Jobsheet 4

Praktikum 1

1.

```
void main(){
  var list = [1, 2, 3];
  assert(list.length == 3);
  assert(list[1] == 2);
  print(list.length);
  print(list[1]);

  list[1] = 1;
  assert(list[1] == 1);
  print(list[1]);
}
```

2. Pada baris pertama menginisialisasikan variabel list bervalue, kemudian dilakukan assert yang mana berfungsi untuk memeriksa suatu kebenaran terhadap kondisi yang ditentukan, dan jika kondisi tidak terpenuhi maka akan tampil error message. Dengan menjalankan perintah 'dart --enable-asserts jobsheet4.dart' maka akan mengaktifkan assert tersebut.
   List.length untuk mengetahui panjang value dari list, dan list[1] berarti membuka nilai dari list index ke 1 yang mana bernilai 2

3.

```
import 'dart:io';

void main() {
  final list = [];
  list.length=5;
  list[1] = 'Fiki Suganda';
  list[2] = 2141720111;
  assert(list.length == 5);
  print(list.length);
  print(list[1]);
  print(list[2]);
}
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart --enable-asserts jobsheet4.dart
5
Fiki Suganda
2141720111
```

Praktikum 2

1.

```
void main() {

  var halogens = {'fluorine', 'chlorine', 'bromine', 'iodine', 'astatine'};
  print(halogens);
}
```

2.

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
{fluorine, chlorine, bromine, iodine, astatine}
```

Terdapat variabel halogens dengan tipe data set dengan elemen seperti diatas, kemudian variabel tersebut dicetak

3.

```
void main() {


var halogens = {'fluorine', 'chlorine', 'bromine', 'iodine', 'astatine'};
print(halogens);

var names1 = <String>{};
Set<String> names2 = {}; // This works, too.
var names3 = {}; // Creates a map, not a set.

names1.add('Fiki Suganda');
names1.add('2141720111');

names2.addAll(names1);

print(names1);
print(names2);
print(names3);
}
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
{fluorine, chlorine, bromine, iodine, astatine}
{Fiki Suganda, 2141720111}
{Fiki Suganda, 2141720111}
{}
```

Praktikum 3

1.

```dart
void main() {

  var gifts = {
    // Key:    Value
    'first': 'partridge',
    'second': 'turtledoves',
    'fifth': 1
  };

  var nobleGases = {
    2: 'helium',
    10: 'neon',
    18: 2,
  };

  print(gifts);
  print(nobleGases);
}
```

2.

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
{first: partridge, second: turtledoves, fifth: 1}
{2: helium, 10: neon, 18: 2}
```

Menginisialisasikan variabel gifts dengan tipe data map, dengan mencoba tipe data string dan int pada key dan value. Dan pada baris akhir di cetak

3.

```dart
  var mhs1 = Map<String, String>();

  gifts['first'] = 'partridge';
  gifts['second'] = 'turtledoves';
  gifts['fifth'] = 'golden rings';

  var mhs2 = Map<int, String>();
  nobleGases[2] = 'helium';
  nobleGases[10] = 'neon';
  nobleGases[18] = 'argon';

  print(gifts);
  print(nobleGases);
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
{first: partridge, second: turtledoves, fifth: 1}
{2: helium, 10: neon, 18: 2}
{first: partridge, second: turtledoves, fifth: golden rings}
{2: helium, 10: neon, 18: argon}
```

Kode di atas cuma merepresentasikan pembuatan variabel dengan tipe data map dengan cara yang berbeda, yaitu dengan menginisialisasikan tipedata key dan value di awal

Praktikum 4

1.

```
var list = [1, 2, 3];

var list2 = [0, ...list];
print(list);
print(list2);
print(list2.length);
```

2.

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
[1, 2, 3]
[0, 1, 2, 3]
4
```

Dengan ...list , dapat menyatukan/menggabungkan isi list ke list tujuan

3.

```
var list1 = ['1', '2', null];

print(list1);
var list3 = [0, ...?list1];
print(list3.length);
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
jobsheet4.dart:12:23: Warning: Operand of null-aware operation '...?' has type 'List<String?>' which excludes null.
 - 'List' is from 'dart:core'.
  var list3 = [0, ...?list1];
                      ^
[1, 2, null]
4
```

Muncul warning tetapi null masih teridentifikasi dan list3.lenght = 4 karena null terbaca

4.

```
var promoActive = true;

var nav = ['Home', 'Furniture', 'Plants', if (promoActive) 'Outlet'];
```

```
  print(nav);
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
[Home, Furniture, Plants, Outlet]
```

jika

```
var promoActive = false;
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
[Home, Furniture, Plants]
```

5.
6.

