

LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK



JUDUL :

FUNDAMENTAL DART

Disusun oleh:

Mivtakhul Janah (21102089)

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
BANYUMAS, JAWA TENGAH

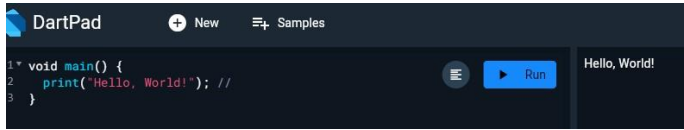
2024

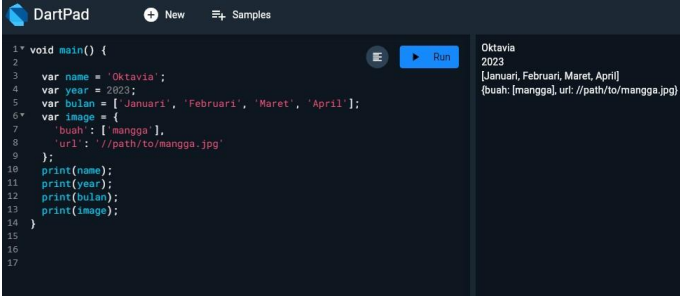
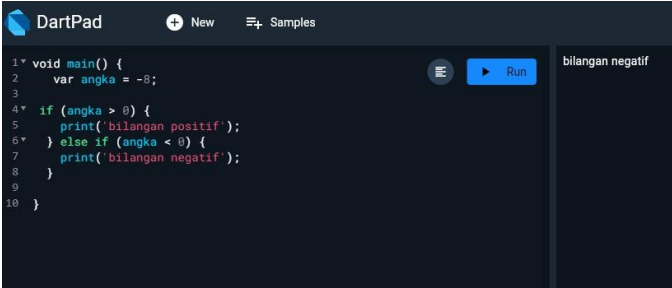
Pembahasan

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.

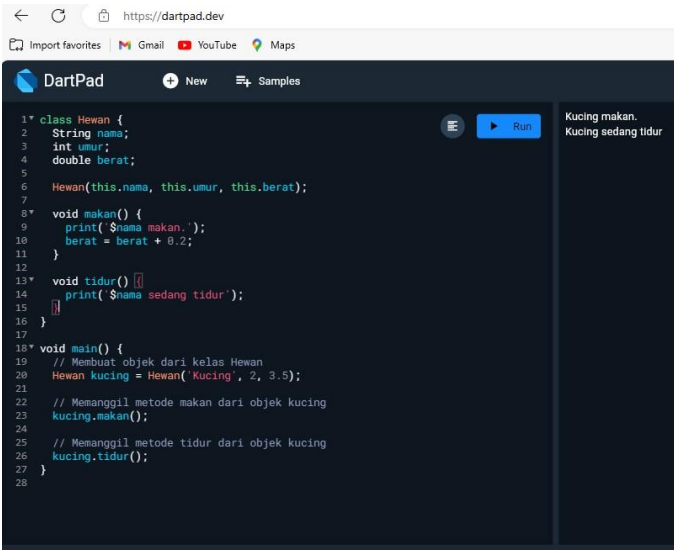
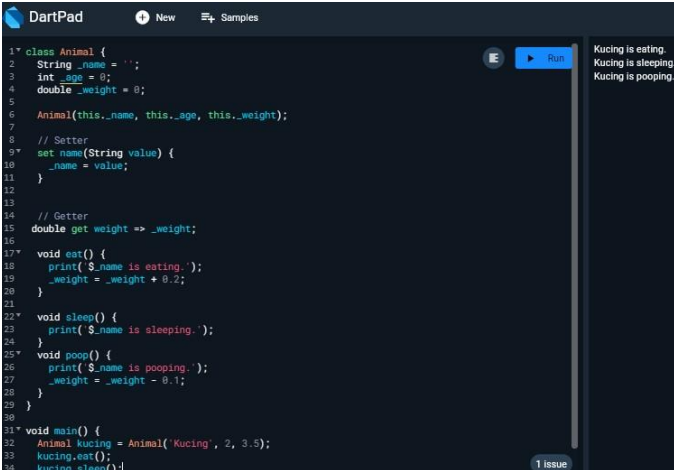
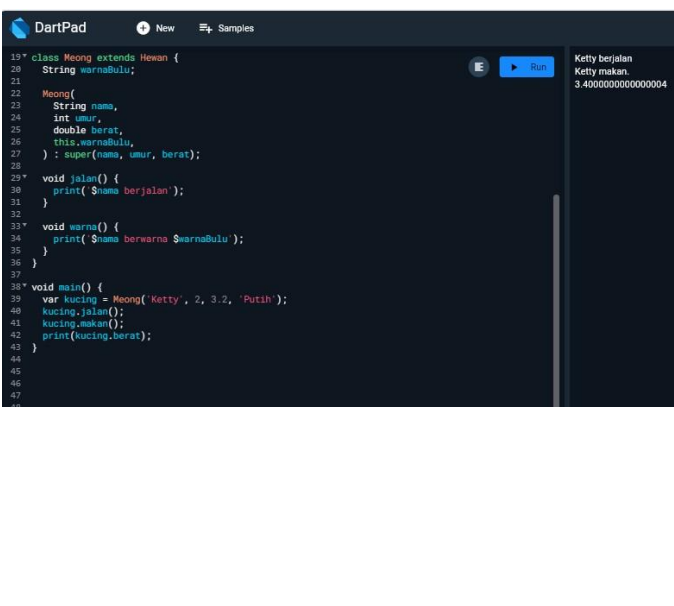
Selain itu, praktikum Fundamental Dart bertujuan untuk melatih kemampuan mahasiswa dalam menulis kode yang bersih, efisien, dan mudah dipahami. Melalui latihan-latihan yang terstruktur, mahasiswa akan diajak untuk mengimplementasikan konsep-konsep Dart dalam skenario-skenario nyata, sehingga mereka dapat 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.

