Nama : Gabriel Dimas
NIM : 2141720181
KELAS : TI – 3E
Mata Kuliah : Pemrograman Mobile

Tugas Minggu 5 Bagian 4

1. Screenshoot
    a. Praktikum 1

```
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5> dart week5_1.dart
3
2
1
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5>
```

    b. Praktikum 2

```
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5> dart week5_2.dart
{fluorine, chlorine, bromine, iodine, astatine}
{}
{}
{}
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5>
```

    c. Praktikum 3

```
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5> dart week5_3.dart
{first: partridge, second: turtledoves, fifth: 1}
{2: helium, 10: neon, 18: 2}
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5>
```

    d. Praktikum 4

```
PS D:\Pertugasan\SMT 5\Mobile\code\mobile-programming-study-2023\week5> dart week5_4.dart
week5_4.dart:10:23: Warning: Operand of null-aware operation '...?' has type 'List<int>' which excludes null.
 - 'List' is from 'dart:core'.
  var list3 = [0, ...?list1];
                      ^
[1, 2, 3]
[0, 1, 2, 3]
4
[1, 2, 3]
4
[Home, Furniture, Plants, Outlet]
[Home, Furniture, Plants, Inventory]
[#0, #1, #2, #3]
```

    e. Praktikum 5
2. Fungsi digunakan untuk mengelompokkan kode secara logis, sehingga Anda dapat memanggil dan menjalankan kode tersebut kapan saja tanpa perlu menuliskannya ulang. Dalam Dart, fungsi adalah objek yang dapat disimpan dalam variabel, dikirim sebagai argumen ke fungsi lain, dan bahkan dikembalikan dari fungsi lain. Ini memungkinkan Anda untuk membuat kode yang lebih modular, mudah dipahami, dan dapat digunakan kembali.
3. Dalam bahasa Dart, terdapat beberapa jenis parameter yang dapat digunakan dalam definisi fungsi. Jenis-jenis parameter ini memungkinkan Anda untuk mengirimkan informasi ke fungsi dengan cara yang berbeda. Berikut adalah beberapa jenis parameter dalam Dart beserta contoh sintaksnya:
    a. Positional Parameters (Parameter Posisi):

```dart
void printName(String firstName, String lastName) {
  print('Nama lengkap: $firstName $lastName');
}

void main() {
  printName('John', 'Doe'); // Memanggil fungsi dengan parameter-posisi
}
```

    b. Named Parameters (Parameter Bernama):

```dart
void printName({String firstName, String lastName}) {
  print('Nama lengkap: $firstName $lastName');
}

void main() {
  printName(firstName: 'John', lastName: 'Doe'); // Memanggil fungsi dengan parameter bernama
}
```

c. Default Parameters (Parameter Default):

```dart
void greet(String name, {String greeting = 'Hello'}) {
  print('$greeting, $name!');
}

void main() {
  greet('John'); // Menggunakan nilai default untuk parameter 'greeting'
}
```

d. Required Parameters (Parameter Wajib):

```dart
void printName({required String firstName, required String lastName}) {
  print('Nama lengkap: $firstName $lastName');
}

void main() {
  printName(firstName: 'John', lastName: 'Doe'); // Memanggil fungsi dengan parameter bernama yang wajib
}
```

e. Rest Parameters (Parameter Sisa):

```dart
void printNumbers(String title, int first, int second, [int... restNumbers]) {
  print(title);
  print('First: $first');
  print('Second: $second');
  print('Rest: $restNumbers');
}

void main() {
  printNumbers('Numbers', 1, 2, 3, 4, 5); // Memanggil fungsi dengan parameter sisa
}
```

4. Dalam pemrograman, konsep "functions as first-class objects" (fungsi sebagai objek kelas pertama) mengacu pada kemampuan bahasa pemrograman untuk memperlakukan fungsi seperti objek lainnya, seperti variabel, array, atau objek lain. Dalam bahasa yang mendukung fungsi sebagai objek kelas pertama, Anda dapat mengirimkan fungsi sebagai argumen ke fungsi lain, menyimpannya dalam variabel, dan mengembalikannya sebagai hasil dari fungsi. Ini memberikan fleksibilitas yang tinggi dalam desain program dan memungkinkan Anda untuk menerapkan konsep-konsep seperti "higher-order functions" (fungsi yang mengambil atau mengembalikan fungsi lain).

