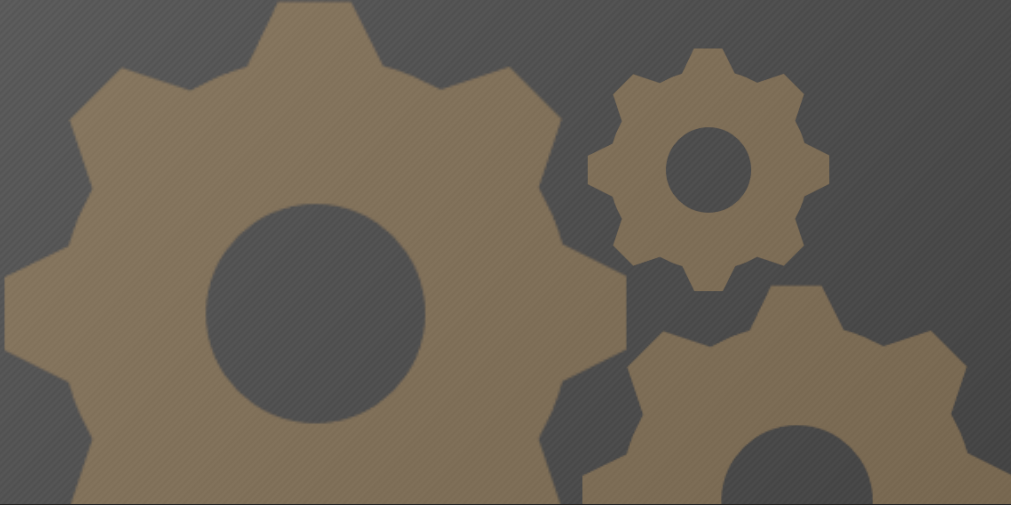




Dart programming language

Application and Game Development for Mobile Device



DART : Part 3

เตรียมความพร้อมสำหรับการสร้าง Mobile Application

<https://dart.dev>

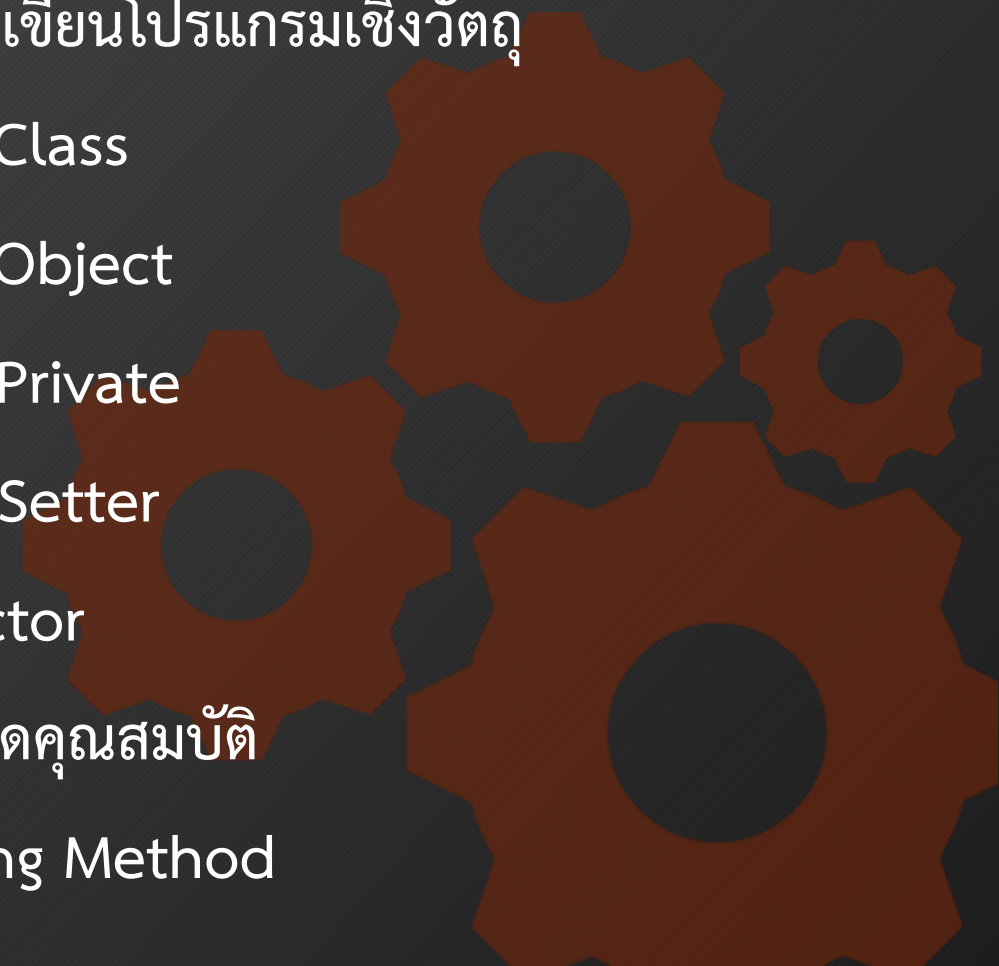


เนื้อหา (Part 3)



- โครงสร้างข้อมูล List
- List Properties & Function
- เข้าถึงสมาชิกใน List ด้วย For
- เข้าถึงสมาชิกใน List ด้วย ForEach
- ฟังก์ชันเพิ่มสมาชิกใน List
- ฟังก์ชันลบสมาชิกใน List
- โครงสร้างข้อมูล Map
- แปลง List เป็น Map
- แปลง Map เป็น List

- ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ
- การสร้าง Class
- การสร้าง Object
- Public / Private
- Getter / Setter
- Constructor
- การสืบทอดคุณสมบัติ
- Overriding Method



โครงสร้างข้อมูล List

- List เป็นโครงสร้างข้อมูล หรือ ตัวแปรที่ใช้สำหรับเก็บข้อมูลได้หลายค่า
- โดยการใช้ชื่ออ้างอิง และใช้หมายเลขตำแหน่ง (index) เริ่มต้นที่ 0
- สามารถกำหนดให้ข้อมูลที่เก็บอยู่ใน List เป็นข้อมูลชนิดเดียวกัน หรือต่างชนิดกันได้
- List vs Array
 - ขนาดของ List ยืดหยุ่นได้ แต่ Array มีขนาดที่แน่นอน
 - ข้อมูลใน Array ต้องมีชนิดข้อมูลที่เหมือนกัน

โครงสร้างข้อมูล List (ต่อ)

- รูปแบบ การสร้าง List

List ชื่อตัวแปร = ['ข้อมูล'];

