Chapter 2| Part2

การเขียนแอพพลิเคชันเบื้องต้นด้วยภาษา

Dart

By Aj.Lattagarn Kuikaew

Outline

ประวัติ

โปรแกรมแรก

โครงสร้าง

คลาส

คลาส MaterialApp

แนวความคิดภาษาเชิงวัตถุ

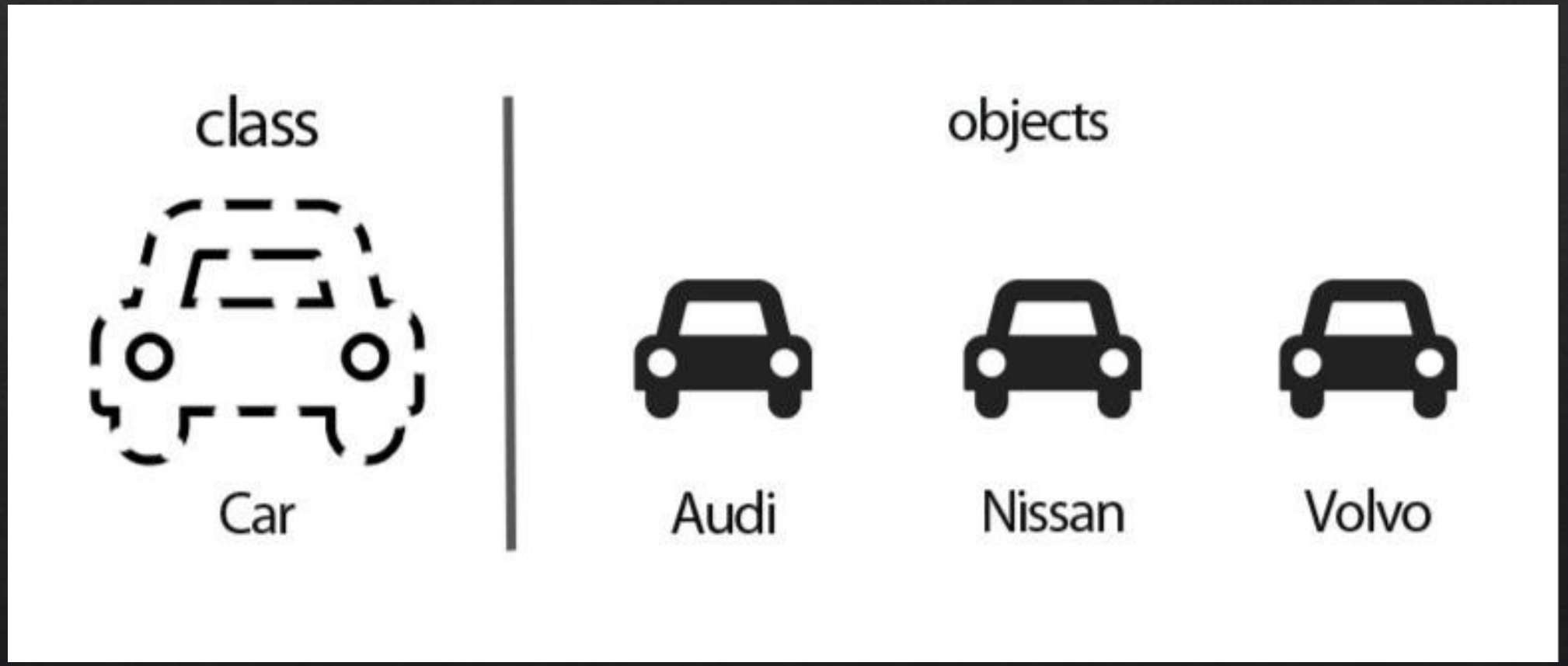
คลาส (class) คือต้นแบบของวัตถุ การจะสร้างวัตถุขึ้นมาอย่างหนึ่งจะต้องสร้างคลาสขึ้นมาเป็นโครงสร้างต้นแบบสำหรับวัตถุก่อนเสมอ

วัตถุ หรือออบเจ็ค (object) คือสิ่งที่ประกอบไปด้วยคุณสมบัติ 2 ประการ คือ คุณลักษณะและพฤติกรรม