   Dalam bahasa Dart, fungsi adalah first-class objects, yang berarti Anda dapat melakukan hal-hal berikut dengan fungsi:

   a. Menyimpan fungsi dalam variabel:

```dart
int add(int a, int b) {
  return a + b;
}

var functionReference = add; // Menyimpan referensi fungsi dalam variabel
```

   b. Mengirimkan fungsi sebagai argumen ke fungsi lain:

```
void executeFunction(Function myFunction) {
  myFunction(); // Memanggil fungsi yang diterima sebagai argumen
}

void sayHello() {
  print('Hello, world!');
}

executeFunction(sayHello); // Mengirimkan fungsi 'sayHello' sebagai argumen
```

c. Mengembalikan fungsi dari fungsi lain:

```
Function getMultiplier(int factor) {
  return (int number) {
    return number * factor;
  };
}

var timesTwo = getMultiplier(2); // Mengembalikan fungsi perkalian dengan faktor 2
print(timesTwo(5)); // Memanggil fungsi 'timesTwo' untuk mengalikan 5 dengan 2
```

d. Menyimpan fungsi dalam koleksi (seperti List atau Map):\

```
void main() {
  List<Function> functionList = [sayHello, () => print('Goodbye')];

  for (var func in functionList) {
    func(); // Memanggil fungsi dalam list
  }
}
```

5. Anonymous functions (fungsi anonim) adalah jenis fungsi yang tidak memiliki nama yang terkait dengan mereka. Mereka juga dikenal sebagai lambda functions atau function literals. Fungsi ini sering digunakan dalam pemrograman untuk membuat kode yang lebih ringkas dan ekspresif. Fungsi anonim biasanya digunakan dalam dua konteks utama:
   a. Sebagai argumen untuk fungsi lain: Anda dapat mengirimkan fungsi anonim sebagai argumen ke fungsi lain (biasanya dalam bentuk higher-order functions) untuk menjalankan tindakan tertentu.
   b. Sebagai ekspresi yang mengembalikan fungsi: Anda dapat mengembalikan fungsi anonim sebagai hasil dari fungsi lain.

   Berikut adalah contoh penggunaan fungsi anonim dalam Dart:
   a. Menggunakan Fungsi Anonim Sebagai Argumen Fungsi:

```
void main() {
  List<int> numbers = [1, 2, 3, 4, 5];

  // Menggunakan fungsi anonim dengan metode 'forEach' untuk mencetak setiap elemen
  numbers.forEach((number) {
    print(number);
  });
}
```

   Dalam contoh di atas, kita menggunakan fungsi anonim (number) { print(number); } sebagai argumen untuk metode forEach dari List numbers. Fungsi anonim ini akan dijalankan untuk setiap elemen dalam list dan mencetak elemen tersebut.

   b. Mengembalikan Fungsi Anonim Dari Fungsi Lain:

```
Function multiplyBy(int factor) {
  // Mengembalikan fungsi anonim yang mengalikan angka dengan faktor yang diberikan
  return (int number) => number * factor;
}

void main() {
  var timesTwo = multiplyBy(2);
  var timesThree = multiplyBy(3);

  print(timesTwo(5));   // Output: 10 (2 * 5)
  print(timesThree(7)); // Output: 21 (3 * 7)
}
```

Dalam contoh ini, fungsi multiplyBy mengembalikan fungsi anonim yang mengalikan angka dengan faktor yang diberikan. Kami kemudian menyimpan referensi ke fungsi anonim tersebut dalam variabel timesTwo dan timesThree, dan kemudian memanggilnya dengan argumen yang sesuai.

6. Lexical scope dan lexical closures adalah dua konsep yang saling terkait dalam pemrograman, tetapi memiliki makna yang berbeda. Mari kita bahas perbedaannya dan berikan contohnya:
    a. Lexical Scope (Cakupan Lexical):
    Lexical scope mengacu pada aturan bagaimana variabel di dalam fungsi dapat diakses berdasarkan posisi di dalam kode sumber (lexically) atau berdasarkan tempat variabel tersebut dideklarasikan di dalam kode. Dalam cakupan lexical, akses ke variabel ditentukan oleh tempat di mana variabel itu berada dalam kode, terlepas dari waktu eksekusi program.

```
void outerFunction() {
  int outerVariable = 10;

  void innerFunction() {
    print(outerVariable); // Variabel 'outerVariable' dapat diakses di dalam 'innerFunction' karena berada dalam cakupan lexical yang lebih tinggi.
  }

  innerFunction();
}

void main() {
  outerFunction();
}
```

    b. Lexical Closures (Penutupan Lexical):
    Lexical closures adalah kemampuan sebuah fungsi untuk "mengingat" lingkungan (termasuk variabel) di mana ia didefinisikan, bahkan setelah fungsi itu selesai dieksekusi. Dengan kata lain, fungsi yang memiliki closure dapat mengakses variabel-variabel yang ada di sekitarnya, bahkan setelah fungsi tersebut tidak lagi aktif.

```
Function makeCounter() {
  int count = 0;

  return () {
    count++;
    print(count);
  };
}

void main() {
  var counter = makeCounter();

  counter(); // Output: 1
  counter(); // Output: 2
  counter(); // Output: 3
}
```

7. Contoh cara membuat return multiple value di Functions

```
class MultipleValues {
  final int value1;
  final String value2;

  MultipleValues(this.value1, this.value2);
}

MultipleValues returnMultipleValues() {
  int number = 42;
  String text = "Hello, World!";

  return MultipleValues(number, text);
}

void main() {
  MultipleValues result = returnMultipleValues();

  print("Value 1: ${result.value1}");
  print("Value 2: ${result.value2}");
}
```

8. Link repo github
   https://github.com/gabrieldimas/mobile-programming-study-2023/tree/main/week5