List <ชนิดข้อมูล> ชื่อตัวแปร = ['ข้อมูล'];

var ชื่อตัวแปร = ['ข้อมูล'];

```
void main() {  
    List name = ['Piyaphol', 'Pattaphon', 'Akekapong'];  
    List <String> color = ['Red', 'Green', 'Orange'];  
    var number = [100, 200, 300];  
  
    print(name[0]); // Output: Piyaphol  
    print(color[1]); // Output: Green  
    print(number[2]); // Output: Orange  
}
```

โครงสร้างข้อมูล List (ต่อ)

- ตัวอย่าง

List Properties & Function

- การเพิ่มสมาชิก

```
List <String> color = ['Red', 'Green', 'Orange'];  
print(color);           // [Red, Green, Orange]  
color.add("Blue");  
print(color);           // [Red, Green, Orange, Blue]
```

- การแสดงจำนวนสมาชิก

- การแสดงสมาชิกตัวแรก และสมาชิกตัวสุดท้าย

```
var countColor = color.length;  
var firstColor = color.first;  
var lastColor = color.last;  
print('จำนวนข้อมูลของ color = $countColor');  
print('ข้อมูลแรกของ color = $firstColor');  
print('ข้อมูลสุดท้ายของ color = $lastColor');
```

```
print('จำนวนข้อมูลของ color = ${color.length}');  
print('ข้อมูลแรกของ color = ${color.first}');  
print('ข้อมูลสุดท้ายของ color = ${color.last}');
```