คุณลักษณะ (attribute หรือ data member) คือสิ่งที่บ่งบอกลักษณะทั่วไปของวัตถุ

พฤติกรรม (behavior หรือ method) คือพฤติกรรมทั่วไปของวัตถุที่สามารถกระทำได้

แนวความคิดภาษาเชิงวัตถุ

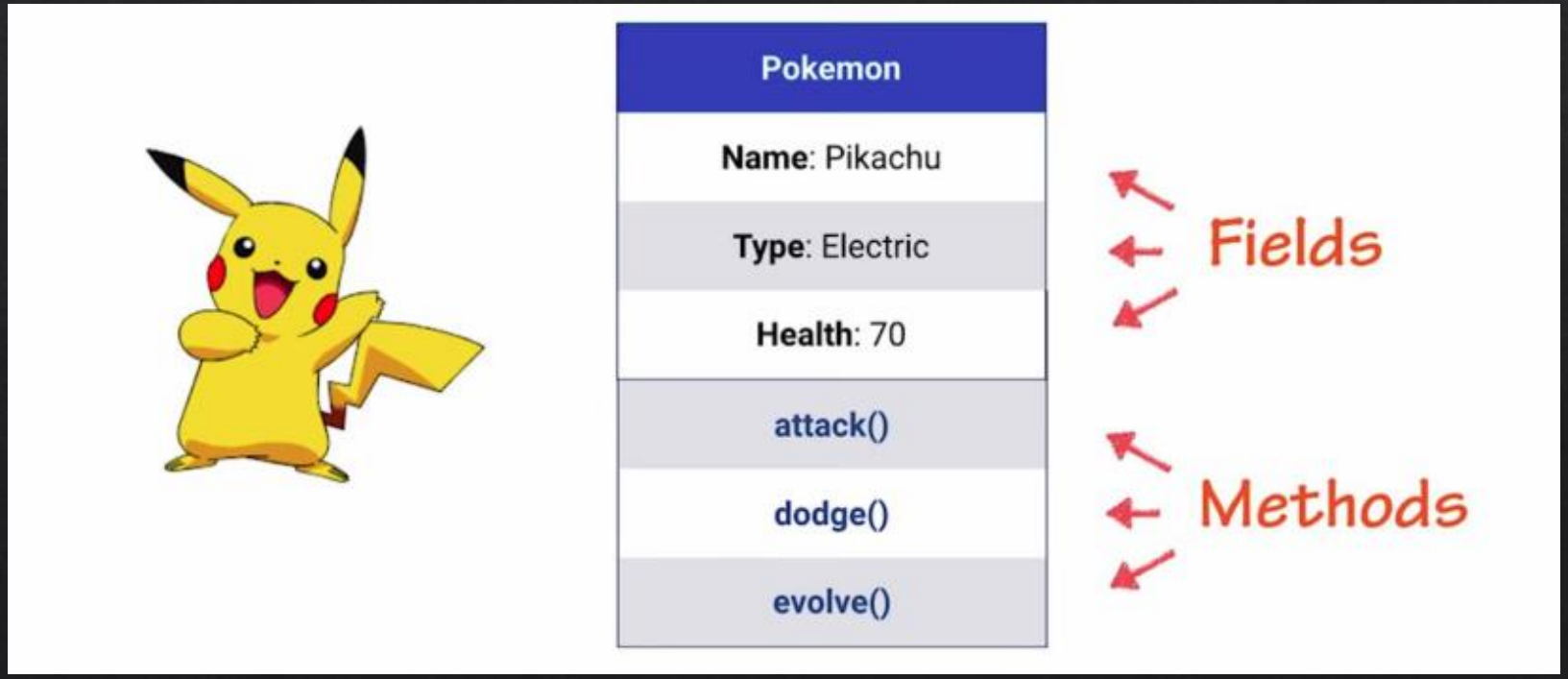


คลาส

```
class <ชื่อของคลาส> {
    [ fields ]
    [ getters/setters ]
    [ constructors ]
    [ functions ]
}
```

Fields

ฟิลด์หรือเขตข้อมูล เป็นตัวแปรใดๆ ที่ประกาศภายในคลาส ซึ่งใช้สำหรับการแสดง ข้อมูลที่เกี่ยวข้องกับวัตถุ



Getters/Setters

เป็นการอนุญาตให้โปรแกรม สามารถเริ่มต้นการอ่านและเขียนค่าข้อมูลให้กับตัวแปรในวัตถุ **โดยการเข้าถึงข้อมูลในฟิลด์** ซึ่งเรียกว่า **อินสแตนซ์ [instance Variable]**

