<u>ใบงานการทดลองที่ 4</u>

เรื่อง ภาษา Dart

1. จุดประสงค์

1. เขียน class และเรียกใช้ class ที่สร้างขึ้นได้
2. เขียนประกาศ property และสร้าง method ได้อย่างถูกต้อง
3. เขียนประกาศ constructor ใน class ในได้อย่างถูกต้อง

2. ทฤษฎี

บทที่ 2.2 การเขียนโปรแกรมแบบ OOP

การเขียนโปรแกรมแบบ OOP (Object-oriented programming) เป็นรูปแบบการเขียนโปรแกรมที่เน้นการสร้างวัตถุ (object) ขึ้นมาแทนที่การเขียนโปรแกรมแบบดั้งเดิมที่เน้นการสร้างตัวคุณสมบัติและพฤติกรรม วัตถุใน OOP ประกอบด้วยคุณสมบัติและพฤติกรรม ที่สามารถทำงานร่วมกันได้ ในภาษา Dart วัตถุจะถูกสร้างจากคลาส (class) คลาสเป็นแม่แบบสำหรับการสร้างวัตถุใหม่ คลาสสามารถมีตัวแปรและฟังก์ชันได้ คุณสมบัติของคลาสเรียกว่าพร็อปเพอร์ตี้ (Property) พฤติกรรมของคลาสเรียกว่า เมธอด (Method)

2.2.1 คลาส (class)

สำหรับคลาสจะเป็นที่รวมของออบเจ็กต์หลาย ๆ ออบเจ็กต์ที่มีลักษณะเดียวกัน แต่อาจมีข้อมูลประจำตัวหรือคุณลักษณะต่างกัน เช่น รถจักรยานหลาย ๆ คัน อาจมีสีต่างกันมีขนาดต่างกัน แต่ก็เรียกรวมว่ารถจักรยาน ดังนั้นในการเขียนโปรแกรมเชิงวัตถุนั้นจะต้องรู้จักการนิยามคลาส (class) คลาสจะเป็นการจัดกลุ่มของออบเจ็กต์ที่มีคุณลักษณะและพฤติกรรมบ้างอย่างเหมือนกันเสมือนพิมพ์เขียวของออบเจ็กต์ หรือเป็นแม่แบบสำหรับออบเจ็กต์ ตัวอย่างเช่น หากพิจารณาออบเจ็กต์ที่เป็นรถ ไม่ว่ารถสีเขียว สีดำ สีขาว ขนาดเล็ก ขนาดกลาง ขนาดใหญ่ วิ่งได้ หรือจอดอยู่เราก็เรียกว่าออบเจ็กต์เหล่านั้นอยู่ในคลาสเดียวกัน โดยอาจจะให้ชื่อคลาสนั้นชื่อคลาส รถ

โครงสร้างคลาส

```
Class className {
        //property
        //method
        //Constructor
}
```

ตัวอย่างการสร้างคลาส

```
class Dog{
  late String size ; // property
  late int age ; // property
  late String color; // property
  void Eat(){ // method
    print("The dog is eating !");
  } //method
```

2.2.2 ออบเจ็กต์

ในคลาสถ้าหากมีการประกาศตัวแปรหรือสร้างข้อมูลขึ้นมา ข้อมูลนั้นก็จะถูกใช้ในออบเจ็กต์นั้น ๆ การกระทำกับออบเจ็กต์จะกระทำผ่านเมธอดของคลาสนั้น ๆ ส่วนสำคัญของออบเจ็กต์คือ Property ซึ่งเป็นข้อมูลของออบเจ็กต์ และ Method เป็นตัวบอกว่าออบเจ็กต์กำลังทำสิ่งใดอยู่ ตัวอย่าง ถ้านาฬิกาปลุกเป็นออบเจ็กต์