เข้าถึงสมาชิกใน List ด้วย For

```
List<String> color = ['Red', 'Green', 'Orange', 'Blue'];
```

```
print('color[0] = ${color[0]}');  
print('color[1] = ${color[1]}');  
print('color[2] = ${color[2]}');  
print('color[3] = ${color[3]}');
```

```
color[0] = Red  
color[1] = Green  
color[2] = Orange  
color[3] = Blue
```

```
for(int i=0; i<color.length; i++){  
    print('color[$i] = ${color[i]}');  
}
```

```
color[0] = Red  
color[1] = Green  
color[2] = Orange  
color[3] = Blue
```


เข้าถึงสมาชิกใน List ด้วย ForEach

- เป็นการใช้งานตัวแปรเข้ามารับข้อมูลแล้วนำไปใช้งาน
- และจะเริ่มตั้งแต่ตำแหน่งแรก แล้ววนเข้าไปจนถึงตำแหน่งสุดท้ายของ List

```
var index=0;  
for(var item in number){  
    print("Member[$index] = $item");  
    index++;  
}
```

```
Member[0] = 83  
Member[1] = 69  
Member[2] = 63  
Member[3] = 17  
Member[4] = 79  
Member[5] = 19  
Member[6] = 49  
Member[7] = 24  
Member[8] = 48  
Member[9] = 64
```

ฟังก์ชันเพิ่มสมาชิกใน List

- `add(value)` // เพิ่มสมาชิกต่อท้าย
- `addAll(list)` // เพิ่มสมาชิกจากตัวแปร List
- `insert(index, value)` // เพิ่มสมาชิกโดยระบุตำแหน่งเพื่อแทรกข้อมูล
- `insertAll(index, list)` // เพิ่มสมาชิกโดยระบุตำแหน่งเพื่อแทรกข้อมูลจากตัวแปร List

ฟังก์ชันเพิ่มสมาชิกใน List (ต่อ)

- ตัวอย่าง

```
List <String> color = ['Red', 'Green', 'Blue'];  
    print(color);  
// เพิ่ม 1 สมาชิกต่อท้าย  
color.add('Orange');  
    print(color);  
// เพิ่มหลายสมาชิกจาก List ต่อท้าย  
List <String> color2 = ['white', 'black', 'Yellow'];  
color.addAll(color2);  
    print(color);  
// แทรก 1 สมาชิกในตำแหน่งที่ระบุ  
color.insert(0, 'Brown');  
    print(color);  
// แทรกหลายสมาชิกจาก List เริ่มจากตำแหน่งที่ระบุ  
color.insertAll(2, color2);  
    print(color);
```

ฟังก์ชันลบสมาชิกใน List

- `remove(value)` // ลบสมาชิก
- `removeRange(start, end+1)` // ลบสมาชิกแบบกำหนดช่วงของตำแหน่ง
- `removeAt(index)` // ลบสมาชิกในตำแหน่งที่ต้องการ
- `removeWhere(condition)` // ลบสมาชิกแบบกำหนดเงื่อนไข

เช่น `removeWhere((item) => item % 2 == 0);`

ฟังก์ชันลบสมาชิกใน List

- ตัวอย่าง

```
// ลบสมาชิก
color.remove('white');
print(color);
// ลบสมาชิกแบบกำหนดช่วงของตำแหน่ง
color.removeRange(0, 1);
print(color);
// ลบสมาชิกในตำแหน่งที่ต้องการ
color.removeAt(3);
print(color);
// ลบสมาชิกแบบกำหนดเงื่อนไข
color.removeWhere((item) => item == 'white');
print(color);
```

โครงสร้างข้อมูล Map

- เป็นโครงสร้างข้อมูลที่เก็บข้อมูล Key กับ Value มีลักษณะคล้ายกับ List
- สามารถกำหนดชื่อของ Index ได้
- รูปแบบการสร้าง

Map <key, value>

Map <dynamic, dynamic>

- รูปแบบการเข้าถึง

ชื่อMap[ชื่อ Key];

โครงสร้างข้อมูล Map (ต่อ)

- ตัวอย่าง

```
void main() {  
    Map <String, String> color = {"Red": "สีแดง", "Green": "สีเขียว", "Blue": "สีน้ำเงิน"};  
    Map <int, String> gender = {0: "ผู้ชาย", 1: "ผู้หญิง"};  
  
    print('Red = ${color['Red']}');  
    print('code 0 = ${gender[0]}');  
}
```