Setters การกำหนดค่าให้ Object

```
class Product{
    String _name;
    void setName(String name){
        this._name=name;
    }
}
```

Getters การกำหนดค่าให้ Object

```
class Product{
    String _name;
    String getName() => _name;
}
```

Constructors

คือการสร้าง**เมธอดหรือฟังก์ชัน**ที่ทำหน้าที่ในการกำหนดการทำงานต่างๆ ให้กับวัตถุใหม่ของคลาสที่สร้างขึ้น โดยชื่อของเมธอดจะเหมือนกับชื่อ Class ใช้สำหรับกำหนดค่าเริ่มต้นให้กับ Object

```
class Product{
    Product(){
        print(“default constructor”)
    }
}
```

```
class Product{
    String _name; String _price;
    Product(String name,String price){
        this._name =name;
        this._price=price;
    }
}
```

Constructors

รู้ลำดับการกำหนดค่าให้วัตถุสามารถลดรูปคำสั่งได้

```
class Product {
    String _name; String _price;
    Product(this._name,this._price);
 }
```

Functions

หมายถึงฟังก์ชันที่ใช้แสดงถึงการกระทำของวัตถุที่สามารถทำได้ ในภาษา Dart อาจเรียกว่า**เมธอด**

ชุดคำสั่งที่นำมาเขียนรวมกันเป็นกลุ่ม เพื่อให้เรียกใช้งานตามวัตถุประสงค์ที่ต้องการและลดความซ้ำซ้อนของคำสั่งที่ใช้งานบ่อยๆ ฟังก์ชันสามารถนำไปใช้งานได้ทุกที่และแก้ไขได้ในภายหลัง ทำให้โค้ดในโปรแกรมมีระเบียบและใช้งานได้สะดวกมากยิ่งขึ้น

โปรแกรมย่อยที่นำเข้ามาเป็นส่วนหนึ่งของโปรแกรมหลัก เพื่อให้สามารถเรียกใช้งานได้โดยไม่จำเป็นต้องเขียนโค้ดคำสั่งใหม่ทั้งหมด

รูปแบบของ Functions

1. ฟังก์ชันที่ไม่มีการรับและส่งค่า

```
void ชื่อฟังก์ชัน(){
    // คำส่งต่างๆ
}
```

การเรียกใช้งานฟังก์ชัน

```
    ชื่อฟังก์ชัน ();
```

รูปแบบของ Functions

2. ฟังก์ชันที่มีการรับค่าเข้ามาทำงาน

```
void ชื่อฟังก์ชัน(parameter1,parameter2,.....){
    // คำส่งต่างๆ
}
```

การเรียกใช้งานฟังก์ชัน

```
    ชื่อฟังก์ชัน (argument1,argument2,.....);
```

อาร์กิวเมนต์ คือ ตัวแปรหรือค่าที่ต้องการส่งมาให้กับฟังก์ชัน (ตัวแปรส่ง)
พารามิเตอร์ คือ ตัวแปรที่ฟังก์ชันสร้างไว้สำหรับรับค่าที่จะส่งเข้ามาให้กับฟังก์ชัน (ตัวแปรรับ)

รูปแบบของ Functions

3. ฟังก์ชันที่มีการส่งค่าออกมา

```
type ชื่อฟังก์ชัน(){
    return ค่าที่จะสั่งออกไป
}
```

รูปแบบของ Functions

4. ฟังก์ชันที่มีการรับค่าเข้ามาและส่งค่าออกไป

```
type ชื่อฟังก์ชัน(parameter1,parameter2,.....){
    return ค่าที่จะสั่งออกไป
}
```

