List in Dart (start-ex4)

จัดทำโดย

630710340 สุธีมนต์ ยศยิ่ง

รายงานนี้เป็นส่วนหนึ่งของรายวิชา

517321

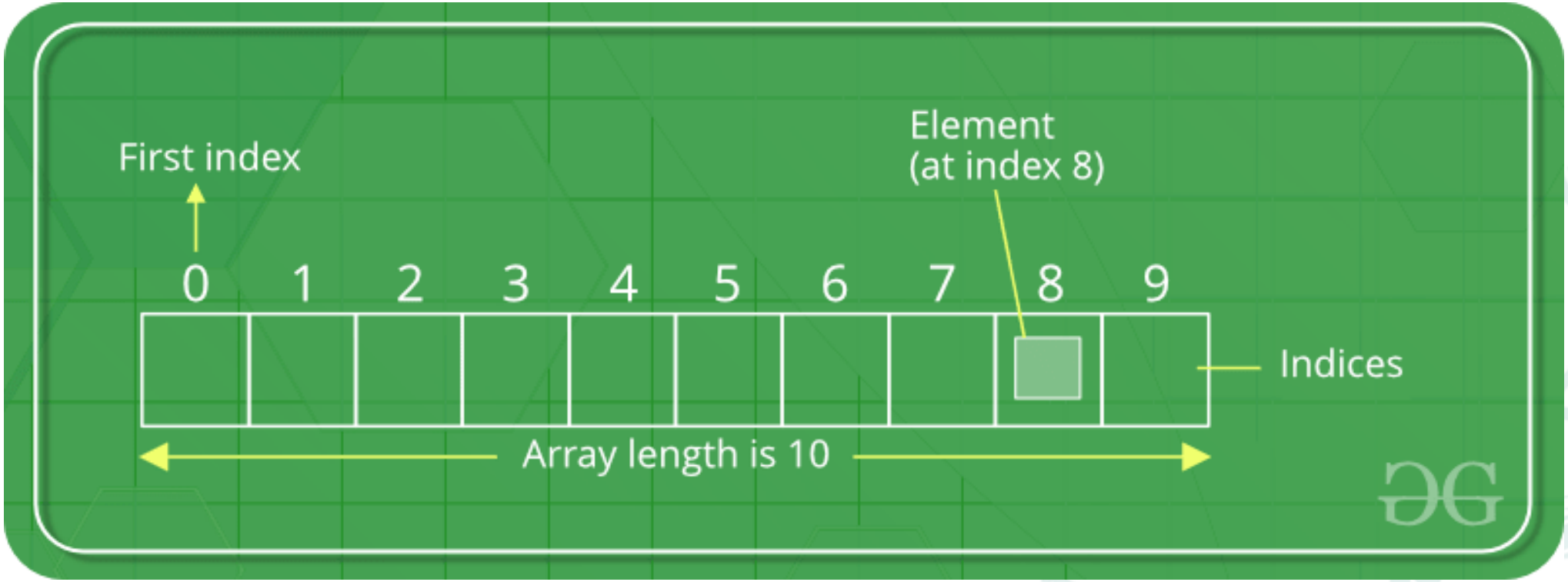
PRINCIPLES OF PROGRAMMING LANGUAGES

List in Dart คืออะไร?

List ในภาษา Dart ใช้ในการเก็บข้อมูลหลายๆ ประเภทในพื้นที่จัดเก็บที่สร้างขึ้น โดยจะเรียงลำดับของข้อมูลด้วย

โดย List จะคล้ายกับ Array ในภาษาโปรแกรมอื่นๆ โดยขอยกตัวอย่างภาษา Java แต่จะแตกต่างกันในส่วนของ Array นั้นจำเป็นต้องจัดเก็บข้อมูลประเภทเดียวกัน ไม่สามารถเก็บข้อมูล หรือตัวแปรต่างประเภทกันได้

Logical of List



จากภาพจะเห็นได้ว่าการเก็บข้อมูลนั้นจะเก็บเป็นส่วนๆ โดยจะมีเลข Index ของแต่ละตำแหน่งของข้อมูลที่เราทำการจัดเก็บ การที่จะนำข้อมูลมาแสดงผล จำเป็นต้องใช้เลขตำแหน่ง หรือ Index นั่นเอง

วิธีการสร้าง List

- การสร้าง List เราจะใช้ [ ] (วงเล็บก้ามปู) ในการ fill ข้อมูล และหากมีการ fill ข้อมูลพร้อมกันหลายๆ ตัว จำเป็นต้องใส่เครื่องหมาย , (comma) คั่นระหว่างข้อมูล