โครงสร้างข้อมูล Map (ต่อ)

- การเพิ่มสมาชิก / แก้ไขข้อมูล

ชื่อ Map[ชื่อ Key] = ข้อมูล

- การลบสมาชิก

ชื่อ Map.remove(ชื่อ Key);

โครงสร้างข้อมูล Map (ต่อ)

- ตัวอย่าง

```
Map <String, String> color = {"Red": "สีแดง", "Green": "สีเขียว", "Blue": "สีน้ำเงิน"};
print(color);
// Output: {Red: สีแดง, Green: สีเขียว, Blue: สีน้ำเงิน}

color["Black"] = "สีดำ";
print(color);
// Output: {Red: สีแดง, Green: สีเขียว, Blue: สีน้ำเงิน, Black: สีดำ}

color.remove("Red");
print(color);
// Output: {Green: สีเขียว, Blue: สีน้ำเงิน, Black: สีดำ}
```

แปลง List เป็น Map

- รูปแบบ

```
Map<int, ชนิดข้อมูลของ List> ชื่อMap = ชื่อ List.asMap();
```

- ตัวอย่าง

```
List<String> color = ['Red', 'Green', 'Blue'];
```

```
Map<int, String> item = color.asMap();
```

แปลง Map เป็น List

- รูปแบบ

```
var ชื่อตัวแปร = ชื่อMap.keys;           // กลุ่มข้อมูล Key  
var ชื่อตัวแปร = ชื่อMap.values;         // กลุ่มข้อมูล Value
```

- ตัวอย่าง

```
Map <String, String> color = {"Red":"สีแดง", "Green":"สีเขียว"};  
var color1 = color.keys;           // ข้อมูล ชื่อสี EN  
var color2 = color.values;         // ข้อมูล ชื่อสี TH
```

แปลง Map เป็น List (ต่อ)

- รูปแบบการแปลงข้อมูล

List ชื่อตัวแปร = ชื่อMap.**keys.toList()**;

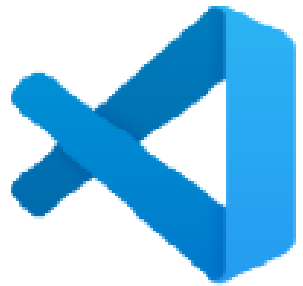
List ชื่อตัวแปร = ชื่อMap.**values.toList()**;

- ตัวอย่าง

```
Map <String, String> color = {"Red":"สีแดง", "Green":"สีเขียว"};
```

```
List color1 = color.keys.toList();    // List ชื่อสี EN ทั้งหมด
```

```
List color2 = color.values.toList();  // List ชื่อสี TH ทั้งหมด
```



Visual Studio Code

ติดตั้งโปรแกรม Visual Studio Code และติดตั้งส่วนเสริมให้สามารถใช้งานได้

Download Install and Setup

Flutter Dev
With
Visual Studio Code



โปรแกรม VS Code

- <https://code.visualstudio.com/download>

ส่วนเสริม

- Dart
- Flutter

Set Environment PATH

- C:\src\flutter2.2.1-stable\bin

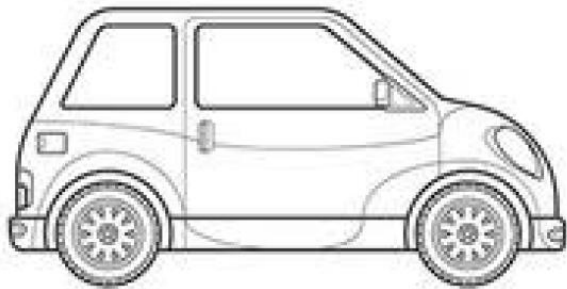
ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ

- class คือ ต้นแบบของวัตถุ เป็นการสร้างโครงสร้างต้นแบบของวัตถุ
- object คือ สิ่งประกอบไปด้วยคุณสมบัติ 2 ประการ(คุณลักษณะ และพฤติกรรม)
 - attribute / data member / fields
คือ สิ่งบ่งบอกลักษณะทั่วไปของวัตถุ
 - behavior / method
คือ พฤติกรรมทั่วไปของวัตถุที่สามารถกระทำได้

ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ (ต่อ)

- class car

Class



Blueprint of a car

Object

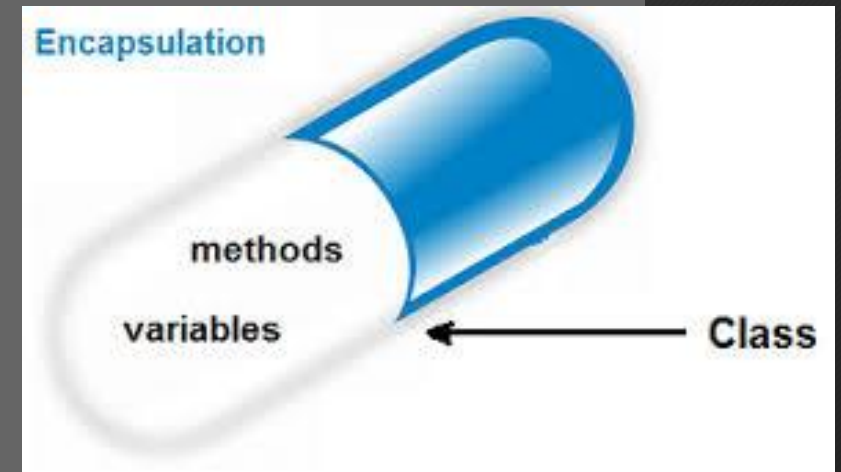


Car

ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ (ต่อ)

กระบวนการเขียนโปรแกรม

- **Encapsulation (การห่อหุ้ม)**
 - เป็นกระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายในไม่ให้ภายนอกสามารถมองเห็นได้
 - ทำให้ภายนอกไม่สามารถทำการเปลี่ยนแปลงข้อมูลที่อาจจะทำให้เกิดความเสียหายแก่ข้อมูลภายในได้
 - สร้างความปลอดภัยให้แก่ข้อมูลได้เนื่องจากข้อมูลจะถูกเข้าถึงได้เฉพาะผู้ที่มีสิทธิ์เท่านั้น

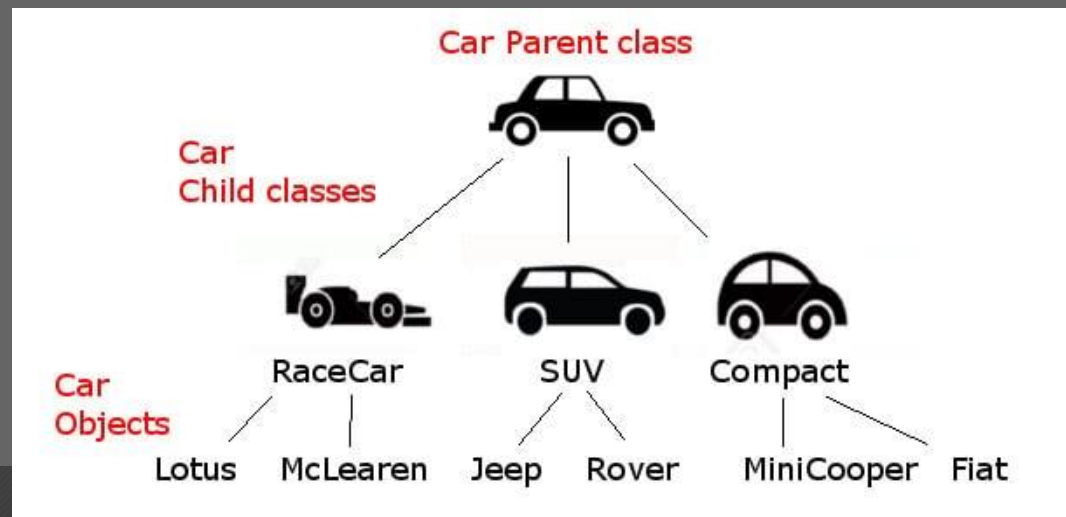


ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ (ต่อ)

