Dart

1. Kiểu dữ liệu (Tất cả đều là object nên là null)

+Number (int , double , num(dung chung giữa int và double))

+Strings

+Booleans

+List

+Maps

+Runes(Unicode point)

(https://en.wikipedia.org/wiki/List\_of\_Unicode\_characters)

* Ví dụ

Runes runes = Runes(

'\u{2651} \u{2652} \u{2653} \u{2654} '

);

print(String.fromCharCodes(runes));

+Symbol

1. Khai báo biến

**var a = 'Hello World';**

+ Var sẽ tự suy ra kiểu dữ liệu của giá trị

+ Nếu gán a bằng khác kiểu dữ liệu thì không được vì a đã có kiểu giá trị từ khi định nghĩa ra a

**dynamic a = 'Hello World';**

**a = 10;**

+ Khai báo này thì biến a có thể thay đổi kiểu dữ liệu

1. Check type (runtimeType)

var number = "123";

number.runtimeType;

1. String buffer

Khi nối chuỗi ta có thể dùng String buffer để nối đơn giản hơn

StringBuffer buffer = new StringBuffer();

buffer.write("1");

buffer.write("2");

buffer.write("3");

print(buffer.toString());

1. Dart String Interpolation

“ Pham tấn phát ${age}”

1. Multiple line

String multipleLine =

'SELECT \* '

'FROM table\_name'

'Where id = 1';

1. Final và const

+Final : thường được sử dụng trong quá trình runtime

var a = 10;

final PI = a;

+const : chỉ sử dụng khi giá trị được complie rồi

var a = 10;

//Giá trị a được tạo ra trong quá trình run time

const PI = a;

1. Toán tử

// ?? check null

var a;

if (a == null){

a = "abc";

}

print(a);

var b;

b ??= "def";

print(b);

// is check type

var a = 1;

if(a is int){

print("Type is number");

}

// cascade ..

var person = Person()

..name = "Nguyen Van teo"

..email = "nguyenvanteo@gmail.com";

print(person);

// as Typecast

var example = "123";

(example as Person).email = "a";

1. Câu điều kiện

// if else

var i = 11;

if (i % 2 == 0){

print("i chia hết cho 2");

}else {

print("i không chia hết cho 2");

}

// switch case

var status = "CONFIRM";

switch(status){

case "CONFIRM" :

print("status confirm");

break;

case "PENDING" :

print("status pending");

break;

default:

print("Unspecified state");

break;

}

1. Assert

var a;

assert(a != null , "number không được bằng null");

1. Function

+Nếu không có giá trị trả về mặc định trả về null

1. Positional option parameter

+ Tham số truyền vào cho function được đặt trong [] có thể truyền hoặc không truyền.

+Nếu không truyền các giá trị sẽ là null

void main() {

showCountry("Phat");

}

void showCountry([String city1 , String city2]){

print(city1);

print(city2);}

1. Optional name parameter

+ Tham số truyền vào sẽ được gắn theo name và có thể đặt ở đâu cũng được

+ Các tham số được đặt trong {}

+ Có thể gắn các giá trị default nếu không truyền vào

void main() {

showCountry(city1: "Vũng tàu",city2: "Đà lạt");

}

void showCountry({String city1 = “Sài gòn” , String city2}){

print(city1);

print(city2);

}

1. List

Có 2 cách khai báo list

+ Cách 1 : var list = List(5);

+ Cách 2 : var list1 = [1,2,3,4];

Ngoài ra dart hỗ trợ các collection if

var age = 19;

var languages = [

"java",

"c++",

"c#",

if (age > 18) 'dart'

];

print(languages);

Sử dụng collection for

var listOfString = [

for (var i = 0 ; i < 10 ; i++) "item $i"

];

print(listOfString);

1. Set

Sử dụng mảng set cho các giá trị không trùng lập với nhau

var sets = {1,2,3,2,4,3};

print(sets);

1. Map

Sử dụng map cho việc lưu các giá trị dưới dạng key và value

var users = {

"name" : "Nguyen Van Teo",

"age" : 10

};

print(users["name"]);

1. Handling Error

+Try on dùng cho biết lỗi là gì

try{

int a = 12 ~/ 0;

print(a);

}on IntegerDivisionByZeroException{

print("Lỗi chia với 0");

}

+Try catch dùng cho chưa biết lỗi cụ thể

1. Hướng đối tượng

Khi kế thừa lớp con không kế thừa lại constructor của cha

+ Cách kế thừa lớp cha

class **Person** {

final \_name;

**Person**(this.\_name);

}

class **Musican** extends **Person** {

**Musican**(**String** name): super(name);}

+ Named constructors

class **Point** {

num x, y;

