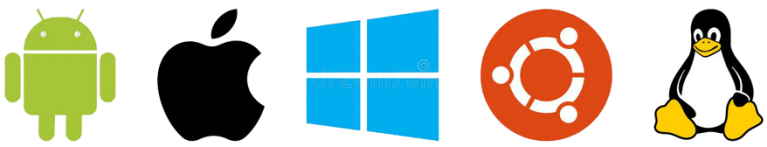
Build Your IT Skill

ETEC IT PROFESSIONAL TRAINING CENTER

Dart Programming



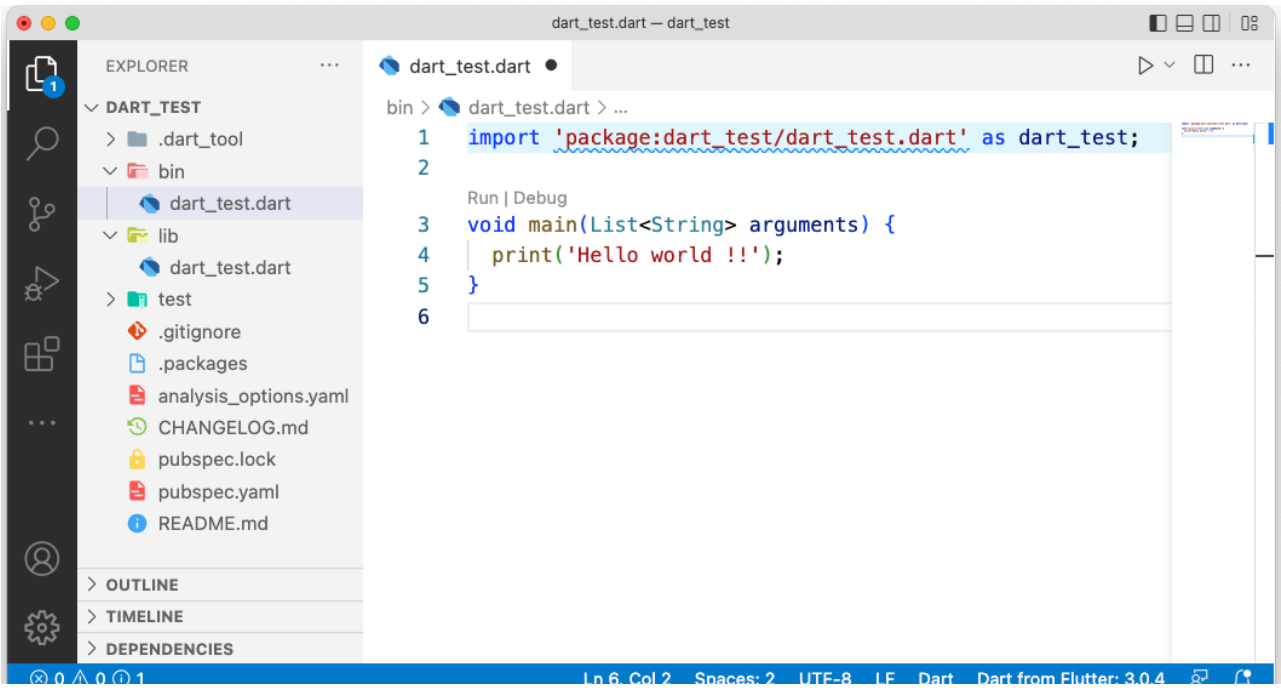
រៀបរៀងដោយ៖ ជា ជាន់ណា

Software Developer

i. Introduction Dart

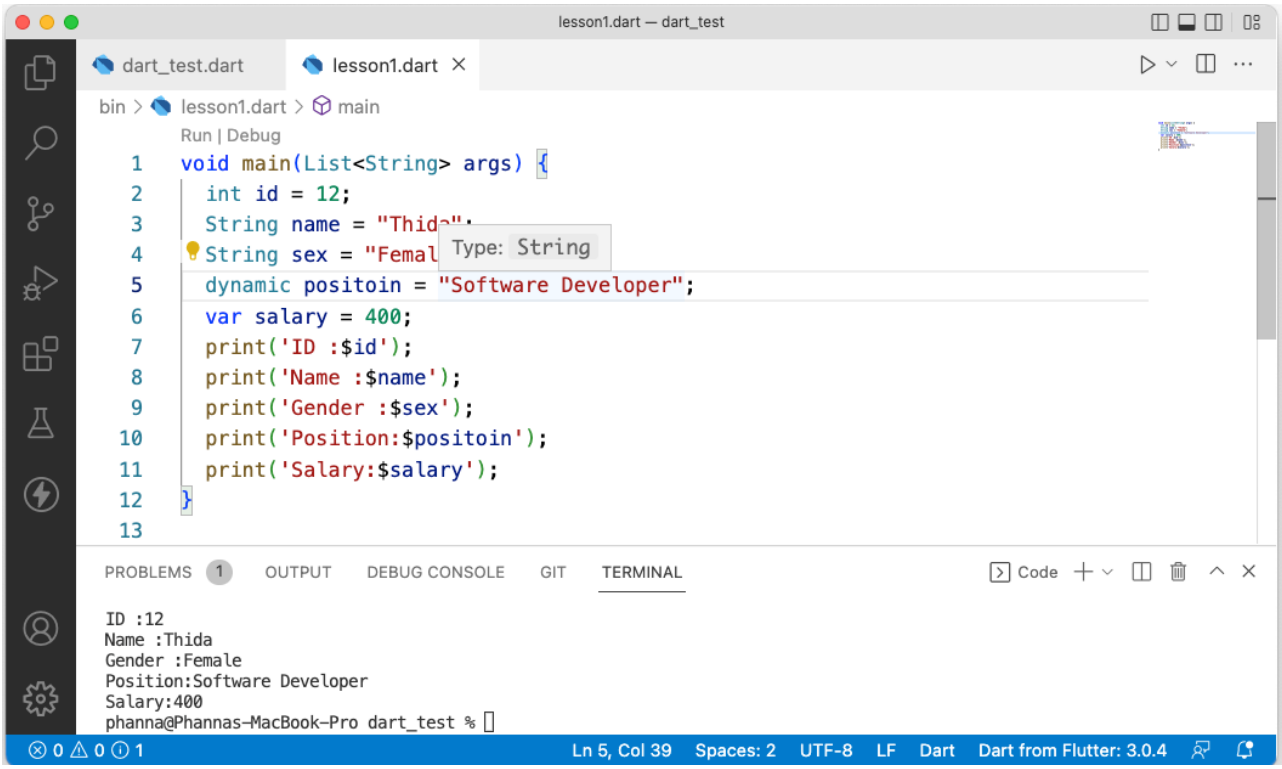
Dart Programming ត្រូវបានបង្កើតដោយ Google នៅឆ្នាំ 2011 ដែលដើមឡើងប្រើសម្រាប់សរសេរ Web,service តែក្រោយមកទៀតគេបាន Develop អោយប្រើប្រាស់ សម្រាប់ build

Web , Mobile ,Desktop ជាមួយ Flutter SDK



ii. Dart I/O

Example 1:



Example2

input.dart

bin > input.dart > main

```dart
import 'dart:io';

Run | Debug
void main(List<String> args) {
  int? id;
  String? name;
  double? score;

  stdout.write('Input Id:');
  id = int.parse(stdin.readLineSync()!);
  stdout.write('Input Name:');
  name = stdin.readLineSync();
  stdout.write('Input score:');
  score = double.parse(stdin.readLineSync()!);
  print('========= Output ==========');
  print('Id=$id');
  print('Name=$name');
  print('Score=$score');
}
```