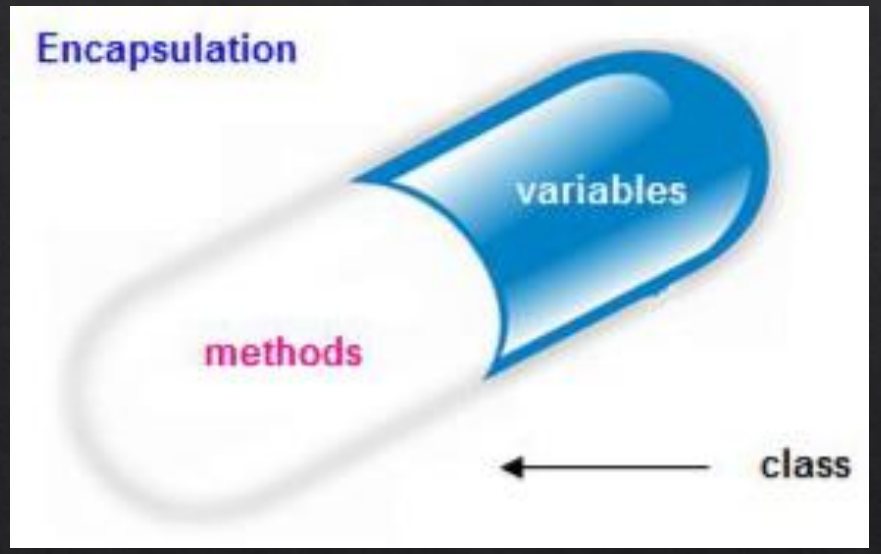
```
class Car {
      //----ฟิลด์
      String engine = “DT-800”;
      //----ฟังก์ชัน
      void disp() {
            print(engine);
      }
}
void main() {
      //เรียกใช้ฟังก์ชันในคลาส Car
      var myCar = new Car();
      //เรียกใช้งานเมธอด/ฟังก์ชัน disp()
      myCar.engine = “TT685”;
      myCar.isp();
}
```

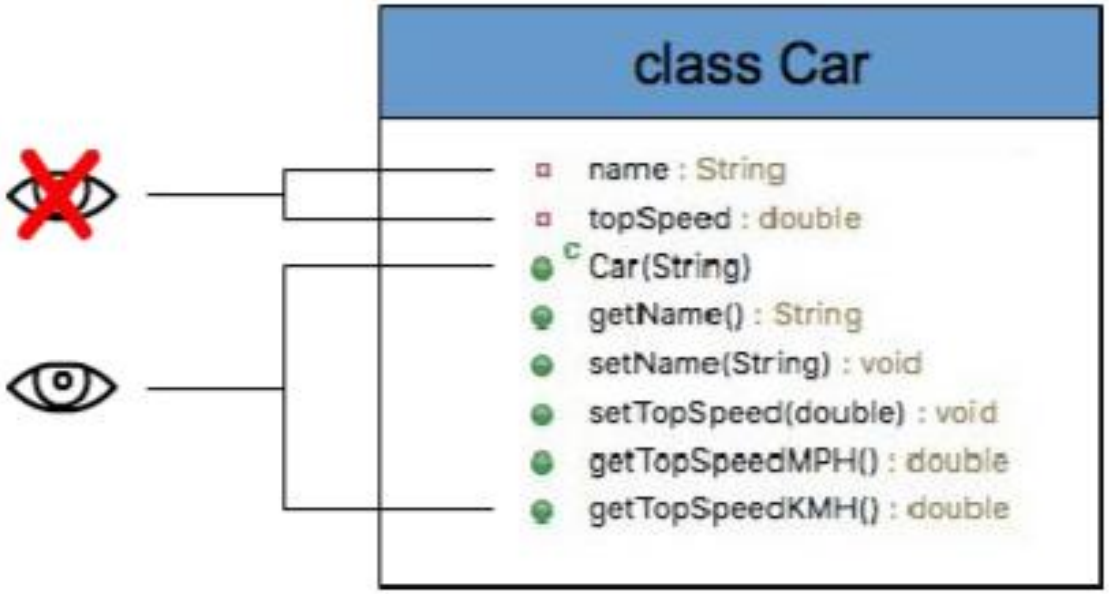
การห่อหุ้ม (Encapsulation)

- เป็นกระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายในไม่ให้ภายนอกสามารถมองเห็นได้

- ทำให้ภายนอกไม่สามารถทำการเปลี่ยนแปลงแก้ไขข้อมูลภายในได้ ซึ่งเป็นผลทำให้เกิดความเสียหายแก่ข้อมูล

- ข้อดีของการห่อหุ้มคือสามารถสร้างความปลอดภัยให้แก่ข้อมูลได้ เนื่องจากข้อมูลจะถูกเข้าถึงจากผู้มีสิทธิ์เท่านั้น