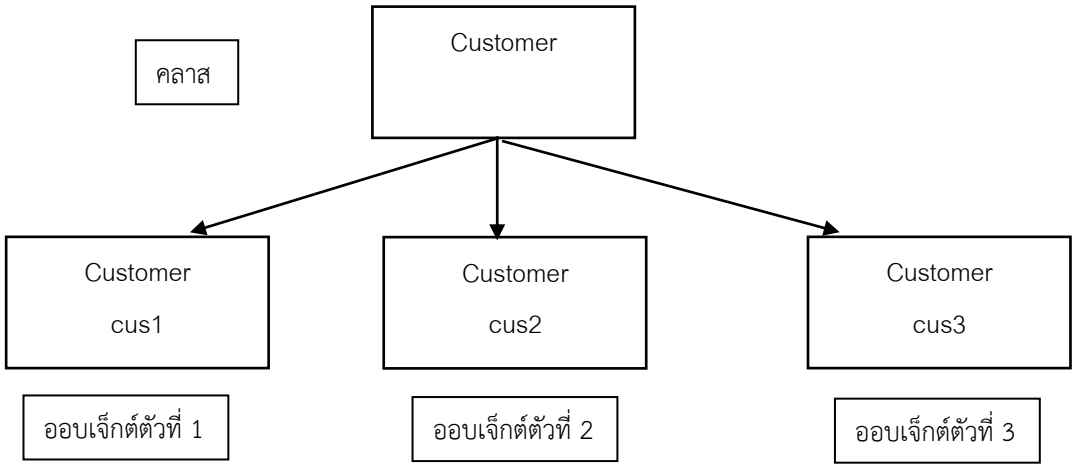
Property ได้แก่

- เวลาเป็นวินาที
- เวลาเป็นนาที
- เวลาเป็นชั่วโมง
- เวลาที่ตั้งปลุก
- ต้องการให้ปลุกหรือไม่ปลุก

Method ได้แก่

- เมธอดที่ใช้ตั้งเวลา
- เมธอดที่ใช้ตั้งเวลาปลุก
- เมธอดที่ให้ปลุกได้
- เมธอดที่ไม่ให้ปลุก

ในการเขียนโปรแกรมเชิงวัตถุนั้นจะไม่มีการใช้งานคลาสตรง ๆ แต่จะใช้คลาสเป็นพิมพ์เขียวเพื่อสร้างออบเจ็กต์ต่าง ๆ ขึ้นมา ออบเจ็กต์ที่ถูกสร้างจากคลาสจะเรียกว่า “อินสแตนซ์” (instance) ของคลาสนั้น โดยที่คลาสสามารถสร้างออบเจ็กได้หลายตัวแต่ละตัวจะมีชื่อที่แตกต่างกันไป ตัวอย่างเช่น หากมีคลาสของลูกค้า ชื่อ Customer เราสามารถสร้างออบเจ็กต์ของลูกค้าหลาย ๆ คนได้ เช่น cus1, cus2, cus3 เป็นต้น



โครงสร้างออบเจ็กต์

```
Classname propertyname = Classname();
```

ตัวอย่างการสร้างออบเจ็กต์

```
Person person = Person();
```

2.2.3 Property

Property คือ การจัดเก็บข้อมูลบางอย่างของคลาส เพื่อที่จะนำไปใช้ต่อ เช่น ถ้าเราสร้างคลาส รถ ก็ควรมี Property ที่เก็บ รุ่นรถ สีรถ ขนาดรถ เครื่องยนต์ เป็นต้น โดยการกำหนด Property ก็เหมือนกับการสร้างตัวแปรขึ้นมาในคลาส ซึ่งเราต้องระบุชนิดข้อมูล ชื่อProperty และต้องกำหนดค่าdefaultให้กับมันเสมอ ยกเว้นกรณี Nullable Type และการกำหนดภายหลัง

โครงสร้าง Property

```
Class className {
        Datatype propertyname =value;
        //method
        //Constructor
}
```

ตัวอย่างการสร้าง Property

```
class Dog{
    String size = “medium” ;
    int age = 1;
    String color = “black”;
```

2.2.4 Method

เมธอด (Method) คือ สมาชิกของคลาสอีกอย่างหนึ่ง ซึ่งใช้สำหรับการกำหนดการกระทำต่าง ๆ โดยการสร้างเมธอดจะใช้หลักการเดียวกับฟังก์ชันทั้งหมด แต่เมธอดจะต้องเรียกใช้งานผ่านอินสแตนซ์ของคลาสนั้น

โครงสร้างของ Method

```
Retruntype Methodname {
        Method
}
```

ตัวอย่างการสร้าง Method

```
void Eat(){
   print("The dog is eating !");
 }
```

2.2.5 การเข้าถึง Property และ Method

การเข้าถึง Property และ Method จะใช้วิธีเดียวกันในการเข้าถึง โดยการทำผ่านอินสแตนซ์ของคลาสด้วยกันใช้ **(.)** โดยใช้รูปแบบดังนี้