```
var listOfInts = [1, 2, 3];

  var listOfStrings = ['#0', for (var i in listOfInts) '#$i'];
  assert(listOfStrings[1] == '#1');
  print(listOfStrings);
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart --enable-asserts jobsheet4.dart
[#0, #1, #2, #3]
```

jadi collection list dapat dijalankan perulangan didalamnya

Praktikum 5

1.

```
var record = ('first', a: 2, b: true, 'last');

  print(record);
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
(first, last, a: 2, b: true)
```

2. **Membuat variabel bertipe data records**
3.

```
void main() {

  var record = ('first', a: 2, b: true, 'last');
  print(record);
```

```
  var result = tukar((1, 2));
  print('Setelah ditukar: $result');
}

(int, int) tukar((int, int) record) {
  var (a, b) = record;
  return (b, a);
}
```

```
PS E:\kuliah\Kelas\semester 5\Mobile\P2\hello_world\lib> dart jobsheet4.dart
(first, last, a: 2, b: true)
Setelah ditukar: (2, 1)
```

Dibuat fungsi tukar berparameter untuk menukar nilai

4.

```
// Record type annotation in a variable declaration:

  (String, int) mahasiswa;
  print(mahasiswa = ('asd', 1));
```

```
(asd, 1)
```

5.

```
var mahasiswa2 = ('first', a: 2, b: true, 'last');


  print(mahasiswa2.$1); // Prints 'first'
  print(mahasiswa2.a); // Prints 2
  print(mahasiswa2.b); // Prints true
  print(mahasiswa2.$2); // Prints 'last'
```

```
first
2
true
last
```

Tugas Praktikum

1.
2. Function atau method adalah suatu potongan kode yang dapat dipanggil di fungsi main dan dapat diberi parameter maupun tidak
3.
    1. **Positional Parameters:**
        - Parameter-posisi adalah jenis parameter yang menerima argumen berdasarkan posisinya.
        - Contoh sintaksis:

    void printInfo(String name, int age) { print('Name: $name, Age: $age'); } // Pemanggilan fungsi dengan parameter-posisi printInfo('John', 25);
    2. **Named Parameters:**
        - Parameter-nama adalah jenis parameter yang memungkinkan Anda memberikan nilai ke parameter berdasarkan namanya, sehingga Anda dapat mengabaikan urutan posisinya.
        - Contoh sintaksis:

    void printInfo({String name, int age}) { print('Name: $name, Age: $age'); } // Pemanggilan fungsi dengan parameter-nama
    printInfo(name: 'John', age: 25);
    3. **Default Parameters:**
        - Parameter default adalah jenis parameter yang memiliki nilai default jika tidak diberikan saat pemanggilan fungsi.
        - Contoh sintaksis:

    void printInfo(String name, {int age = 25}) { print('Name: $name, Age: $age'); } // Pemanggilan fungsi tanpa memberikan nilai untuk age (menggunakan nilai default)
    printInfo('John');
    4. **Positional Parameters with Default Values:**
        - Kombinasi parameter-posisi dan parameter default.
        - Contoh sintaksis:

    void printInfo(String name, [int age = 25]) { print('Name: $name, Age: $age'); } // Pemanggilan fungsi tanpa memberikan nilai untuk age (menggunakan nilai default)
    printInfo('John');
    5. **Required Parameters:**
        - Parameter yang wajib harus diberikan nilai saat pemanggilan fungsi.
        - Contoh sintaksis:

    void printInfo(String name, int age) { print('Name: $name, Age: $age'); } // Pemanggilan fungsi yang menyertakan nilai untuk semua parameter yang wajib
    printInfo('John', 25);
    6. **Dynamic Parameters (Use of ...):**
        - Parameter yang dapat menangani jumlah argumen yang tidak terbatas.
        - Contoh sintaksis:

    void printNames(String first, String second, String third, [String... rest]) { print('First: $first, Second: $second, Third: $third'); print('Rest: $rest'); } // Pemanggilan fungsi dengan argumen dinamis

printNames('John', 'Doe', 'Smith', 'Alice', 'Bob');

7. **Function as a Parameter:**
   - Parameter yang berupa fungsi (Higher-Order Function).
   - Contoh sintaksis:

   void executeFunction(void Function() func) { print('Executing function...'); func(); } // Pemanggilan fungsi dengan fungsi sebagai parameter
   executeFunction(() { print('Hello from the passed function!'); });
8. **Generic Parameters:**
   - Jenis parameter yang memungkinkan fungsi bekerja dengan tipe data yang dapat ditentukan.
   - Contoh sintaksis:

   T identity<T>(T value) { return value; } // Pemanggilan fungsi generic
   String result = identity<String>('Hello');

4. fungsi dapat disimpan dalam variabel, dikirim sebagai argumen ke fungsi lain, dan dapat dikembalikan sebagai nilai dari suatu fungsi

```
// Fungsi sebagai nilai
void sayHello() {
  print('Hello!');
}

void main() {
  // Menyimpan fungsi dalam variabel
  var myFunction = sayHello;

  // Memanggil fungsi melalui variabel
  myFunction();  // Output: Hello!