Integer

```
List<int> ages = [10, 30, 23];
```

Mixed

```
var mixed = [10, "John", 18.8];
```

String

```
List<String> names = ["Raj", "John", "Rocky"];
```

การ fill ข้อมูลประเภท String ต้องใส่เครื่องหมาย ‘ ‘ (Single qoute) หรือ " " (Double qoute) ด้วย

การสร้าง List ในภาษาอื่นๆ

Java

```
//Creating a List of type String using ArrayList
List<String> list=new ArrayList<String>();

//Creating a List of type Integer using ArrayList
List<Integer> list=new ArrayList<Integer>();

//Creating a List of type Book using ArrayList
List<Book> list=new ArrayList<Book>();

//Creating a List of type String using LinkedList
List<String> list=new LinkedList<String>();
```

Python

```
# list with elements of different data types
list1 = [1, "Hello", 3.4]

# list with duplicate elements
list1 = [1, "Hello", 3.4, "Hello", 1]

# empty list
list3 = []
```

ประเภทของ List

List ถูกแบ่งออกเป็น 2 ประเภท

- Fixed Length List (List ที่กำหนดขอบเขต)
- Growable List (List ที่ไม่มีการกำหนดขอบเขต)

Fixed Length List

การสร้าง List โดยกำหนดขอบเขตความยาวของพื้นที่จัดเก็บไว้ก่อน จะไม่สามารถเปลี่ยนแปลงหรือเพิ่มความยาวของพื้นที่จัดเก็บได้อีกในภายหลัง

ตัวอย่างที่ 1

```
void main() {
  var list = List<int>.filled(5,0);
  print(list);
 }
```

โดยตัวอย่างนี้คือการประกาศตัวแปร list ที่มีข้อมูลด้านในเป็น 0 ทั้งหมด และมีความยาวที่ 5 (Index ที่ 0-4)

Output

```
[0, 0, 0, 0, 0]
```

Fixed Length List

ตัวอย่างที่ 2

```
void main()
  {
          List? gfg = List.filled(5, null, growable: false);
          gfg[0] = 'Geeks';
          gfg[1] = 'For';
          gfg[2] = 'Geeks';

          // Printing all the values in List (แสดงค่าทั้งหมดใน List)
          print(gfg);

          // Printing value at specific position (แสดงเฉพาะค่าในตำแหน่งที่ 3
(Index = 2))
          print(gfg[2]);
  }
```

Output

```
[Geeks, For, Geeks, null, null]
Geeks
```

ตัวอย่างนี้จะประกาศตัวแปร gfg ซึ่งเป็น List ที่มีความยาวอยู่ที่ 5 และข้อมูลใน List เป็น null จากนั้นมีการเพิ่มข้อมูลลงใน List 3 ตัว

Growable List

การสร้าง List โดยที่ไม่ได้มีการกำหนดขอบเขตความยาวของพื้นที่จัดเก็บไว้ก่อน ทำให้สามารถเปลี่ยนแปลงหรือเพิ่มความยาวของพื้นที่จัดเก็บได้ และเป็นที่นิยมมากกว่า Fixed Length List

ตัวอย่างที่ 1

```
void main() {
    var list1 = [210,21,22,33,44,55];
    print(list1);
   }
```

Output

```
[210, 21, 22, 33, 44, 55]
```

การเข้าถึงข้อมูลภายใน List

เราสามารถเข้าถึงข้อมูลใน List โดยใช้ Index หรือเลขตำแหน่ง ซึ่งจะเริ่มต้นที่ 0

```
void main() {
    var list = [210, 21, 22, 33, 44, 55];

    print(list[0]);
    print(list[1]);
    print(list[2]);
    print(list[3]);
    print(list[4]);
    print(list[5]);
  }
```

ตัวอย่างนี้จะแสดงข้อมูลออกมาทีละตำแหน่งตั้งแต่ 0 จนถึง 5

```
void main() {
    var scores = [1, 3, 4, 2];
    print(scores[2]);
  }
```

Output

```
4
```

ตัวอย่างนี้จะแสดงการเข้าถึงข้อมูลในตำแหน่งที่ 3 (Index = 2)

การเข้าถึงตำแหน่งโดยใช้ค่าของข้อมูลภายใน List

เราสามารถเข้าถึงตำแหน่งข้อมูลได้เช่นกัน โดย:

```
void main() {
    var list = [210, 21, 22, 33, 44, 55];

    print(list.indexOf(22));
    print(list.indexOf(33));
  }
```

Output

```
2
3
```

การหาความยาวของ List

การหาความยาวของ List สามารถใช้ **.length** ในการหาและแสดงผลออกมา

```
void main(){
        List<String> names = ["Raj", "John", "Rocky"];
        print(names.length);
}
```