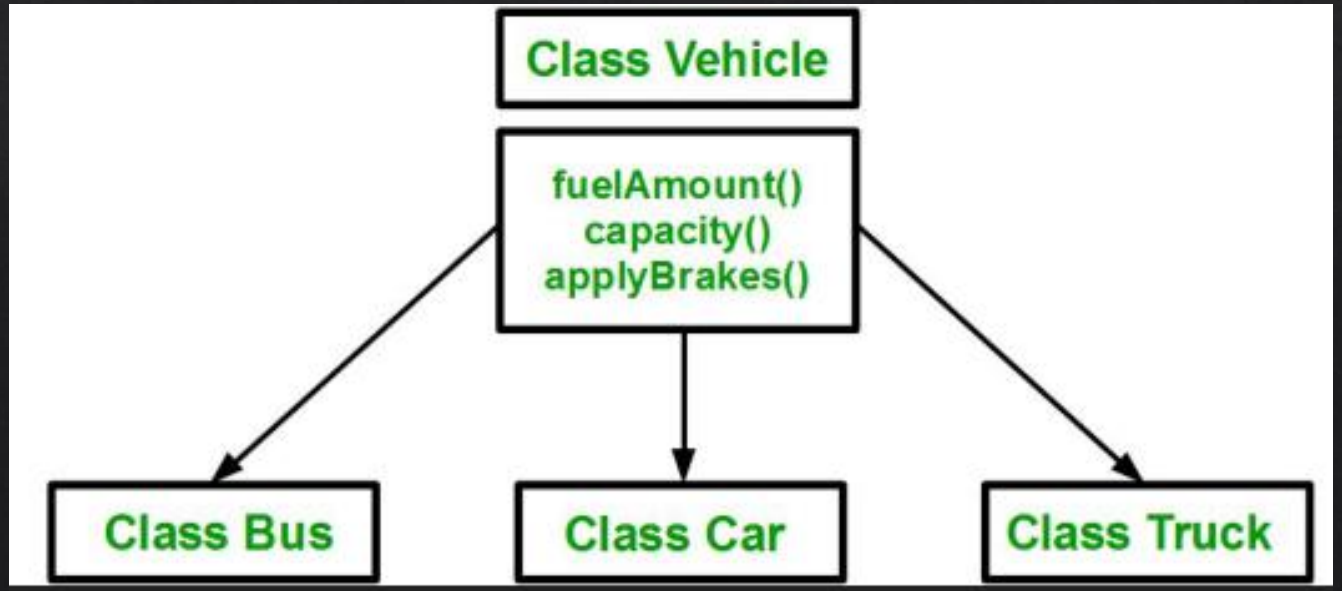



class Car
name : String
topSpeed : double
Car(String)
getName() : String
setName(String) : void
setTopSpeed(double) : void
getTopSpeedMPH() : double
getTopSpeedKMH() : double