Point(this.x, this.y);

*// Named constructor*

Point.origin() {

x = 0;

y = 0;

}}

Trong trường hợp kế thừa

class **Person** {

**String** firstName;

**Person**.fromJson(**Map** data) {

print('in **Person**');

}

}

class **Employee** extends **Person** {

*// Person does not have a default constructor;*

*// you must call super.fromJson(data).*

**Employee**.fromJson(**Map** data) : super.fromJson(data) {

print('in **Employee**');

}}

+ with :

Nếu muốn đa kết thừa class ta dùng từ khóa with.Khi 1 lớp được kế thừa lại nhiều lần như Musical thì ta có thể dùng with có nghĩa lớp này sẽ được kế thừa lại nhiều nơi và đặc biệt nó sẽ không được có constructor

class Musician extends Performer with Musical {

// ···

}

class Maestro extends Person

with Musical, Aggressive, Demented {

Maestro(String maestroName) {

name = maestroName;

canConduct = true;

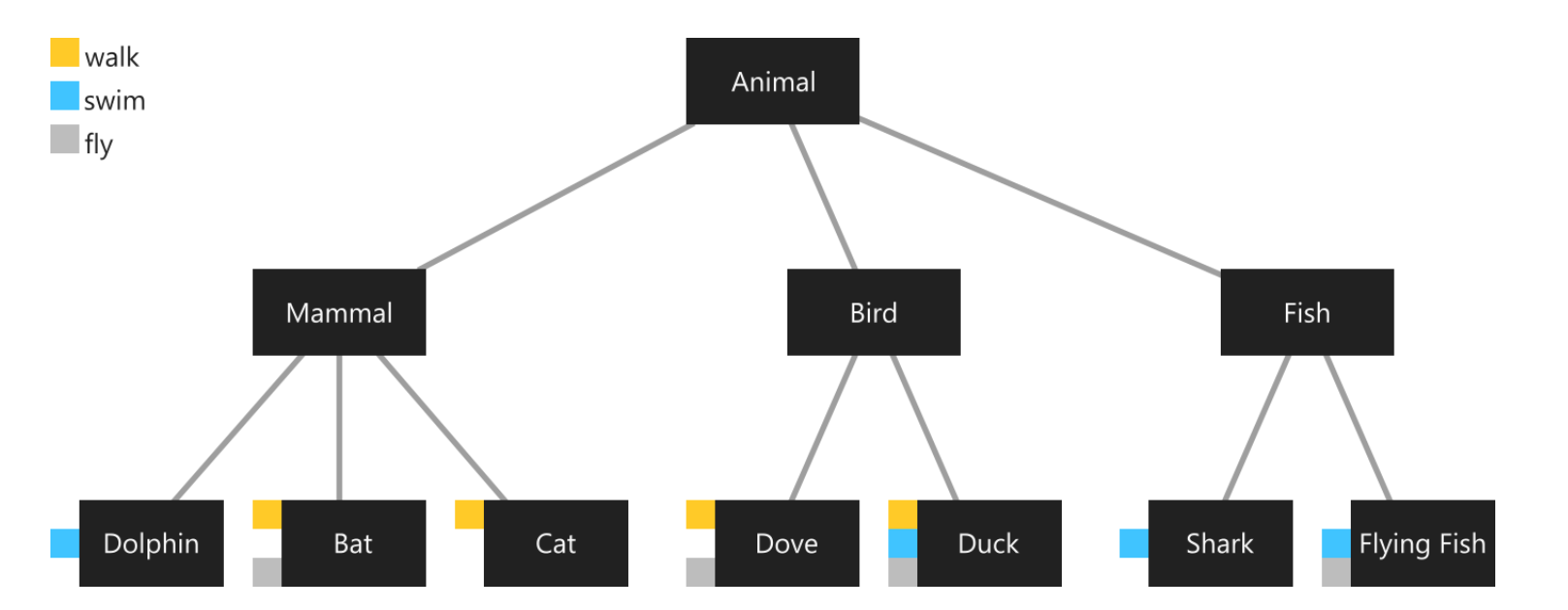
}

}

+ mixin :

Đây là 1 lớp đặc biệt dùng cho định nghĩa các feature có thể dùng chung cho các lớp

Ví dụ :



Ta thấy các lớp con sẽ kế thừa lại các chức năng như fly , walk , fly vì thế nếu ta định nghĩa ra 1 class rồi cho các lớp con kế thừa lại thì không tối ưu

Code mẫu

class Dolphin extends Manual with Walker{

@override

void walk() {

// TODO: implement walk

}

}

mixin Walker{

void walk();

}