**Nhóm 5**

Thành viên: Phạm Lê Nhật Quang

Hà Thị Thùy Trang

Nguyễn Thanh Tùng

1. **Tính tổng hiệu tích thương**

int a = 1;  
int b = 2;  
int tong = a + b;  
int hieu = a - b;  
int tich = a \* b;  
dynamic thuong = a / b;  
  
print("tong: "+ tong.toString());  
print("hieu: "+ tong.toString());  
print("tich: "+ tong.toString());  
print("thuong: "+ tong.toString());

Kết quả:

A computer screen shot of a program

Description automatically generated

1. **Bài tập 2**

void main() {  
 print(numberToWords(1235678));  
}  
  
String numberToWords(int number) {  
 if (number == 0) {  
 return "Không";  
 }  
  
 String result = "";  
  
 // Hàng đơn vị  
 List<String> unit = [  
 "", "một", "hai", "ba", "bốn", "năm", "sáu", "bảy", "tám", "chín"  
 ];  
  
 // Hàng chục  
 List<String> tens = [  
 "", "mười", "hai mươi", "ba mươi", "bốn mươi", "năm mươi", "sáu mươi", "bảy mươi", "tám mươi", "chín mươi"  
 ];  
  
 // Hàng trăm  
 List<String> hundreds = [  
 "", "một trăm", "hai trăm", "ba trăm", "bốn trăm", "năm trăm", "sáu trăm", "bảy trăm", "tám trăm", "chín trăm"  
 ];  
  
 // Hàng nghìn  
 List<String> thousands = [  
 "", "nghìn", "triệu", "tỷ", "nghìn tỷ", "triệu tỷ", "tỷ tỷ"  
 ];  
  
 int count = 0;  
  
 while (number > 0) {  
 int temp = number % 1000;  
  
 if (temp != 0) {  
 String tempStr = "";  
  
 if (temp ~/ 100 > 0) {  
 tempStr += hundreds[temp ~/ 100] + " ";  
 }  
  
 if (temp % 100 > 10 && temp % 100 < 20) {  
 tempStr += "mười ";  
 tempStr += unit[temp % 10] + " ";  
 } else {  
 if ((temp % 100 ~/ 10) > 0) {  
 tempStr += tens[temp % 100 ~/ 10] + " ";  
 }  
  
 if ((temp % 10) > 0) {  
 tempStr += unit[temp % 10] + " ";  
 }  
 }  
  
 tempStr += thousands[count] + " ";  
 result = tempStr + result;  
 }  
  
 count++;  
 number ~/= 1000;  
 }  
  
 return result.trim();  
}

Kết quả:

A screenshot of a computer program

Description automatically generated

1. **Các dạng phương thức trong dart**

Phương thức là gì?

Trong ngôn ngữ lập trình Dart, phương thức là một khối mã (block of code) được định nghĩa bên trong một lớp và chỉ chạy khi được gọi. Các phương thức chia nhiệm vụ lớn thành các phần nhỏ và thực hiện một hoạt động cụ thể của chương trình. Quá trình này làm tăng khả năng tái sử dụng mã và nâng cao cách tiếp cận mô-đun của chương trình.

Về cơ bản, phương thức được chia làm 3 loại:

* Phương thức thông thường (không tĩnh và không trừu tượng).
* Phương thức tĩnh (static method).
* Phương thức trừu tượng (abstract method).

1. Phương thức thông thường

Giống như 1 phương thức thông thường hay sử dụng trong các ngôn ngữ nó bao gồm

* return\_type: Kiểu dữ liệu trả về của phương thức. Sử dụng từ khoá void như một kiểu trả về nếu phương thức không trả về gì cả.
* method\_name: Tên phương thức. Khác với các ngôn ngữ khác như Java, Dart không cho phép 2 phương thức cùng tên, kể cả khi chúng có các tham số khác nhau. Điều này là cần thiết để có thể chuyển đổi mã Dart sang JavaScript.
* arguments: Các tham số của phương thức.

**return\_type methodname (arguments) {**

**// statement(s)**

**}**

Ví dụ: void SaySomeThing(string value){

Print(“$value”)

}

1. Phương thức tĩnh

Dart sử dụng từ khoá static và cú pháp định nghĩa một phương thức thông thường để định nghĩa một phương thức tĩnh.