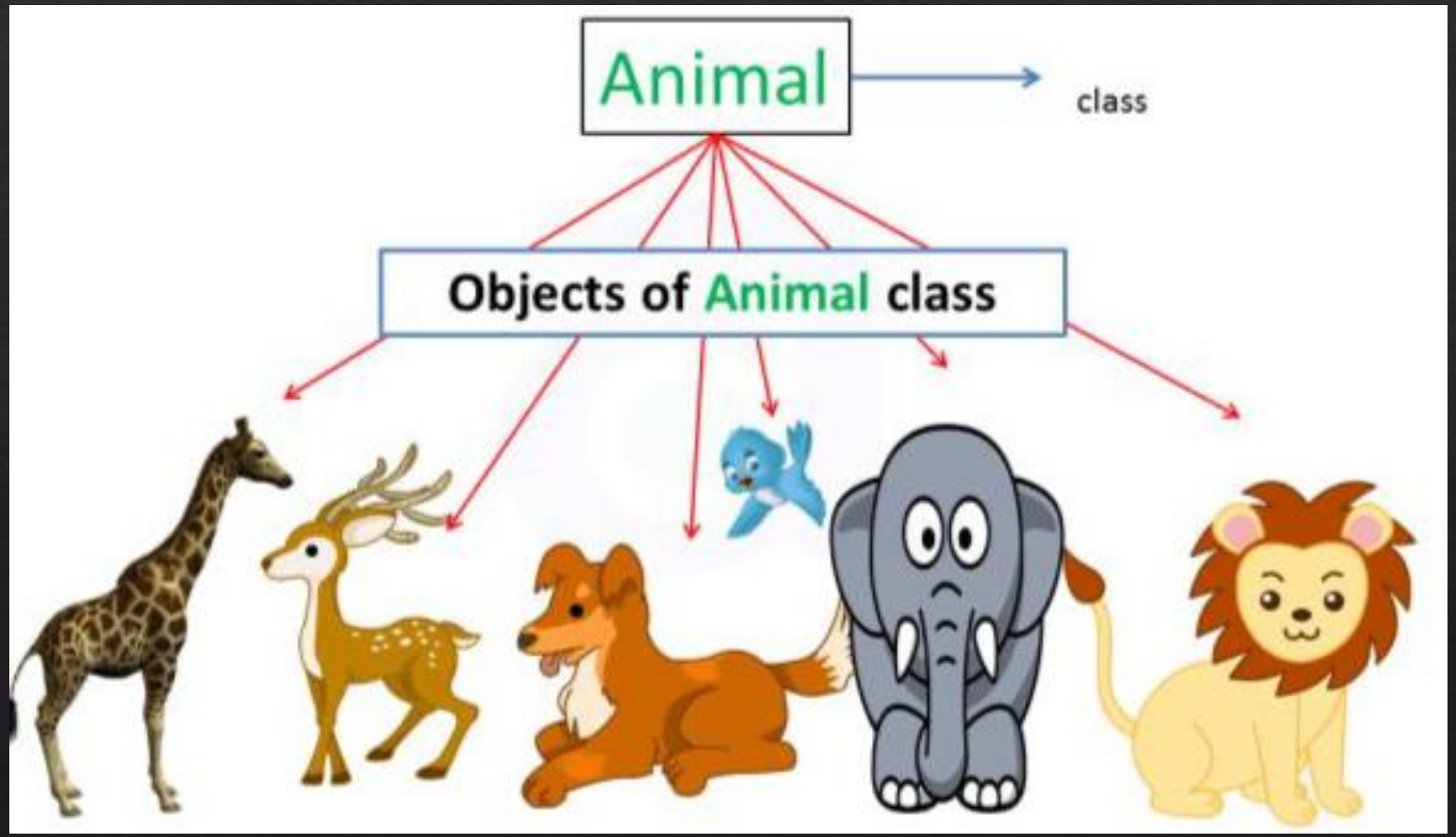

บัตรประจำตัวประชาชน Thai National ID Card
Identification Number
ชื่อตัวและชื่อสกุล
ชื่อ นามสกุล
Name
Last name
Date of Birth
Date of Issue
Date of Expiry

การสืบทอดคุณสมบัติ (Inheritance)

หลักการของ inheritance คือ ทำการสร้างสิ่งใหม่ขึ้นด้วยการสืบทอด หรือรับเอา (inherit) คุณสมบัติบางอย่างมาจากสิ่งเดิมที่มีอยู่แล้ว โดยการสร้างเพิ่มเติมจากสิ่งที่มีอยู่แล้วได้เลย ข้อดีของการ inheritance คือ จากการที่สามารถนำสิ่งที่เคยสร้างขึ้นแล้วนำกลับมาใช้ใหม่ (re-use) ได้ ทำให้ช่วยประหยัดเวลาการทำงานลง เนื่องจากไม่ต้องเสียเวลาพัฒนาใหม่หมด



คลาสแม่ [Superclass] คลาสลูก [Subclass]

