nama; 3 int umur; 4 double berat; 5 6 Hewan(this.nama, this.umur, this.berat); 7 8* void makan() { 9 print('\$nama makan. '); 10 berat = berat + 0.2; 11 } 12 13* void tidur() { 14 print('\$nama sedang tidur'); 15 } 16 17 18* class Flyable { 19 void fly() {} 20 } 21 22* class Burung extends Hewan implements Flyable { 23 String warna; 24 25 Burung(String nama, int umur, double berat, this.warna) : super(nama, umur, berat); 26 27 @override 28 void fly() { 29 print('\$nama is flying'); 30 } 31 } 32 33* void main() { 34 Burung elang = Burung('Elang', 5, 2.5, 'brown'); 35 elang.fly(); 36 } 37 </pre>	<p>interface pada dart dikenal sebagai <i>implicit interface</i>. Untuk mengimplementasikan interface, perlu menggunakan keyword <code>implements</code>.</p>
<h3>Enumerated Types</h3>  <pre> 1* enum Pelangi { 2 merah, jingga, kuning, hijau, biru, nila, ungu 3 } 4 5* enum Status { 6 Today, In_Progress, In_Review, Done 7 } 8 9* void main() { 10 print(Pelangi.values); 11 print(Pelangi.kuning); 12 print(Pelangi.biru.index); 13 } 14 </pre>	<p>Enum mewakili kumpulan konstan yang membuat kode kita lebih jelas dan mudah dibaca.</p> <p>Pada fungsi <code>main()</code>, beberapa operasi dilakukan menggunakan enumerasi <code>Pelangi</code>, yaitu mencetak semua nilai yang ada di dalamnya dengan <code>Pelangi.values</code>, mencetak nilai spesifik</p>

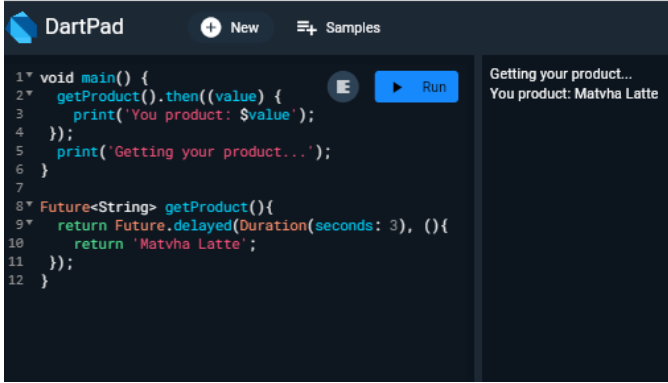
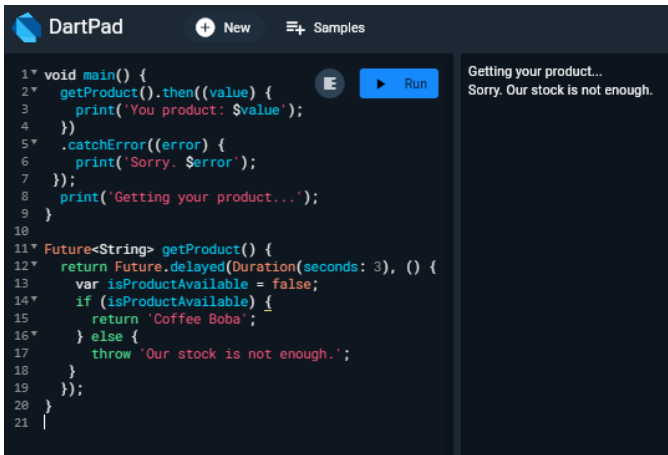
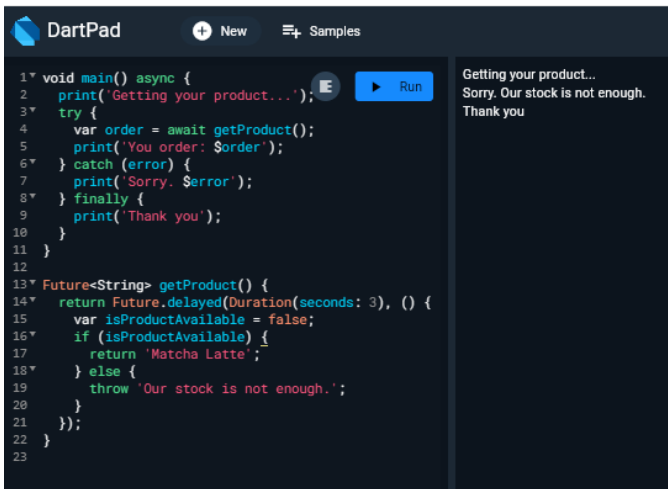
	<p>kuning, dan mencetak indeks dari nilai biru.</p>
<p>Paradigma Functional Programing</p> <ul style="list-style-type: none"> - Pure function  <pre> 1* int sum(int angka1, int angka2) { 2 return angka1 + angka2; 3 } 4 5* void main() { 6 int hasil = sum(3, 4); 7 print('Hasil penjumlahan: \$hasil'); 8 } 9 10 </pre> <p>Hasil penjumlahan: 7</p> <ul style="list-style-type: none"> - Recursion  <pre> 1* int fibonacci(int n) { 2* if (n <= 0) { 3 return 0; 4* } else if (n == 1) { 5 return 1; 6* } else { 7 return fibonacci(n - 1) + fibonacci(n - 2); 8 } 9 } 10 11* void main() { 12 int n = 10; 13 int hasil = fibonacci(n); 14 print('Nilai fibonacci ke-\$n adalah: \$hasil'); 15 } </pre> <p>Nilai fibonacci ke-10 adalah: 55</p>	<p>Pada pure function ndefinisikan sebuah fungsi bernama “sum” yang mengambil dua parameter angka1 dan angka2, keduanya bertipe data integer. Fungsi ini mengembalikan hasil penjumlahan dari kedua parameter tersebut dengan menggunakan operator +.</p> <p>Lalu pada recursion merupakan implementasi dari fungsi rekursif fibonacci, yang menghitung nilai deret Fibonacci untuk bilangan bulat positif n. Fungsi ini mengembalikan 0 jika n kurang dari atau sama dengan 0, mengembalikan 1 jika n sama dengan 1, dan jika n lebih besar dari 1, maka nilai Fibonacci dihitung dengan cara</p>