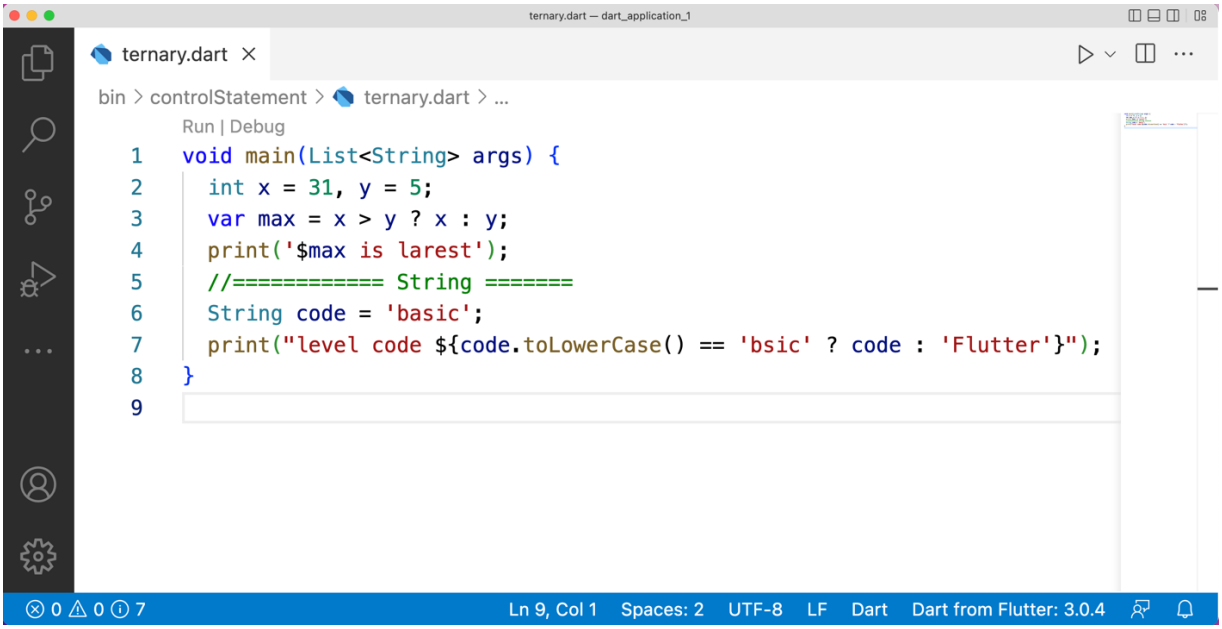
iii. Control Statement

input.dart | ex1.dart

bin > controlStatement > ex1.dart > main

```dart
Run | Debug
void main(List<String> args) {
  int? x = 54;
  int? y = 34;

  // ===== ១ លក្ខខណ្ឌ =======
  if (x > y) {
    print('X = $x is largest than Y');
  }
  // ===== ២ លក្ខខណ្ឌ =======
  if (x > y) {
    print('X =$x biggest');
  } else {
    print('Y=$y biggest');
  }
  // ===== ៣ លក្ខខណ្ឌ =======
  int number = 0;
  if (number < 0) {
    print('$number is Negative number');
  } else if (number > 0) {
    print('$number is  positive number');
  } else {
    print('number is $number');
  }
}
```

- Ternary



- Switch Statement

```dart
Run | Debug
void main(List<String> args) {
  int codeDay;
  stdout.write('Please Choose 1-7:');
  codeDay = int.parse(stdin.readLineSync()!);
  switch (codeDay) {
    case 1:
      print('Monday');
      break;
    case 2:
      print('Tuesday');
      break;
    case 3:
      print('Wednesday');
      break;
    case 4:
      print('Thursday');
      break;
    case 5:
      print('Friday');
      break;
    case 6:
      print('Saturday');
      break;
    case 7:
      print('Sunday');
      break;
  }
}

```

iv. Loop statement

Example1:

bin > loopStatement > loop1.dart > main

```dart
import 'dart:io';

Run | Debug
void main(List<String> args) {
  // ====== forloop =======
  for (int i = 1; i < 10; i = i + 2) {
    stdout.write('$i ');
  }
  print('\n========');
  for (int i = 20; i >= 1; i = i - 2) {
    stdout.write('$i ');
  }

  // ===== while loop ======
  int i = 1;
  while (i <= 10) {
    stdout.write('$i ');
    i = i + 2;
  }

  // ===== do while loop =====
  print(' do while loop ');
  i = 2;
  do {
    print(i);
    i = i + 2;
  } while (i <= 15);
}
```

Example2:

bin > loopStatement > loop_foreach.dart > ...

```dart
Run | Debug
void main() {
  List names = ['Kim long', 'Chanthu', 'lisa', 'dalin'];
  for (var temp in names) {
    print('Name:$temp');
  }
}

```

PROBLEMS 7 OUTPUT DEBUG CONSOLE GIT TERMINAL

```
Name:Kim long
Name:Chanthu
Name:lisa
Name:dalin
phanna@Phannas-MacBook-Pro dart_application_1 %
```

Code

Code

v. Function

- Non- Return Function

Example1:

```
Run | Debug
1   void main(List<String> args) {
2     sum2(32, 44);
3     sum1();
4   }
5
6   //  Non- return funtion not parameter
7   void sum1() {
8     int x = 34, y = 76;
9     print('X+Y=${x + y}');
10  }
11
12  //  Non- return funtion have parameter
13  void sum2(int a, int b) {
14    print('A+B=${a + b}');
15  }
16
```