ตัวอย่างการเข้าถึง Property และ Method

```
class Dog{
  late String size ; //Property
  late int age ; //Property
  void Eat(){ //Method
    print("The dog is eating !");
  }
void main(){
  Dog dog1 = Dog(); //objcet
  dog1.size = "samll"; // การเข้าถึง Property
  dog1.age = 1; // การเข้าถึง Property
  dog1.Eat(); // การเข้าถึง Method
```

2.2.6 คอนสตรักเตอร์ของคลาส

คอนสตรักเตอร์ของคลาส (constructor) เป็นเมธอดพิเศษที่เรียกใช้เมื่อวัตถุของคลาสถูกสร้างขึ้น คอนสตรักเตอร์สามารถใช้เพื่อกำหนดค่าเริ่มต้นให้กับพร็อปเพอร์ตี้ของวัตถุหรือเพื่อดำเนินการอื่นๆ ที่ต้องการเมื่อวัตถุถูกสร้างขึ้น

โดยลักษณะสำคัญของคอนสตรักเตอร์มีคราวๆดังนี้

- คอนสตรักเตอร์ของคลาสต้องมีชื่อเดียวกับคลาส
- คอนสตรักเตอร์ต้องไม่มีการส่งค่ากลับ แต่ไม่ต้องระบุ void นำหน้า เพราะมันจะทำให้กลายเป็นเมท็อด
- การรับพารามิเตอร์เข้ามาในคอนสตรักเตอร์ สามารถนำหลักการรับพารามิเตอร์ของฟังก์ชันมาใช้ได้ทั้งหมด

รูปแบบของคอนสตรักเตอร์ในภาษา Dart ดังนี้

Constructor จะการแบ่งออกเป็นสองประเภทคือ Default constructor และ Parameterized constructor หรือ คอนสตรัคเตอร์ที่รับพารามิเตอร์

Default constructor

ในทุกคลาสจะมี constructor ที่เป็น Default constructor อยู่ 1 อันเสมอ ซึ่งเป็น constructor ที่ถูกสร้างขึ้นมาโดยอัตโนมัติ จะมีชื่อเดียวกับชื่อคลาสและไม่มีการรับค่าพารามิเตอร์ใด ๆ

Parameterized constructors

จะเป็น constructor ที่สามารถรับค่าพารามิเตอร์ได้ โดยวิธีการรับค่าของพารามิเตอร์ของ constructor ก็จะมีการรับแบบเดียวกันกับวิธีการรับพารามิเตอร์ของฟังก์ชัน

โครงสร้าง Constructor

```
Classname (parameter){


}
```

ตัวอย่างการสร้าง Constructor

```
class Dog{
  late String size ;
  late int age ;
  late String color;
  Dog(String size, int age, String color){
    this.size = size = "not";
//this.size อ้างถึง Property size
//size อ้างถึง parameter size
    this.age = age = 0;
    this.color = color = "not";
  }
```

Short-form

หลักการของ short-form คือ เราจะนำพร็อปเพอร์ตี้มาตั้งเป็นพารามิเตอร์ ดังนั้น ให้ระบุพารามิเตอร์ในรูปแบบ this.ชื่อพร็อปเพอร์ตี้ ไม่ต้องระบุชนิดเพราะกำหนดไว้แล้วที่ขั้นตอนการสร้างพร็อปเพอร์ตี้ ไม่ต้องมีบล็อกในการนำค่าไปกำหนดให้แก่พร็อปเพอร์ตี้อีก เพราะค่าที่รับเข้ามาจะถูกกำหนดให้แก่พร็อปเพอร์ตี้โดยอัตโนมัติ

โครงสร้าง Short – form

```
Classname (this.parameter);
//ไม่ต้องมีการกำหนดค่า เพราะค่าที่รับเข้ามาจะถูก
กำหนดให้แก่ Property โดยอัตโนมัติ
```

ตัวอย่างการสร้าง Short – form

```
class Dog{
  late String size ;
  late int age ;
  late String color;
  Dog(this.size,this.age,this.color);
```