	<p>memanggil fungsi fibonacci untuk n-1 dan n-2, dan menjumlahkan hasilnya.</p>
<p>Anonymous Function</p>  <pre> 1 2* void main() { 3* var sum = (int angka1, int angka2) { 4* return angka1 + angka2; 5* }; 6 7* Function printLambda = () { 8* print('Ini adalah fungsi lambda'); 9* }; 10 11 printLambda(); 12 print(sum(3, 4)); 13 } </pre>	<p>Merupakan penerapan fungsi lambda di Dart, yang dapat digunakan untuk membuat kode lebih ringkas dan ekspresif. Kode tersebut mendefinisikan fungsi main() yang mendeklarasikan sebuah variabel sum sebagai lambda expression yang menjumlahkan dua bilangan bulat.</p>
<p>Higher-Order Function</p>  <pre> 1* void main() { 2* void contohHigherOrderFunction(String message, Function myFunction) { 3* print(message); 4* print(myFunction(3, 4)); 5* } 6 7* // Opsi 1 8* Function sum = (int num1, int num2) => num1 + num2; 9* contohHigherOrderFunction('Hello', sum); 10 11* // Opsi 2 12* contohHigherOrderFunction('Hello', (num1, num2) => num1 + num2); 13 } 14 </pre>	<p>Higher order function adalah fungsi yang menggunakan fungsi lainnya sebagai parameter, menjadi tipe kembalian, atau keduanya</p>
<p>Closures</p>  <pre> 1* void main() { 2* var contohClosure = penjumlahan(2); 3* contohClosure(); 4* contohClosure(); 5* } 6 7* Function penjumlahan(base) { 8* var s = 1; 9* return () => print('Nilainya adalah \${base + s++}'); 10 } 11 </pre>	<p>Closures adalah fungsi yang dapat mengakses variabel di dalam lexical scope-nya. Fungsi ini mendeklarasikan sebuah</p>

	<p>variabel <code>a</code> dengan nilai awal 1, dan mengembalikan sebuah closure yang mengambil nilai dari <code>base</code> dan menambahkannya dengan nilai <code>a</code>, sementara <code>a</code> sendiri bertambah setiap kali closure dipanggil.</p>
<p>Dart Type System</p> <ul style="list-style-type: none"> - Generic  <ul style="list-style-type: none"> - Type Inference 	<p>Type system adalah sistem logis yang terdiri dari seperangkat aturan yang menetapkan properti atau tipe ke berbagai konstruksi program komputer, seperti variabel, expression, fungsi, atau modul.</p> <p>Pada generic menginisialisasi tiga list, yaitu <code>bilangan</code> yang berisi bilangan bulat, <code>kata</code> yang berisi string, dan <code>dynamicList</code> yang berisi elemen dengan tipe data dinamis.</p> <p>Pada type Inference mendefinisikan sebuah map dengan nama</p>