- Return function

Example2:

```
1   void main(List<String> args) {
2     var x = sum3();
3     print('Sum3=$x');
4     var y = sum4(34, 5);
5     print('Sum4=$y');
6     var sumOfFun = sum3()! + sum4(4, 8);
7     print('Sum of Funtion=$sumOfFun');
8   }
9
10  int? sum3() {
11    sum4(3, 4);
12    int x = 4, y = 6;
13    return x + y;
14  }
15
16  //return funtion have  Paramater [ int , double , bool ,class ,String ]
17  double sum4(int x, int y) {
18    return (x + y).toDouble();
19  }
```

▪ FunctionExpressions

```dart
void main(List<String> args) {
  printText();
  printTextParam('Become to Flutter');
  print(printText2());
  print(printTextParam2('Next level is Flutter'));
}

void printText() => print('Hello Dart Programming');
void printTextParam(String text) => print(text);
String printText2() => 'Start with Dart';
String printTextParam2(String nextLevel) => nextLevel;
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE GIT TERMINAL

```
Hello Dart Programming
Become to Flutter
Start with Dart
Next level is Flutter
phanna@Phannas-MacBook-Pro dart_application_1 %
```

▪ Optional Parameter

```dart
void main(List<String> args) {
  student(43, 'Chhanon');
  print('==============');
  student(3, 'KimSeang', sex: 'Female');
}

void student(int id, String name, {String? sex = 'male'}) {
  print('ID=$id');
  print('Name=$name');
  print('Sex=$sex');
}
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE GIT TERMINAL

```
ID=43
Name=Chhanon
Sex=male
==============
ID=3
Name=KimSeang
Sex=Female
phanna@Phannas-MacBook-Pro dart_application_1 %
```