กระบวนการเขียนโปรแกรม

- **Inheritance (การสืบทอดคุณสมบัติ)**

- เป็นการสร้างสิ่งใหม่ด้วยการสืบทอดคลาส หรือรับเอา (Inherit) คุณสมบัติบางอย่างจากสิ่งเดิมที่มีอยู่แล้ว (re-use) แล้วสร้างเพิ่มเติมจากสิ่งที่มีอยู่แล้วได้โดยไม่ต้องสร้างขึ้นใหม่ทั้งหมด



ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ (ต่อ)

กระบวนการเขียนโปรแกรม

- **Polymorphism (การพ้องรูป)**
 - เกิดจาก poly(หลากหลาย) + morphology(รูปแบบ)
 - ในการเขียนโปรแกรม คือ
การที่ method ชื่อเดียวกัน สามารถรับ argument ที่แตกต่างกันได้หลายรูปแบบ
 - โดย method นี้ถูกเรียกว่า **overload method**
- ในภาษา Dart ไม่รองรับ
- เพราะ มีการเรียกใช้ **Optional parameter** อยู่แล้วใช้แทนกันได้เลย

ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ (ต่อ)



การสร้าง Class

- ชื่อ class == ชื่อไฟล์
- ภายในคลาส
 - สร้างคุณสมบัติ
 - สร้าง (method)

การสร้าง Class (ต่อ)

- ตัวอย่าง

```
lib > class_employee.dart > Employee > showData
1  class Employee {
2      // public variable
3      String ?_name;
4      String ?_position;
5      double ?_salary;
6
7      Employee(int number){
8          print("New object created. => $number");
9      }
10
11     void endLine(){
12         print("-----");
13     }
14
15     void showData(){
16         print("-----");
17         print("ชื่อพนักงาน : $_name");
18         print("ตำแหน่ง : $_position");
19         print("เงินเดือน : $_salary");
20     }
21 }
```

การสร้าง Object

- เรียกใช้งาน class โดยการ import ไฟล์ class ที่ต้องการใช้งาน

- รูปแบบ

ชนิดข้อมูลคลาส(ชื่อคลาส) object = ชื่อคลาส(); // call constructor

- ตัวอย่าง

```
Employee emp1 = Employee();
```

การสร้าง Object (ต่อ)

- ตัวอย่าง

```
// สร้าง Object
Employee emp1 = Employee(1);
emp1.setName("ปิยพล");
emp1.setPosition("นักวิเคราะห์ข้อมูล");
emp1.setSalary(30000);

Employee emp2 = Employee(2);
emp2.setName("เอกชัย");
emp2.setPosition("นักพัฒนาระบบ");
emp2.setSalary(20000);
```

Public / Private

- Access Modifiers

```
class Product {  
    String name;    // public  
    String _name;  // private  
    void show()=>print("public method");  
    void _show()=>print("private method");  
}
```

Public / Private (ต่อ)

- ตัวอย่าง

```
// public variable  
String ?_name;  
String ?_position;  
double ?_salary;  
  
// private variable  
String ?name;  
String ?position;  
double ?salary;
```

THIS / Super

Keyword

- **this** คือ การเรียกใช้งาน constructor หรือคุณสมบัติอื่นๆ ที่อยู่ในคลาสเดียวกัน
- **super** คือ การเรียกใช้งาน constructor ของคลาสแม่ให้ทำงาน
(keyword **super** นี้ในการเรียกใช้งาน constructor ของคลาสแม่ จะต้องเรียกใช้งานที่บรรทัดแรกสุดของ constructor นั้นเท่านั้น)

Getter / Setter

- Getter คือ การเรียกใช้งานตัวแปรภายในคลาส
- Setter คือ การกำหนดค่าให้กับตัวแปร
- โดยทั้ง Getter และ Setter จะทำงานผ่าน method ภายในเท่านั้น และสามารถใช้งานด้วยการ import