2.2.7 การสืบทอดระหว่างคลาส

การสืบทอด (inheritance) เป็นแนวคิดของโปรแกรมเชิงวัตถุ (object-oriented programming) ที่ช่วยให้เราสามารถสร้างคลาสใหม่จากคลาสที่มีอยู่เดิม โดยคลาสใหม่จะมีคุณสมบัติและพฤติกรรมของคลาสที่มีอยู่เดิมบางส่วนหรือทั้งหมด

คลาสที่ทำหน้าที่เป็นพิมพ์เขียวสำหรับคลาสใหม่เรียกว่า คลาสแม่ (superclass) และคลาสใหม่ที่สืบทอดจากคลาสแม่เรียกว่า คลาสลูก (subclass)

การสืบทอดช่วยให้เราสามารถประหยัดเวลาในการพัฒนาซอฟต์แวร์ได้ เนื่องจากเราสามารถนำคุณสมบัติและพฤติกรรมของคลาสที่มีอยู่เดิมมาใช้ซ้ำในคลาสใหม่ได้ รูปแบบการเขียนดังข้างล่าง

โครงสร้าง

โครงสร้างการสืบทอดคลาส

```
class supclassname extends superclassname {
        //property
        //method
        //Constructor
}
```

ตัวอย่างการสืบทอดคลาส

```
class Animal {
  String name;

  void speak() {
    print('I am an animal.');
  }
}

class Dog extends Animal {
  void bark() {
    print('Woof!');
  }
}
```

```
void main() {
  // สร้างวัตถุจากคลาส Dog
  Dog dog = Dog();
  dog.name = 'Fido';

  // เข้าถึงคุณสมบัติ name
  print(dog.name); // Fido

  // เรียกใช้เมธอด speak
  dog.speak(); // I am an animal.

  // เรียกใช้เมธอด bark
  dog.bark(); // Woof!     }
```

คำสั่ง ให้นักศึกษาเขียนคำตอบตามที่โจทย์กำหนดให้ถูกต้อง (สามารถแนบรูปโค้ดและผลลัพธ์คำตอบของโปรแกรมได้)

1. ให้สร้าง **class Car** โดยมี properties ตามกล่องข้อความด้านล่าง และปฏิบัติตามดังต่อไปนี้

1.1 สร้าง constructor ประกอบด้วยพารามิเตอร์ระบุชื่อดังนี้ brand, model, cc และ color

1.2 ประกาศตัวแปรวัตถุ car1, car2 และ car3 ใน main โดยใช้ constructor ข้อ 1.1

เช่น var car1 = Car(“Honda”, “civic”, 1500, “black”)

1.3 สร้าง Methods ทั้ง 3 อย่างนี้

1.3.1 start() – รถกำลังเคลื่อนที่

1.3.2 stop() – รถหยุดเคลื่อนที่

1.3.3 changeGear(number) – เปลี่ยนเกียร์ให้ใช้ได้แค่ 1, 2, 3, 4, 5 และ R นอกจากนี้ให้แสดงข้อความว่าไม่สามารถใช้งานเกียร์นี้ได้

1.4 ทดลองเรียกใช้ Method ทั้งสามตัวตามข้อ 1.3 ผ่าน main ตัวอย่างการเรียกใช้ เช่น car1.start()

class Student

```
class Student {
    late String brand;
    late String model;
    late int cc;
    late String color;

    Car(this.brand, this.model, this.cc, this.color);

    void start() {
        คำสั่ง..
    }
    void stop() {
        คำสั่ง..
    }
    void changeGear(int gear) {
        คำสั่ง..
    }
}
```