vi. Collection

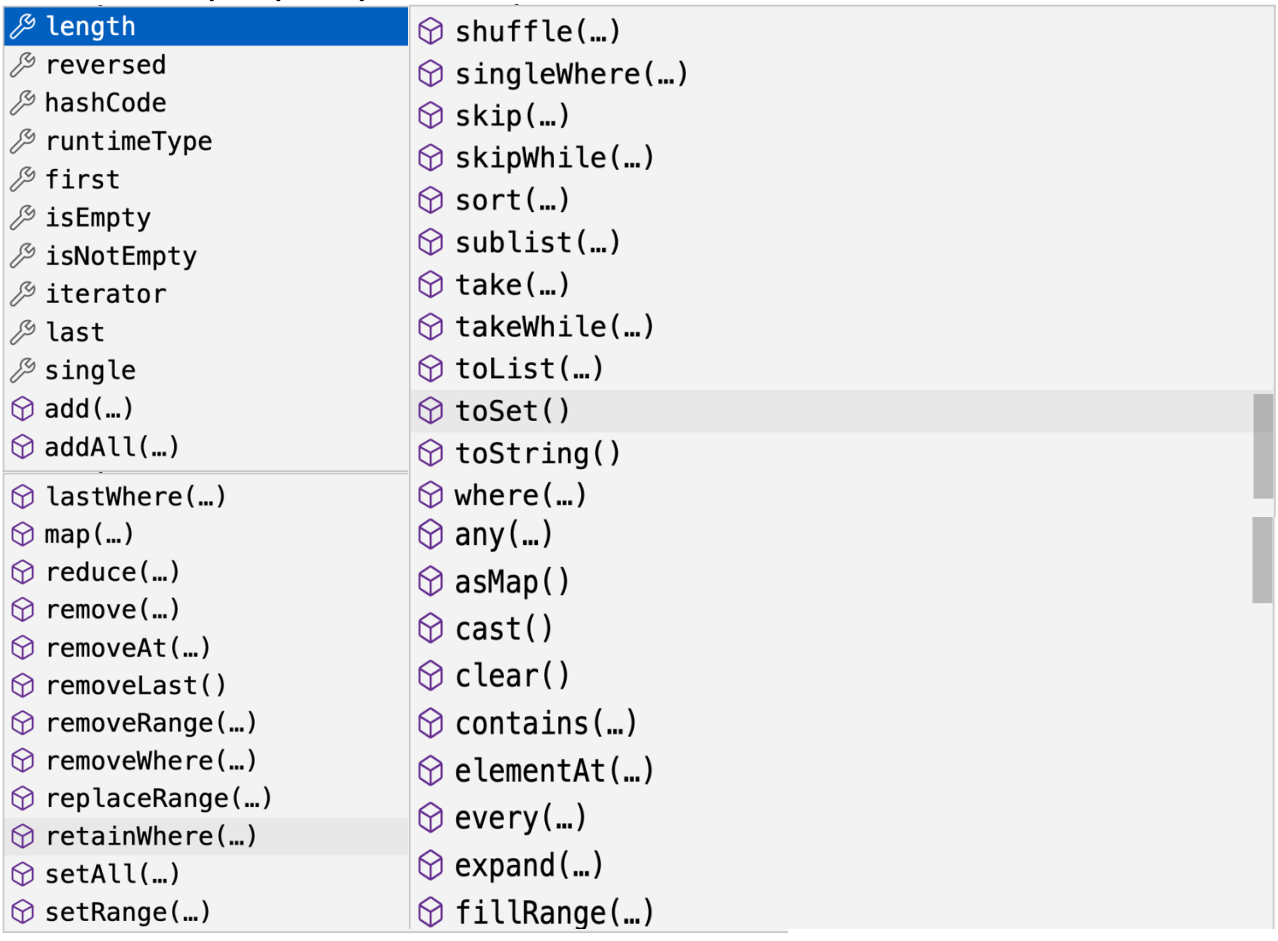
i. List

bin > collection > ex1.dart > ...

Run | Debug

```dart
void main(List<String> args) {
  // syntax 1
  List list = [12, 54, 634, 66, 34, 7, 'Animal'];
  print(list);
  // or
  for (var temp in list) {
    print('Value=$temp');
  }
  // syntax 2
  List<int> listInt = [23, 555];
  print(listInt);

  List<String> name = ['channa', 'nary', 'Davuth'];
  for (var names in name) {
    print('name=$names');
  }
}
```

Collection,property & Method :



Method & property ដែលជួបច្រើនមានដូចជា ៖

- **property**

  length , first ,last ,isEmpty ,isNotEmpty ....