Getter / Setter (ต่อ)

```
void main() {  
    print("main file : test employee");  
    // สร้าง Object  
    Employee emp1 = Employee(1);  
    emp1.setName("ปีย์พล");  
    emp1.setPosition("นักวิเคราะห์ข้อมูล");  
    emp1.setSalary(30000);  
  
    Employee emp2 = Employee(2);  
    emp2.setName("เอกชัย");  
    emp2.setPosition("นักพัฒนาระบบ");  
    emp2.setSalary(20000);  
  
    emp1.endLine();  
  
    print(emp1.getName());  
    print(emp1.getPosition());  
    print(emp1.getSalary());  
    emp1.endLine();  
  
    print(emp2.getName());  
    print(emp2.getPosition());  
    print(emp2.getSalary());  
    emp2.endLine();  
}
```

```
void setName(String name){  
    _name = name;  
}  
void setPosition(String position){  
    _position = position;  
}  
void setSalary(double salary){  
    _salary = salary;  
}  
  
String getName(){  
    return _name.toString();  
}
```

```
main file : test employee  
New object created. => 1  
New object created. => 2  
-----  
ปีย์พล  
นักวิเคราะห์ข้อมูล  
30000.0  
-----  
เอกชัย  
นักพัฒนาระบบ  
20000.0  
-----
```

```
tion(){  
tion.toString();  
  
ary(){  
ry;
```

Constructor

- การสร้าง method ที่ชื่อเหมือนกับชื่อ class
- ใช้สำหรับกำหนดค่าเริ่มต้นให้กับ Object

```
class Product{  
    Product(){  
        print("default constructor");  
    }  
}
```

Constructor (ต่อ)

```
Employee(int number, String name, String position, double salary){  
    print("New object created. => $number");  
    _name = name;  
    _position = position;  
    _salary = salary;  
}
```

```
Employee(int number, this._name, this._position, this._salary){  
    print("New object created. => $number");  
    // _name = name;  
    // _position = position;  
    // _salary = salary;  
}
```

```
// สร้าง Object  
Employee emp1 = Employee(1, "ปิยพล", "นักวิเคราะห์ข้อมูล", 30000);  
Employee emp2 = Employee(2, "เอกชัย", "นักพัฒนาระบบ", 20000);
```

```
emp1.showData();  
emp2.showData();
```

```
New object created. => 1  
New object created. => 2  
-----  
ชื่อพนักงาน : ปิยพล  
ตำแหน่ง : นักวิเคราะห์ข้อมูล  
เงินเดือน : 30000.0  
-----  
ชื่อพนักงาน : เอกชัย  
ตำแหน่ง : นักพัฒนาระบบ  
เงินเดือน : 20000.0
```

การสืบทอดคุณสมบัติ

- สืบทอดคุณสมบัติ และเพิ่มเติมคุณสมบัติ
- รูปแบบ

```
class ชื่อคลาสลูก extends ชื่อคลาสแม่ {  
    // create constructor to call super(...)  
    // ชื่อคลาสลูก (constructor ของคลาสแม่) : super(argument);  
    // กำหนดคุณสมบัติเพิ่มเติม  
    // สร้าง method  
}
```

การสืบทอดคุณสมบัติ (ต่อ)

```
import 'package:dart_oop/class_employee.dart';  
class Sa extends Employee{  
  Sa(int number, String? name, String? position, double? salary) : super(number, name, position, salary);  
}
```

```
import 'package:dart_oop/class_employee.dart';  
class Dev extends Employee {  
  Dev(int number, String? name, String? position, double? salary) : super(number, name, position, salary);  
}
```

```
// สร้าง Object จากการสืบทอด  
Sa systemAnalyst = Sa(1, "ปิยพล", "นักวิเคราะห์ข้อมูล", 30000);  
systemAnalyst.showData();
```

```
Dev developer = Dev(2, "เอกชัย", "นักพัฒนาระบบ", 20000);  
developer.showData();
```

New object created. => 1

ชื่อพนักงาน : ปิยพล
ตำแหน่ง : นักวิเคราะห์ข้อมูล
เงินเดือน : 30000.0

New object created. => 2

ชื่อพนักงาน : เอกชัย
ตำแหน่ง : นักพัฒนาระบบ
เงินเดือน : 20000.0

Overriding Method

- Overloading method คือ method ที่มีชื่อเหมือนกัน และอยู่ภายในคลาสเดียวกัน สิ่งที่แตกต่างกันของ method ที่เป็น overload method คือ parameter เป็นผลมาจากคุณสมบัติ polymorphism
(ในภาษา Dart ใช้รูปแบบของ Optional Parameter แทน)
- Overriding method คือ method ของคลาสลูก (subclass) ที่มีชื่อเหมือนกับ method ของคลาสแม่ (superclass) เป็นผลมาจากคุณสมบัติ Inheritance