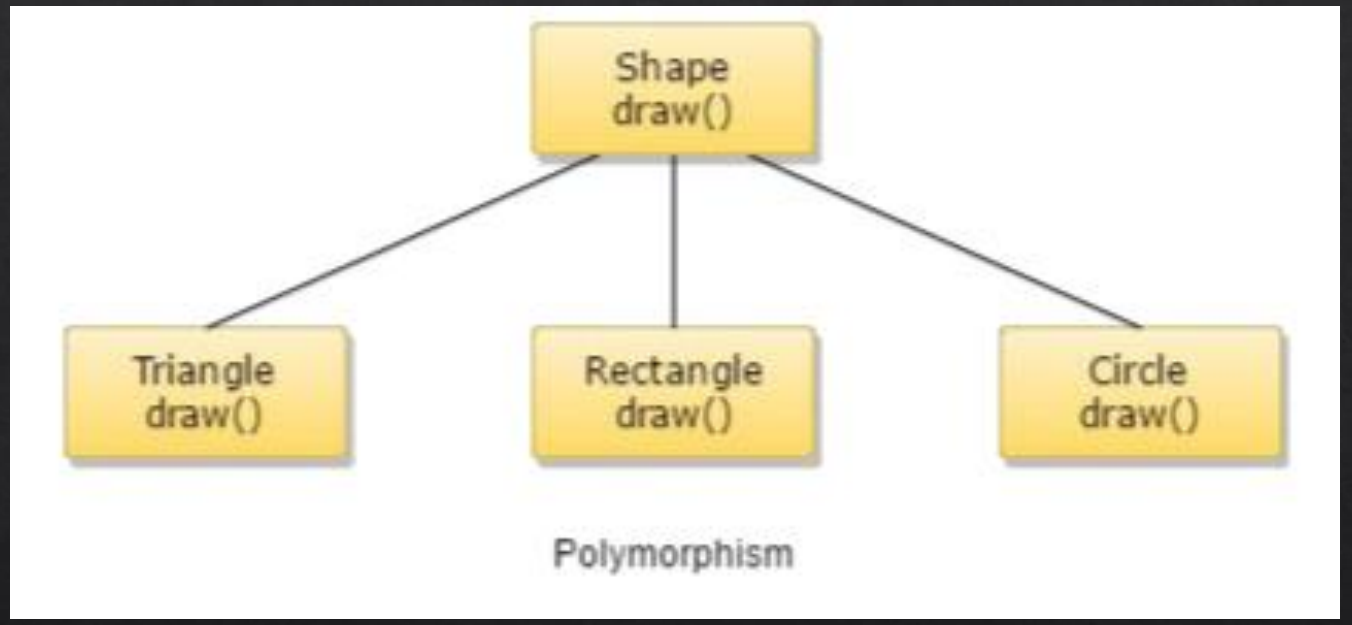


Employee


ผู้จัดการ
- รหัสพนักงาน
- ชื่อ
- เงินเดือน
- ที่จอดรถ
+ คำนวณเงินเดือน()
+ แสดงรายละเอียด()
พนักงานขาย
- รหัสพนักงาน
- ชื่อ
- เงินเดือน
- ค่าคอมมิชั่น
+ คำนวณเงินเดือน()
+ แสดงรายละเอียด()
พนักงานฝ่ายผลิต
- รหัสพนักงาน
- ชื่อ
- เงินเดือน
- ค่าล่วงเวลา
+ คำนวณเงินเดือน()
+ แสดงรายละเอียด()

การพ้องรูป (POLYMORPHISM)

Polymorphism เกิดจาก poly (หลากหลาย) + morphology (รูปแบบ)



ในทางโปรแกรม คือการที่เมธอดชื่อเดียวกัน สามารถรับอาร์กิวเมนต์ที่แตกต่างกัน ได้หลายรูปแบบ โดยเมธอดนี้จะถูกเรียกว่า **overload method** (เมธอดถูกโอเวอร์โหลด)

สรุป

- Class - ต้นแบบของวัตถุ
- Object - สิ่งที่ถูกสร้างขึ้นมาจาก Class ประกอบด้วย
    - คุณสมบัติ (Attribute)
    - พฤติกรรม (Method)

- **คุณสมบัติของการเขียนโปรแกรมเชิงวัตถุ**
    - การห่อหุ้ม(Encapsulation)
    - การสืบทอด (Inheritance)
    - การพ้องรูป (POLYMORPHISM)

คลาส MaterialApp

ในการสร้างแอพพลิเคชันด้วย Flutter จะต้องมีการเรียกใช้งานเครื่องมือต่างๆ ที่ช่วยในการออกแบบโครงร่าง เรียกว่า **Widget** ซึ่งจะถูกสร้างโดยผ่านการเรียกใช้งานคลาสที่มีชื่อว่า **MaterialApp** เพื่อควบคุมการแสดงผลหน้าจอ รวมถึงการกำหนดเส้นทางการปรับเปลี่ยนฉากหรือหน้าจอของแอพ

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: Scaffold());
      appBar: AppBar(),
   }
}
```

Reference

- KongRuksiam Official. https://www.youtube.com/c/KongRuksiamOfficial
- เอกรินทร์ วทัญญูเลิศสกุล. (2563). พัฒนา Mobile App ด้วย Flutter & Dart. โปรวิชั่น, บจก.
- จีราวุธ วารินทร์. (2564). ต่อยอดพัฒนาโมไบล์แอพด้วย Flutter + Firebase. ซิมพลิฟาย, สนพ.