- **Method**

  .add(...) : add ធាតុចូលម្តងមួយ

  .addAll(...) : add ធាតុចូលម្តងទាំងអស់

  .firstWhere(...) : filter រកធាតុដែលដូច និង យកដំបូងគេ

  .map(...) : បំលែងធាតុទៅជា map ដែលប្រើ Key and Value

  .remove(...) : លុបធាតុណាមួយចេញពី list

  .sort(...) : តម្រៀបធាតុ

  .toString(...) : បំលែង data ទៅជា string

  .where(...) : រកធាតុណាមួយ

  .every(...) : យកធាតុទាំងអស់ដែលមានដូច

  .indexWhere(....) : ចាប់យក index ធាតុទាំងអស់ដែលមានដូច

  .contains(...) : យកធាតុទាំងអស់ដែលមានតួអក្សរដូច

  .clear(...) : លុបធាតុទាំងអស់ចេញពី list

  .toList(...) : បំលែងធាតុទៅជា list

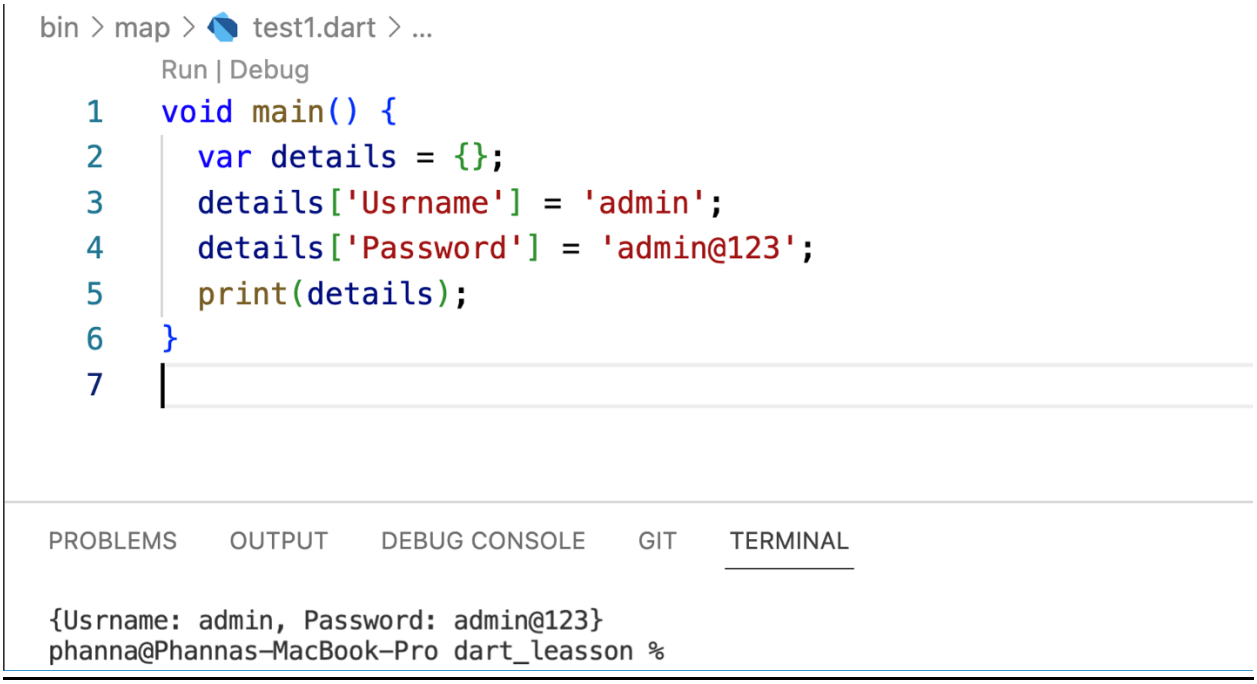
  ,....