ตัวอย่าง output

```
รถ Honda กำลังเคลื่อนที่              // start
รถ Toyota หยุดเคลื่อนที่              // stop
รถ Mazda เปลี่ยนเป็นเกียร์ 3          // changeGear
รถ Toyota ไม่มีเกียร์ 7               // changeGear
```

```dart
class Car {
  late String brand;
  late String model;
  late int cc;
  late String color;
  Car(this.brand, this.model, this.cc, this.color);
  void start() {
    print('รถ ${brand} ${model} ขนาด $cc สี ${color} กำลังเคลื่อนที่');
  }

  void stop() {
    print('รถ ${brand} ${model} ขนาด $cc สี ${color} หยุดเคลื่อนที่');
  }

  void changeGear(dynamic gear) {
    if (gear is int && (gear >= 1 && gear <= 5)) {
      print('รถ $brand เปลี่ยนเป็นเกียร์ $gear');
    } else if (gear == 'R') {
      print('รถ $brand เปลี่ยนเป็นเกียร์ ${gear}');
    } else {
      print('รถ $brand ไม่สามารถใช้งานเกียร์ ${gear} ได้');
    }
  }
}

void main() {
  Car car1 = Car('Honda', 'Civic', 1500, 'Black');
  Car car2 = Car('Honda', 'Civic', 1500, 'red');
  Car car3 = Car('Mazda', 'Civic', 1500, 'Blue');
  car1.start();
  car2.stop();
  car3.changeGear(1);
  car1.changeGear(7);
  car3.changeGear('R');
}
```

```
รถ Honda Civic ขนาด 1500 สี Black กำลังเคลื่อนที่
รถ Honda Civic ขนาด 1500 สี red หยุดเคลื่อนที่
รถ Mazda เปลี่ยนเป็นเกียร์ 1
รถ Honda ไม่สามารถใช้งานเกียร์ 7 ได้
รถ Mazda เปลี่ยนเป็นเกียร์ R
```

2. ให้สร้าง **class Student** โดยมี properties ตามกล่องข้อความด้านล่าง และปฏิบัติตามดังต่อไปนี้

2.1 สร้าง constructor ประกอบด้วยพารามิเตอร์ระบุชื่อดังนี้ id, firstName, lastName และ gpa

2.2 ประกาศตัวแปรวัตถุ student1, student2 และ student3 ใน main โดยใช้ constructor ข้อ 2.1
เช่น var student1 = Student(1, “Phakaphol”, “Kongtoom”, 3.11)
(ให้กำหนดค่าเริ่มต้นของ gpa เป็น 0.00 ไว้ด้วย ไว้ใช้ในกรณีไม่มีการส่งค่า gpa มา)

2.3 สร้าง Methods ทั้ง 3 อย่างนี้

2.3.1 changeName(firstname, lastname) – ใช้สำหรับเรียกใช้เพื่อเปลี่ยนชื่อ-นามสกุล

2.3.2 showInfo() – ใช้สำหรับเรียกใช้เพื่อแสดงข้อมูลทั้งหมดของนักเรียน

2.3.3 checkGrade() - ใช้สำหรับตรวจสอบว่าเราได้ผลการเรียนในระดับใด อิงจากค่า gpa ที่ได้ โดยมีทั้งหมด 5 ระดับคือ A (4.00), B (3.00 ขึ้นไป), C (2.00 ขึ้นไป), D (1.00 ขึ้นไป) และ F (ต่ำกว่า 1.00) เช่น gpa 2.11 จะได้เกรด C เป็นต้น

2.4 ทดลองเรียกใช้ Method ทั้งสามตัวตามข้อ 2.3 ผ่าน main เช่น student1.showInfo()

class Student

```
class Student {
    late int id;
    late String firstName;
    late String lastName;
    late double gpa;

    Student(this.id, this.firstName, this.lastName, {this.gpa = 0.00});

    void changeName({required String stdFirstname, required String stdLastname}) {
        คำสั่ง..
    }
    void showInfo() {
        คำสั่ง..
    }
    void checkGrade() {
        คำสั่ง..
    }
}
```