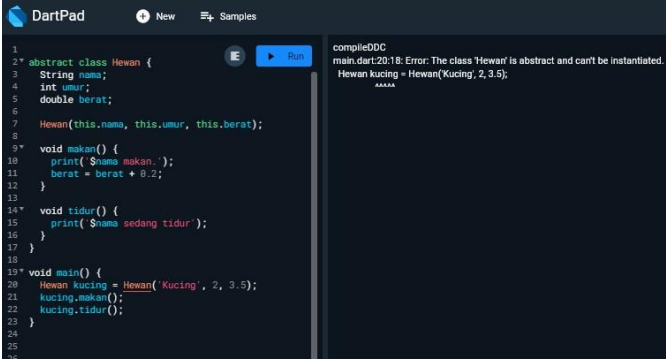
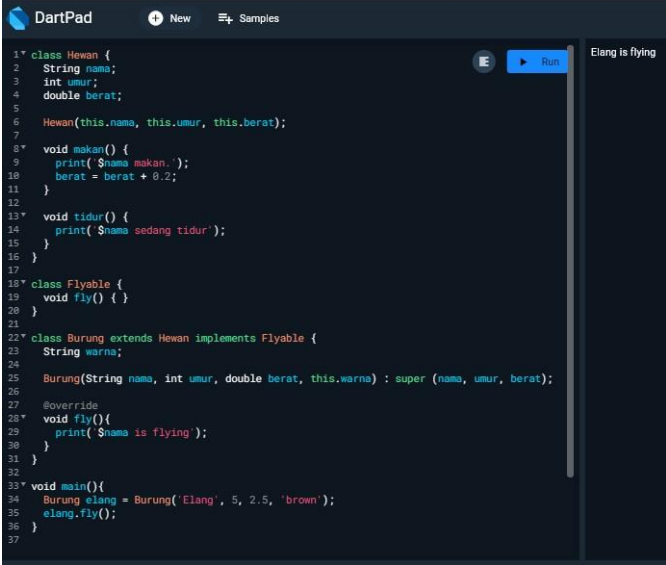
Langkah-Langkah Praktikum

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 'buah': '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> <ul style="list-style-type: none"> - If and else  <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> <ul style="list-style-type: none"> - Switch case - For loops - While and do-while - 	<p>Control flow merupakan</p>
<p>List</p>	<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>	<p>Spread operator berfungsi untuk menambah banyak</p>

	<p>nilai kedalam list dengan cara yang singkat.</p> <p>Operator ini dituliskan dengan tanda titik tiga (...).</p>
Set	<p>Set adalah kumpulan nilai unik tanpa urutan tertentu. Karena menggunakan nilai unik, maka output yang dihasilkan tidak akan terdapat nilai yang duplikat</p>
Map	<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>
Class	<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</p>