ii. Map

Example1:

```dart
void main(List<String> args) {
  Map map = {'id': 123, 'name': 'Thida', 'score': 500};

  print(map);

  Map<String, dynamic> map1 = {'id': 123, 'name': 'Thida', 'score': 500};
  print(map1);

  List<Map<String, dynamic>> list = [
    {'id': 123, 'name': 'Thida', 'score': 500},
    {'id': 143, 'name': 'Vuthy', 'score': 400}
  ];
  print(list);
  for (var temp in list) {
    temp.forEach((key, value) {
      print('$key : $value');
    });
  }
}

```

Example2



Example3

Pratise

ចូរយក Example3 ទៅធ្វើជា Menu ដូចខាងក្រោម ៖

========= Menu =======

1. Add
2. Display
3. Search
4. Update
5. Remove
6. Sort

iii. Collection With Class

Example1:

bin > map > student_model.dart > Student

```dart
class Student {
  late int id;
  late String name;
  late String gender;
  late double score;
  Student(
      {required this.id,
      required this.name,
      required this.gender,
      required this.score});
  void Output() {
    print('$id\t$name\t$gender\t$score');
    print('--------------------------------');
  }
}

Run | Debug
void main(List<String> args) {
  List<Student> stuList = [
    Student(id: 1001, name: 'Angelika', gender: 'F', score: 70.9),
    Student(id: 1002, name: 'ARita', gender: 'M', score: 80)
  ];
  for (var temp in stuList) {
    temp.Output();
  }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE GIT TERMINAL

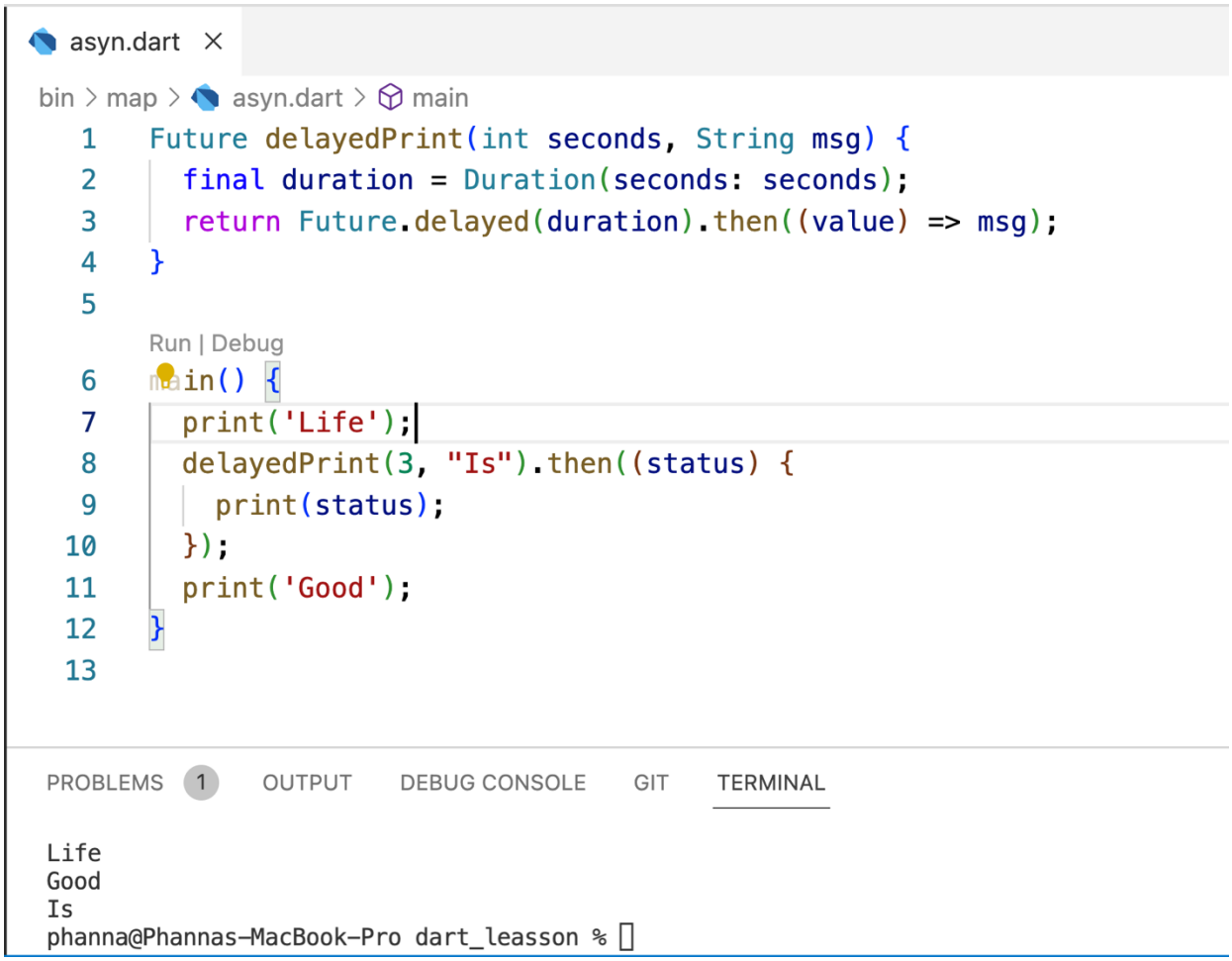
```
1001    Angelika        F       70.9
--------------------------------
1002    ARita   M       80.0
--------------------------------
phanna@Phannas-MacBook-Pro dart_leasson %
```

vii. Asynchronous

Asyncronous គឺជា Function មួយដែលធ្វើអោយ ដំណើរការ Code មិនធ្វើការមុន និង ក្រោយគ្នា ពោលគឺបញ្ចប់នៅ Funtion ដែលមាន asyn & await រួចហើយទើប បន្តទៅមុខទៀត។

ជាទូទៅ function នេះគេប្រើច្រើនជាមួយ Local database, cloud data, api service ,...

- Not use asyn & await



- use asyn & await

Example1:

asyn.dart

bin > map > asyn.dart > main