Overriding Method (ต่อ)

- เพิ่ม @override
- กำหนดชื่อ method ชื่อเดียวกันกับ method ของคลาสแม่
- ปรับแก้คุณสมบัติของ override method

```
@override  
void showData(){  
    print("ชื่อนักพัฒนาระบบ: "+super.getName());  
    print("เงินเดือน: "+super.getSalary().toString());  
}
```

```
New object created. => 1  
-----  
ชื่อพนักงาน : ปิยพล  
ตำแหน่ง : นักวิเคราะห์ข้อมูล  
เงินเดือน : 30000.0  
New object created. => 2  
ชื่อนักพัฒนาระบบ: เอกชัย  
เงินเดือน: 20000.0
```




Part 1 ✓

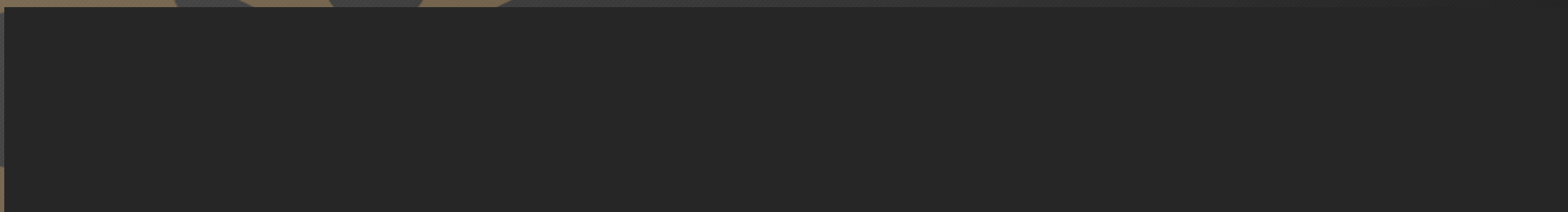
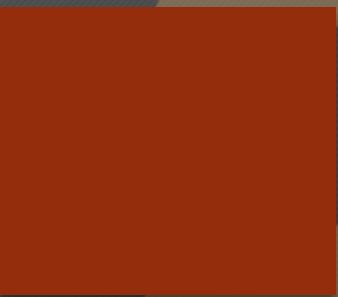
- Dart เบื้องต้น
- การแสดงผลข้อมูล
- Comment
- ตัวแปรและชนิดข้อมูล
- Dynamic Type
- Constant & Final
- กฎการตั้งชื่อตัวแปร
- จัดการอักขระและข้อความด้วย String
- ตัวดำเนินการทางคณิตศาสตร์
- ตัวดำเนินการเปรียบเทียบ
- ตัวดำเนินการเพิ่มและลดค่า
- Compound Assignment
- If Statement
- If..Else
- If แบบหลายเงื่อนไข

Part 2 ✓

- ตัวดำเนินการทางตรรกศาสตร์
- Ternary Operator
- Switch..Case
- While Loop
- For Loop
- Do..While
- Break และ Continue
- การใช้งาน Loop แต่ละแบบ
- การสร้างฟังก์ชัน
- ฟังก์ชันแบบ Return ค่า
- ฟังก์ชันแบบส่ง และ Return ค่า
- Arrow Function
- Optional Parameter
- Named Parameter
- First-Class Function

Part 3 ✓

- โครงสร้างข้อมูล List
- List Properties & Function
- เข้าถึงสมาชิกใน List ด้วย For
- เข้าถึงสมาชิกใน List ด้วย ForEach
- ฟังก์ชันเพิ่มสมาชิกใน List
- ฟังก์ชันลบสมาชิกใน List
- โครงสร้างข้อมูล Map
- แปลง List เป็น Map
- แปลง Map เป็น List
- ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ
- การสร้าง Class
- การสร้าง Object
- Public / Private
- Getter / Setter
- Constructor
- การสืบทอดคุณสมบัติ
- Overriding Method



แบบฝึกหัด