  // Fungsi sebagai argumen
  void say(String message) {
    print(message);
  }

  void greet(Function myFunction, String name) {
    myFunction('Hello, $name!');
  }

  // Mengirim fungsi sebagai argumen
  greet(say, 'John');  // Output: Hello, John!

  // Fungsi sebagai nilai kembalian
  Function multiplier(int factor) {
    return (int value) => value * factor;
  }
```

```
  // Mengembalikan fungsi dari fungsi lain
  var times2 = multiplier(2);
  print(times2(5));  // Output: 10

  // Fungsi sebagai data struktur
  void sayGoodbye() {
    print('Goodbye!');
  }

  // Menyimpan fungsi dalam List
  List<Function> greetings = [sayHello, sayGoodbye];

  // Memanggil fungsi dari List
  greetings.forEach((greeting) => greeting());
  // Output:
  // Hello!
  // Goodbye!

  // Fungsi anonim (Lambda)
  // Menyimpan fungsi anonim dalam variabel
  var myAnonymousFunction = () {
    print('Hello from anonymous function!');
  };

  // Memanggil fungsi anonim melalui variabel
  myAnonymousFunction();  // Output: Hello from anonymous function!
}
```

5. **Anda dapat membuat fungsi anonim (tanpa nama) dan menyimpannya dalam variabel**

```
var myAnonymousFunction = () {
    print('Hello from anonymous function!');
  };

  // Memanggil fungsi anonim melalui variabel
  myAnonymousFunction();  // Output: Hello from anonymous function!
```

6. Lexical scope, ketika sebuah fungsi didefinisikan, lingkup variabelnya sudah ditentukan berdasarkan struktur kode sumber, dan tidak berubah selama eksekusi program.

```
void main() {
  var outsideVariable = 'I am outside!';

  void myFunction() {
```

```
    print(outsideVariable);  // Variabel outsideVariable diakses dari dalam
fungsi
  }

  myFunction();
}
```

Dalam contoh ini, **myFunction** dapat mengakses variabel **outsideVariable** yang berada di luar fungsi tersebut. Ini karena variabel tersebut berada dalam cakupan (scope) lexikal dari **myFunction**.

Lexical closures adalah konsep di mana fungsi dapat menyimpan referensi ke variabel di lingkup luarannya, bahkan setelah fungsi tersebut selesai dieksekusi. Ini berarti fungsi tersebut membawa "tutupan" dari lingkup lexikal di mana ia dibuat.

```
Function outerFunction(int x) {
  void innerFunction(int y) {
    print(x + y);  // x adalah variabel dari lingkup lexikal outerFunction
  }

  return innerFunction;
}

void main() {
  var closure = outerFunction(5);

  // Closure masih memiliki akses ke variabel x meskipun outerFunction sudah
selesai dieksekusi
  closure(3);  // Output: 8
}
```

Dalam contoh ini, **innerFunction** adalah fungsi yang mengandung **closure** karena ia memiliki akses ke variabel **x** dari lingkup lexikal **outerFunction** bahkan setelah **outerFunction** sudah selesai dieksekusi. Saat **outerFunction(5)** dipanggil, ia mengembalikan referensi ke **innerFunction** yang menyimpan **closure** ke variabel **x**.

7.

```
// Fungsi mengembalikan List

List<int> getNumbers() {
  return [1, 2, 3, 4, 5];
}
```

```
// Fungsi mengembalikan Map
Map<String, dynamic> getUserInfo() {
  return {
    'name': 'John Doe',
    'age': 25,
    'city': 'Anytown',
  };
}

// Class untuk mengemas data
class Point {
  int x;
  int y;

  Point(this.x, this.y);
}

// Fungsi mengembalikan instance dari class Point
Point getCoordinates() {
  return Point(3, 7);
}

// Fungsi menggunakan pustaka tuple untuk mengembalikan multiple values
Tuple2<int, String> getValues() {
  return Tuple2(42, 'Hello');
}

void main() {
  // Menggunakan List
  var numbers = getNumbers();
  print(numbers);  // Output: [1, 2, 3, 4, 5]

  // Menggunakan Map
  var userInfo = getUserInfo();
  print(userInfo);  // Output: {name: John Doe, age: 25, city: Anytown}

  // Menggunakan Class
  var coordinates = getCoordinates();
  print('X: ${coordinates.x}, Y: ${coordinates.y}');  // Output: X: 3, Y: 7

  // Menggunakan Tuple
  var values = getValues();
  print('Number: ${values.item1}, Greeting: ${values.item2}');  // Output:
Number: 42, Greeting: Hello
}
```