Các đặc điểm của phương thức tĩnh:

* Phương thức tĩnh được gọi thông qua tên lớp và ký hiệu dấu chấm. Tên lớp và ký hiệu dấu chấm có thể được bỏ qua nếu gọi phương thức tĩnh trong nội bộ lớp đó.
* Các thành viên không tĩnh của một lớp không thể xuất hiện trong phương thức tĩnh trừ khi chúng được truy cập thông qua đối tượng

**static return\_type method\_name(arguments) {**

**// statement(s)**

**}**

Ví dụ: static int sum(int a, int b) {

return a + b;

}

1. Phương thức trừu tượng

Trong ngôn ngữ Dart, phương thức trừu tượng là một phương thức không tĩnh và không có thân. Một lớp có ít nhất một phương thức trừu tượng phải được khai báo là trừu tượng. Một trong các lớp con của nó sẽ ghi đè (override) các phương thức trừu tượng và viết nội dung cho chúng.

**abstract return\_type method\_name(arguments);**

Ví dụ:

abstract class Person {

void sayHello();

}

class EnglishPerson extends Person {

void sayHello() {

print("Hello");

}

}

1. Các tham số tùy chọn

các lớp trong Dart và JavaScript không hỗ trợ các phương thức trùng tên, nhưng một phương thức có thể bao gồm các tham số tuỳ chọn.

**return\_type method\_name(typeM argM, typeN argN,[typeP? argP, typeQ? argQ]){**

**// statement(s)**

**}**

Ví dụ:

class test {

static double sum(double v1, double v2, [double? v3, double? v4]) {

return v1 + v2 + (v3 ?? 0) + (v4 ?? 0);

}

}

void main() {

double value1 = test.sum(1, 2, 3, 4);

double value2 = test.sum(1, 2, 3);

double value3 = test.sum(1, 2);

}

1. Các tham số tùy chọn được đặt tên

Các tham số tuỳ chọn được đặt tên (Optional Named Parameters) là các tham số được đặt trong cặp dấu ngặc { } và là các tham số cuối cùng trong danh sách các tham số. Tất cả các tham số này đều có giá trị mặc định.

**return\_type method\_name(typeM argM, typeN argN,**

**{typeP paramP = defaultValueP, typeQ paramQ = defaultValueQ}){**

**// statement(s)**

**}**

Ví dụ:

class StringConcat {

static String ConcatAndTrim(String s1, String s2, {bool trimLeft = true, bool trimRight = true}) {

var s = s1 + s2;

if (trimLeft && trimRight) {

return s.trim();

} else if (trimLeft) {

return s.trimLeft();

} else if (trimRight) {

return s.trimRight();

}

return s;

}

}

void main() {

var s1 = ' One ';

var s2 = ' Two ';

var result1 = StringConcat.ConcatAndTrim(s1, s2);

var result2 = StringConcat.ConcatAndTrim(s1, s2, trimLeft: false);

var result3 = StringConcat.ConcatAndTrim(s1, s2, trimRight: false);

var result4 = StringConcat.ConcatAndTrim(s1, s2, trimLeft: false, trimRight: false);

}

1. Ghi đè 1 phương thức

Lớp con có thể ghi đè (override) một phương thức của lớp cha nếu các điều kiện sau được thoả mãn:

* Hai phương thức phải cùng tên và có các tham số giống nhau.
* Kiểu trả về của 2 phương thức phải giống nhau hoặc kiểu trả về của phương thức tại lớp con phải là kiểu con của kiểu trả về của phương thức tại lớp cha.

Ví dụ:

class Animal {

void sayAnything() {

print('Nothing');

}

}

class Cat extends Animal {

@override

void sayAnything() {

print('Meooow');

}

}

void main() {

Animal animal = Cat();

animal.sayAnything();

}

1. Final method

Trong ngôn ngữ Java, bạn có thể sử dụng từ khoá final cho một phương thức để đảm bảo rằng các lớp con không thể ghi đè (override) phương thức này. Vậy có gì đó tương tự trong Dart không?

package:meta cung cấp cho bạn @nonVirtual để ngăn chặn việc một lớp con ghi đè một phương thức nào đó của lớp cha. Nó giúp Analyzer (trình phân tích mã) thông báo lỗi nếu bạn vi phạm điều đó, nhưng không có hành động nào khác được thực hiện.

ví dụ:

import 'package:meta/meta.dart';

class Test {

@nonVirtual

void SaySomething() {

print("test thu nonvirtual");

}

}

Class Test1 extends Test {

@override

void SaySomething () {

print("test thu nonvirtual o test 1");

}

}

bạn sẽ nhận được một cảnh báo (warning) từ Analyzer giống như thế này khi thực hiện điều trên:

The member 'SaySomething' is declared non-virtual in 'Test'

and can't be overridden in subclasses.dart(invalid\_override\_of\_non\_virtual\_member)