 <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 // Mengambil metode makan dari objek kucing 22 kucing.makan(); 23 // Mengambil metode tidur dari objek kucing 24 kucing.tidur(); 25 } </pre>	<p>oleh objek yang dibuat dari class tersebut</p>
<h3>Properties & method</h3>  <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 void poop() { 25 print('\$_name is pooping. '); 26 _weight = _weight - 0.1; 27 } 28 } 29 30 void main() { 31 Animal kucing = Animal('Kucing', 2, 3.5); 32 kucing.eat(); 33 kucing.sleep(); 34 kucing.poop(); 35 } </pre>	<p>Kode ini mendefinisikan sebuah kelas Animal yang memiliki atribut <code>_name</code>, <code>_age</code>, dan <code>_weight</code>. Konstruktor Animal digunakan untuk menginisialisasi nilai atribut saat objek dibuat.</p>
<h3>Inheritance</h3>  <pre> 1 class Meong extends Hewan { 2 String warnaBulu; 3 4 Meong(5 String nama, 6 int umur, 7 double berat, 8 this.warnaBulu, 9) : super(nama, umur, berat); 10 11 void jalan() { 12 print('\$nama berjalan'); 13 } 14 15 void warna() { 16 print('\$nama berwarna \$warnaBulu'); 17 } 18 } 19 20 void main() { 21 var kucing = Meong('Ketty', 2, 3.2, 'Putih'); 22 kucing.jalan(); 23 kucing.makan(); 24 print(kucing.berat); 25 } </pre>	<p>Gambar disamping mendefinisikan sebuah subclass Meong yang merupakan turunan dari superclass Hewan. Subclass ini memiliki tambahan atribut <code>warnaBulu</code> yang merupakan warna bulu dari kucing. Konstruktor Meong</p>

	<p>menginisialisasi atribut-atribut yang diwarisi dari superclass Hewan serta atribut tambahan warnaBulu.</p>
<h3>Abstract Class</h3>  <pre> 1 abstract 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 Hewan kucing = Hewan('Kucing', 2, 3.5); 20 kucing.makan(); 21 kucing.tidur(); 22 } </pre> <p>compileDDC main.dart:20:18: Error: The class 'Hewan' is abstract and can't be instantiated. Hewan kucing = Hewan('Kucing', 2, 3.5);</p>	<p>Kita telah membuat class Hewan sebelumnya, untuk menjadikan sebuah kelas menjadi abstract hanya perlu menambahkan keyword abstract sebelum penulisan kelas. Maka akan menghasilkan output error, karena kelas hewan 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 } </pre> <p>Elang is flying</p>	<p>interface pada dart dikenal sebagai <i>implicit interface</i>. Untuk mengimplementasikan interface, perlu menggunakan keyword implements.</p>
<h3>Enumerated Types</h3>	<p>Enum mewakili kumpulan konstan yang membuat kode kita lebih jelas dan</p>

```

1* enum Pelangi {
2  merah, jingga, kuning, hijau, biru, nila, ungu
3 }
4
5* enum Status {
6  Todo, 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

```

Output: [Pelangi.merah, Pelangi.jingga, Pelangi.kuning, Pelangi.hijau, Pelangi.biru, Pelangi.nila, Pelangi.ungu]
Pelangi.kuning
4

mudah dibaca.

Pada fungsi main(), beberapa operasi dilakukan menggunakan enumerasi Pelangi, yaitu mencetak semua nilai yang ada di dalamnya dengan Pelangi.values, mencetak nilai spesifik kuning, dan mencetak indeks dari nilai biru.