```dart
Future delayedPrint(int seconds, String msg) {
  final duration = Duration(seconds: seconds);
  return Future.delayed(duration).then((value) => msg);
}

Run | Debug
main() async {
  print('Life');
  await delayedPrint(3, "Is").then((status) {
    print(status);
  });
  print('Good');
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE GIT TERMINAL

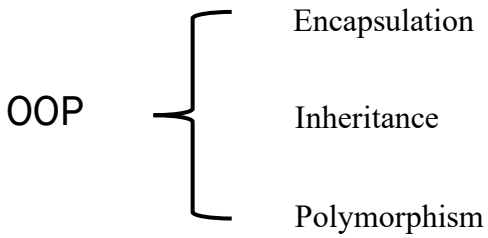
```
Life
Is
Good
phanna@Phannas-MacBook-Pro dart_leasson %
```

Example2:

bin > map > asyn2.dart > ...

```dart
import 'dart:async';

Future<String> firstAsync() async {
  await Future.delayed(const Duration(seconds: 2));
  return "First!";
}

Future<String> secondAsync() async {
  await Future.delayed(const Duration(seconds: 2));
  return "Second!";
}

Future<String> thirdAsync() async {
  await Future.delayed(const Duration(seconds: 2));
  return "Third!";
}

Run | Debug
void main() async {
  var f = await firstAsync();
  print(f);
  var s = await secondAsync();
  print(s);
  var t = await thirdAsync();
  print(t);
  print('done');
}
```

viii. OOP (Object-Oriented Programming)



i. Encapsulation

- Class and Object

bin > OOP > class_object.dart > Person > Person

```dart
class Person {
  int? id;
  String? name;
  String? sex;
  int? age;
  Person(
      {required this.id,
      required this.name,
      required this.sex,
      required this.age});
  Person.newInstances();
  void output() {
    print('$id\t$name\t$sex\t$age');
  }
}

Run | Debug
void main(List<String> args) {
  Person ps1 = Person.newInstances();
  ps1.output();
  print('====================');
  Person ps2 = Person(id: 12, name: 'Vuthy', sex: 'Male', age: 23);
  ps2.output();
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE GIT TERMINAL