output

```
Student [1] Phakaphol Kongtoom GPA 0                 // showInfo
.. Student [1] เปลี่ยนชื่อเรียบร้อย ..               // changeName
Student [1] Ninja HaHa GPA 0                         // showInfo
Student [2] Paripat Taseeruen GPA 2.67               // showInfo
.. Student [3] GPA 3.11 ได้เกรด B ..                 // checkGrade
Student [3] Treenarubet Wongyai GPA 3.11             // showInfo
```

```dart
class Student {
  late int id;
  late String firstName;
  late String lastName;
  late double gpa;
  Student(this.id, this.firstName, this.lastName, {this.gpa = 0.00});
  void changeName({required String stdFirstname, required String stdLastname}) {
    if (stdFirstname is String) {
      firstName = stdFirstname;
    }
    if (stdLastname is String) {
      lastName = stdLastname;
    }
    print('Student ${id} เปลี่ยนชื่อเรียบร้อย ..');
  }

  void showInfo() {
    print('Student ${id} ชื่อ ${firstName} ${lastName} gpa: ${gpa}');
  }

  void checkGrade() {
    String i;
    switch (gpa) {
      case < 1.00:
        i = 'F';
        break;
      case < 2.00:
        i = 'D';
        break;
      case < 3.00:
        i = 'C';
        break;
      case < 4.00:
        i = 'B';
        break;
      default:
        i = 'A';
    }
    print('Student ${id} gpa: ${gpa} ได้เกรด ${i}');
  }
}

void main() {
  Student student1 = Student(1, 'Aee', 'wp', gpa: 3.11);
  Student student2 = Student(2, 'Bee', 'gg');
  Student student3 = Student(3, 'Cee', 'ez', gpa: 4.00);
  student1.showInfo();
  student1.changeName(stdFirstname: 'Dee', stdLastname: 'Eey');
  student1.showInfo();
  student2.showInfo();
  student3.checkGrade();
  student3.showInfo();
}
```

```
Student 1 ชื่อ Aee wp gpa: 3.11
Student 1 เปลี่ยนชื่อเรียบร้อย ..
Student 1 ชื่อ Dee Eey gpa: 3.11
Student 2 ชื่อ Bee gg gpa: 0
Student 3 gpa: 4 ได้เกรด A
Student 3 ชื่อ Cee ez gpa: 4
```

3. ให้สร้าง **class Account** โดยมี properties ตามกล่องข้อความด้านล่าง โดยปฏิบัติตามดังต่อไปนี้

3.1 สร้าง constructor ประกอบด้วยพารามิเตอร์ระบุชื่อดังนี้ accid, accName และ balance

3.2 ประกาศตัวแปรวัตถุ acc1 และ acc2 ใน main โดยใช้ constructor ข้อ 3.1

เช่น var acc1 = Account(1, “Ninja”, 1000)

3.3 สร้าง Methods ทั้ง 3 อย่างนี้

3.3.1 deposit(amount) – ฝากเงิน จำนวน amount (ต้องไม่ติดลบและเป็นศูนย์) ฝากได้ให้แสดงยอดเงินคงเหลือ

3.3.2 withdraw(amount) - ถอนเงิน จำนวน amount (เงินคงเหลือต้องไม่ติดลบ) ถอนได้ให้แสดงยอดเงินคงเหลือ

3.3.3 checkBalance() – เช็คยอดเงินคงเหลือ

3.4 จงหาว่า acc1 และ acc2 เหลือเงินในบัญชีเท่าไหร่ หากเป็นไปตามประโยคดังต่อไปนี้

(เรียกใช้ Methods ที่กำหนดมาเพื่อหาค่าเท่านั้น เช่น acc1.deposite(100))

“ acc1 มีเงินเริ่มต้นที่ 1,000 บาท ต่างจาก acc2 ที่มีเงินเริ่มต้นในบัญชีมากกว่า acc1 สองเท่า
ต่อมา acc1 ฝากเงินเพิ่มเข้าไปในบัญชี 200 บาท แต่ acc2 ถอนเงินออกเพื่อไปซื้อตุ๊กตา 500 บาท
acc1 เห็นตุ๊กตาที่ acc2 ซื้อ จึงอยากได้บ้าง แต่กดถอนเงินผิด กดถอนเงินไป 10,000 บาท ทำให้ไม่สามารถถอนเงินได้ จึงกดถอนเงินใหม่เป็นจำนวน 500 บาท เพื่อซื้อตุ๊กตาตัวนั้น
สรุปแล้วทั้ง acc1 และ acc2 เหลือเงินเท่าไหร่ ”

class Student

```
class Account {
    late int accId;
    late String accName;
    late double balance;

    Account(this.accId, this.accName, {this.balance = 500});

    void deposite(double amount) { คำสั่ง }
    void withdraw(double amount) { คำสั่ง }
    void checkBalance() { คำสั่ง }
}
```

output

..ฝากเงินจำนวน 200 บาท ไปยัง ID 1 สำเร็จ..

..ถอนเงินจำนวน 500 บาท จาก ID 2 สำเร็จ..

..ยอดเงินใน ID 1 ไม่เพียงพอ..

..ถอนเงินจำนวน 500 บาท จาก ID 1 สำเร็จ..