Paradigma Functional Programing

- Pure function

```

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

```

Output: Hasil penjumlahan: 7

- Recursion

```

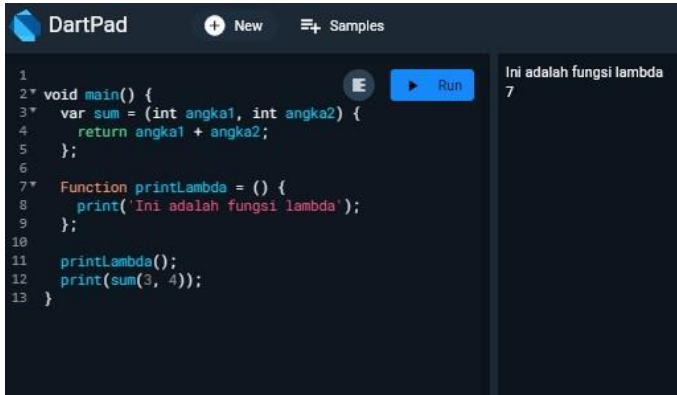
1* int fibonacci(int n) {
2*   if (n <= 0) {
3*     return 0;
4*   } else if (n == 1) {
5*     return 1;
6*   } else {
7*     return fibonacci(n - 1) + fibonacci(n - 2);
8*   }
9* }
10
11* void main() {
12  int n = 10;
13  int hasil = fibonacci(n);
14  print('Nilai fibonacci ke-$n adalah: $hasil');
15 }

```

Output: Nilai fibonacci ke-10 adalah: 55

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 +.

Lalu pada recursion merupakan implementasi dari fungsi rekursif fibonacci, yang menghitung nilai deret

	<p>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 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 men definisikan fungsi main() yang mendeklarasikan sebuah variabel sum sebagai lambda expression yang menjumlahkan dua bilangan bulat.</p>
<p>Higher-Order Function</p>	<p>Higher order function adalah fungsi yang menggunakan fungsi lainnya sebagai</p>



```
1.* void main() {
2.*   void contohHigherOrderFunction(String message, Function myFunction) {
3.*     print(message);
4.*     print(myFunction(3, 4));
5.*   }
6.*
7.*   // Ops 1
8.*   Function sum = (int num1, int num2) => num1 + num2;
9.*   contohHigherOrderFunction('Hello', sum);
10.*
11.*   // Ops 2
12.*   contohHigherOrderFunction('Hello', (num1, num2) => num1 + num2);
13.* }
14.*
```

Output: Hello, Hello, Hello

parameter, menjadi tipe kembalian, atau keduanya

Closures



```
1.* void main() {
2.*   var contohClosure = penjumlahan(2);
3.*   contohClosure();
4.*   contohClosure();
5.* }
6.*
7.* Function penjumlahan(base) {
8.*   var a = 1;
9.*   return () => print('Nilainya adalah ${base + a++}');
10.* }
11.*
```

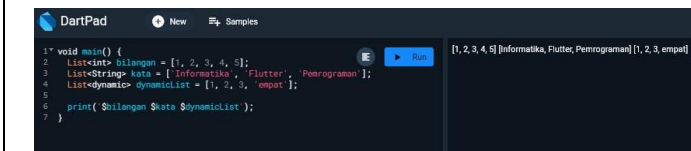
Output: Nilainya adalah 3, Nilainya adalah 4

adalah fungsi yang dapat mengakses variabel di dalam lexical scope-nya.

Fungsi ini mendeklarasikan sebuah variabel a dengan nilai awal 1, dan mengembalikan sebuah closure yang mengambil nilai dari base dan menambahkannya dengan nilai a, sementara a sendiri bertambah setiap kali closure dipanggil.

Dart Type System

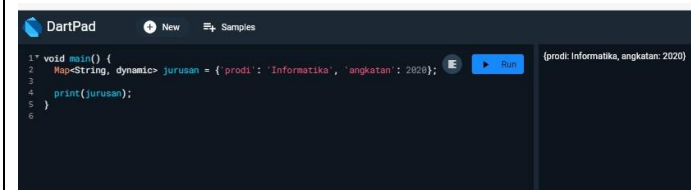
- Generic



```
1.* void main() {
2.*   List<int> bilangan = [1, 2, 3, 4, 5];
3.*   List<String> kata = ['Informatika', 'Flutter', 'Pemrograman'];
4.*   List<dynamic> dynamicList = [1, 2, 3, empat];
5.*   print('$bilangan $kata $dynamicList');
6.* }
7.*
```

Output: [1, 2, 3, 4, 5] [Informatika, Flutter, Pemrograman] [1, 2, 3, empat]

- Type Inference



```
1.* void main() {
2.*   Map<String, dynamic> jurusan = {'prodi': 'Informatika', 'angkatan': 2020};
3.*   print(jurusan);
4.* }
5.*
```

Output: {prodi: Informatika, angkatan: 2020}

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.