	<p>jurusan, yang memiliki dua pasangan kunci-nilai: 'prodi' dengan nilai 'Informatika' dan 'angkatan' dengan nilai 2020.</p>
Coding Convention	
<p>- Do</p> 	<p>DO name type using UpperCamelCase .</p> <p>Class, enum, typedef, dan type parameter harus menggunakan huruf kapital pada huruf pertama dari setiap kata termasuk kata pertama</p>
<p>- Do use?</p> 	<p>Berlaku ketika sebuah expression dapat mengevaluasi nilai true, false, atau null dan programmer perlu meneruskan hasil ke suatu yang tidak menerima nilai null.</p>
<p>- Don't</p>	<p>Aturan yang diawali dengan DON'T tidak baik untuk diterapkan.</p> <p>Contohnya.</p>

 <pre> 1* void main(){ 2 var instance; // good 3 var mInstance; // bad 4 5 print('Instance: \$instance'); 6 print('MInstance: \$mInstance'); 7 } 8 9 </pre> <p>Instance: null MInstance: null</p>	<p>DON'T use prefix letters :</p> <p>“mInstance” merupakan kata prefix,</p>
<p>Dart Future</p> <p>Synchronous vs Asynchronous</p>	<p>Dalam Synchronous kode program dijalankan secara berurutan dari atas ke bawah, sedangkan program asynchronous memungkinkan suatu operasi berjalan sembari menunggu operasi lainnya selesai.</p>
<p>- Future</p>  <pre> 1* void main(){ 2* getProduct().then((result){ 3 print('Your product: \$result'); 4 }); 5 } 6 7 8* Future<String> getProduct() { 9* return Future.delayed(Duration(seconds: 3), () { 10 return 'Matcha Latte'; 11 }); 12 } 13 </pre> <p>Your product: Matcha Latte</p>	<p>Pada fungsi getProduct(), terdapat penundaan selama 3 detik sebelum mengembalikan nilai 'Matcha Latte' sebagai hasil. Kemudian, di dalam fungsi main(), hasil dari getProduct() dicetak setelah promise selesai dieksekusi</p>
<p>- Completed with data</p>	<p>Setelah Future dijalankan, perlu adanya handler untuk menangani status completed with data. Caranya dengan menggunakan method .then()</p>

 <pre> 1* void main() { 2* getProduct().then((value) { 3* print('You product: \$value'); 4* }); 5* print('Getting your product...'); 6* } 7 8* Future<String> getProduct(){ 9* return Future.delayed(Duration(seconds: 3), () { 10* return 'Matvha Latte'; 11* }); 12* } </pre>	<p>dari objek Future.</p>
<p>- Completed with error</p>  <pre> 1* void main() { 2* getProduct().then((value) { 3* print('You product: \$value'); 4* }); 5* .catchError((error) { 6* print('Sorry. \$error'); 7* }); 8* print('Getting your product...'); 9* } 10 11* Future<String> getProduct() { 12* return Future.delayed(Duration(seconds: 3), () { 13* var isProductAvailable = false; 14* if (isProductAvailable) { 15* return 'Coffee Boba'; 16* } else { 17* throw 'Our stock is not enough.'; 18* } 19* }); 20* } 21 </pre>	<p>method <code>.catchError()</code> setelah diletakkan setelah then. Hal ini dilakukan untuk mengatasi eror atau exception</p>
<p>Future dengan async-await</p>  <pre> 1* void main() async { 2* print('Getting your product...'); 3* try { 4* var order = await getProduct(); 5* print('You order: \$order'); 6* } catch (error) { 7* print('Sorry. \$error'); 8* } finally { 9* print('Thank you'); 10* } 11* } 12 13* Future<String> getProduct() { 14* return Future.delayed(Duration(seconds: 3), () { 15* var isProductAvailable = false; 16* if (isProductAvailable) { 17* return 'Matcha Latte'; 18* } else { 19* throw 'Our stock is not enough.'; 20* } 21* }); 22* } 23 </pre>	<p>Dart mempunyai keyword <code>async</code> dan <code>await</code> yang merupakan alternatif untuk dapat menuliskan proses asynchronous layaknya proses synchronous. Di dalam fungsi <code>main()</code>, pertama-tama dicetak teks "Getting your product...". Kemudian, fungsi <code>getProduct()</code> dipanggil menggunakan <code>await</code>, yang akan menunggu hingga Future selesai dieksekusi.</p>

	<p>Jika produk tersedia, maka nilai produk dicetak, jika tidak, akan dijatuhkan pengecualian dan dicetak pesan kesalahan. Akhirnya, pesan "Thank you" akan dicetak.</p>
--	---