```
null    null    null    null
====================
12      Vuthy   Male    23
phanna@Phannas-MacBook-Pro dart_leasson %
```

- Property get() , set()

```dart
class Product {
  int? code;
  String? name;
  int? qty;
  double? price;
  double? total() {
    return price! * qty!;
  }

  void setCode(int code) {
    this.code = code;
  }

  void setName(String name) {
    this.name = name;
  }

  void setPrice(double price) {
    this.price = price;
  }

  void setQty(int qty) {
    this.qty = qty;
  }

  void displayItem() {
    print('$code\t$name\t$qty\t$price\$\t${total()}\$');
  }
}

Run | Debug
void main(List<String> args) {
  Product pro = Product();
  pro.setCode(1001);
  pro.setName('Coca');
  pro.setQty(12);
  pro.setPrice(1.5);
  pro.displayItem();
}
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  GIT  TERMINAL

1001    Coca    12      1.5$    18.0$
phanna@Phannas-MacBook-Pro dart_leasson %
```

ii. Inheritance

- super class

class_object.dart

bin > OOP > class_object.dart > main

```dart
class Person {
  int? id;
  String? name;
  String? sex;
  int? age;
  Person(
      {required this.id,
      required this.name,
      required this.sex,
      required this.age});
  Person.newInstances();
  void output() {
    print('$id\t$name\t$sex\t$age');
  }
}
```

- sub class

class_object.dart | sudent_model.dart

bin > OOP > sudent_model.dart > ...

```dart
import 'class_object.dart';

class Student extends Person {
  double? score;
  Student({this.score, int? id, String? name, String? sex, int? age})
      : super(id: id, name: name, sex: sex, age: age);
  Student.newInstances() : super.newInstances();
  @override
  void output() {
    super.output();
    print('Score:$score');
  }
}

Run | Debug
void main(List<String> args) {
  Student stu =
      Student(id: 1002, name: 'Monica', age: 20, sex: 'Female', score: 80.6);
  stu.output();
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE GIT TERMINAL

```
1002    Monica  Female  20
Score:80.6
phanna@Phannas-MacBook-Pro dart_leasson %
```

Interfaces & Abstract class

❖ Interfaces

- ➢ Not have key word **interfaces**
- ➢ Use key **implements** to implement from super class
- ➢ In super class can have data member & function member

Example:

implement_and_abstract.dart

bin > OOP > implement_and_abstract.dart > Item

```dart
class Product {
  int? code;
  String? name;
  int? qty;
  double? price;
  void addProduct() {}
}

class Item implements Product {
  void updateProduct() {}
  void showProduct() {
    print('$code\t$name\t$qty\t$price');
  }

  @override
  int? code;

  @override
  String? name;

  @override
  double? price;

  @override
  int? qty;

  @override
  void addProduct() {}
}
```

❖ Abstract

- ➢ Use key word **abstract**
- ➢ Use key **extent** to implement from super class
- ➢ In **abstract class** have abstract function , but function not has body

Example:

implement_and_abstract.dart

bin > OOP > implement_and_abstract.dart > Item

```dart
abstract class Product {
  void addProduct();
  void updateProduct();
  void showProduct();
}

class Item extends Product {
  int? code;
  String? name;
  int? qty;
  double? price;

  @override
  void addProduct() {
    // TODO: implement addProduct    TODO: implement addProduct
  }

  @override
  void updateProduct() {
    // TODO: implement updateProduct    TODO: implement updateProduct
  }

  @override
  void showProduct() {
    // TODO: implement showProduct    TODO: implement showProduct
  }
}

```

iii. Polymorphism

poly.dart

bin > OOP > poly.dart > main

```dart
class Car {
  void driving() {
    print("driving car 1");
  }
}

class Honda extends Car {
  //override method overrides generic driving method
  @override
  void driving() {
    print("driving car 2");
    super.driving(); //calls generic driving method
  }
}

Run | Debug
void main() {
  Honda car1 = Honda();
  car1.driving();
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE GIT TERMINAL

```
driving car 2
driving car 1
phanna@Phannas-MacBook-Pro dart_leasson %
```

Learn more :

Dart: https://dart.dev/

Flutter : https://flutter.dev/

Habit can make you learn fast

Thank you..!!