ตรวจสอบยอดเงิน

ID 1 : acc1 มียอดคงเหลือ 700 บาท

ตรวจสอบยอดเงิน

ID 2 : acc2 มียอดคงเหลือ 1500 บาท

```dart
class Account {
  late int accId;
  late String accName;
  late double balance;
  Account(this.accId, this.accName, {this.balance = 500});
  void deposite(double amount) {
    if (balance > 0) {
      balance = balance + amount;
    }
    print('..ฝากเงินจำนวน ${amount} บาท ไปยัง ID ${accId} สำเร็จ..');
  }

  void withdraw(double amount) {
    double check_balance = balance - amount;
    if (check_balance < 0) {
      print('..ยอดเงินใน ID ${accId} ไม่เพียงพอ..');
    } else {
      balance = check_balance;
      print('..ถอนเงินจำนวน ${amount} บาท ไปยัง ID ${accId} สำเร็จ..');
    }
    check_balance = 0;
  }

  void checkBalance() {
    print('ID ${accId} : ${accName} มียอดคงเหลือ ${balance} บาท');
  }
}

void main() {
  Account acc1 = Account(1, 'acc1', balance: 1000);
  Account acc2 = Account(2, 'acc2', balance: 2000);
  acc1.deposite(200);
  acc2.withdraw(500);
  acc1.withdraw(10000);
  acc1.withdraw(500);
  print('ตรวจสอบยอดเงิน');
  acc1.checkBalance();
  print('ตรวจสอบยอดเงิน');
  acc2.checkBalance();
}
```

```
..ฝากเงินจำนวน 200 บาท ไปยัง ID 1 สำเร็จ..
..ถอนเงินจำนวน 500 บาท ไปยัง ID 2 สำเร็จ..
..ยอดเงินใน ID 1 ไม่เพียงพอ..
..ถอนเงินจำนวน 500 บาท ไปยัง ID 1 สำเร็จ..
ตรวจสอบยอดเงิน
ID 1 : acc1 มียอดคงเหลือ 700 บาท
ตรวจสอบยอดเงิน
ID 2 : acc2 มียอดคงเหลือ 1500 บาท
```

5. สรุปผลการทดลอง

ในการทดลองจะพบว่าclassภายในภาษา Dark คือแบบจำลองของข้อมูลและพฤติกรรมที่คล้ายคลึงกัน
คลาสสามารถใช้ในการจัดระเบียบโค้ดและทำให้โค้ดมีความซ้ำซ้อนน้อยลง
class เริ่มต้นด้วยคำสำคัญ class ตามด้วยชื่อของคลาส จากนั้นตามด้วยโค้ดที่นิยามคลาส โค้ดนี้สามารถประกอบด้วยสิ่งต่อไปนี้:
Attributes: คุณสมบัติคือข้อมูลที่เกี่ยวข้องกับวัตถุ คุณสมบัติสามารถกำหนดค่าเริ่มต้นได้หรือกำหนดค่าในภายหลัง
Methods: วิธีการคือฟังก์ชันที่เกี่ยวข้องกับวัตถุ วิธีการสามารถใช้เพื่อดำเนินการกับวัตถุหรือเพื่อรับหรือตั้งค่าคุณสมบัติของวัตถุ
Constructors: ตัวสร้างคือวิธีการพิเศษที่ใช้ในการสร้างวัตถุ ตัวสร้างสามารถกำหนดค่าคุณสมบัติของวัตถุเริ่มต้นได้

6. คำถามหลังการทดลอง

6.1 Property และ Method ในคลาส (class) มีส่วนสำคัญอย่างไรบ้าง

Property และ Method เป็นส่วนสำคัญในคลาส (class) ของภาษา Dart ทำหน้าที่เก็บข้อมูลและการทำงานที่เกี่ยวข้องกันของวัตถุ (object) ช่วยให้โปรแกรมเมอร์สามารถจัดระเบียบโค้ดและทำให้โค้ดมีความซ้ำซ้อนน้อยลง
Property คือข้อมูลที่เกี่ยวข้องกับวัตถุ อาจเป็นค่าคงที่ (constant) หรือค่าตัวแปร (variable) Property สามารถกำหนดค่าเริ่มต้นได้หรือกำหนดค่าในภายหลัง