Output

```
3
```

NOTE: List index จะเริ่มที่ 0 และ ความยาวจะเริ่มที่ 1 เสมอ

การเปลี่ยนค่าใน List

เราสามารถเปลี่ยนค่าใน List โดยใช้ listName[index]=value;

```
void main(){
        List<String> names = ["Raj", "John", "Rocky"];
        names[1] = "Bill"; //แทนค่าตำแหน่งที่ 2 Index = 1 ด้วย "Bill"
        names[2] = "Elon"; //แทนค่าตำแหน่งที่ 3 Index = 2 ด้วย "Elon"
        print(names);
}
```

Output

```
[Raj, Bill, Elon]
```

Mutable And Immutable List

list ที่ ตัวแปรเปลี่ยนค่าได้ (Mutable List) สามารถเปลี่ยนแปลงค่าได้หลังจากการประกาศค่าไปแล้ว และ List ที่ตัวแปรเปลี่ยนค่าไม่ได้ (Immutable List) ไม่สามารถเปลี่ยนแปลงค่าได้หลังจากการประกาศค่าไปแล้ว

```
List<String> names = ["Raj", "John", "Rocky"]; // Mutable List
  names[1] = "Bill"; // possible
  names[2] = "Elon"; // possible

const List<String> names = ["Raj", "John", "Rocky"]; // Immutable List
  names[1] = "Bill"; // not possible
  names[2] = "Elon"; // not possible
```

จะสังเกตได้ว่าก่อนหน้าการประกาศ List ของ String ตัวที่ 2 มี const อยู่ ทำให้เป็น Immutable list ที่ไม่สามารถเปลี่ยนค่าได้นั่นเอง

คุณสมบัติของ List ในภาษา Dart

1) **first:** ใช้คืนค่าของข้อมูลหรือค่าในตำแหน่ง "แรก" ใน List
2) **last:** ใช้คืนค่าของข้อมูลหรือค่าในตำแหน่ง "สุดท้าย" ใน List
3) **isEmpty:** จะคืนค่า true เมื่อภายใน List มีข้อมูล และจะคืนค่า false เมื่อ List นั้นว่างเปล่า
4) **isNotEmpty:** จะตรงข้ามกับ **isEmpty** คือ จะคืนค่า true เมื่อ List นั้นว่างเปล่า และจะคืนค่า false เมื่อ List นั้นมีข้อมูล
5) **length:** จะคืนค่าความยาวของ List
6) **reversed:** จะคืนค่าใน List จากตำแหน่งสุดท้ายจนถึงตำแหน่งแรก หรือการแสดงค่านับจากหลังมาหน้านั่นเอง
7) **single:** ใช้ในการตรวจสอบว่าใน List นั้นมีตัวแปรประเภทเดียวหรือไม่ และคืนค่ากลับไป

การเข้าถึงค่าแรกและค่าสุดท้ายของ List

เราสามารถเข้าถึงค่าตัวแรกและตัวสุดท้ายได้โดย :

```
void main() {
 List<String> drinks = ["water", "juice", "milk", "coke"];
 print("First element of the List is: ${drinks.first}");
 print("Last element of the List is: ${drinks.last}");
}
```

Output

```
First element of the List is: water
Last element of the List is: coke
```

การตรวจสอบว่าเป็น List ที่ว่างเปล่าหรือไม่

```
void main() {
  List<String> drinks = ["water", "juice", "milk", "coke"];
  List<int>  ages = [];
  print("Is drinks Empty: "+drinks.isEmpty.toString());
print("Is drinks not Empty: "+drinks.isNotEmpty.toString());
  print("Is ages Empty: "+ages.isEmpty.toString());
  print("Is ages not Empty: "+ages.isNotEmpty.toString());
}
```

Output

```
Is drinks Empty: false
Is drinks not Empty: true
Is ages Empty: true
Is ages not Empty: false
```

- จะเห็นได้ว่าการแสดงค่าออกมาเป็น true และ false โดยกรณีใช้ isEmpty, List ที่มีค่าจะคืนค่า false ส่วนที่ว่างเปล่าจะคืนค่า true , กรณีที่ใช้ isNotEmpty, List ที่มีค่าจะคืนค่า true ส่วนที่ว่างเปล่าจะคืนค่า false

การกลับด้าน List ในภาษา Dart

การกลับด้าน List สามารถทำได้ง่ายๆ โดยใช้คำสั่ง **.reversed**

```
void main() {
  List<String> drinks = ["water", "juice", "milk", "coke"];
  print("List in reverse: ${drinks.reversed}");
}
```

Output

```
List in reverse: (coke, milk, juice, water)
```

จะเห็นว่าเราได้ List ที่กลับด้านโดยพิมพ์จากด้านหลังมาด้านหน้า

การเพิ่มข้อมูล หรือค่าลงใน List

คำสั่งที่สามารถใช้ได้มีดังนี้

1) **add():** คือการเพิ่มค่า 1 ตัว ใน 1 ครั้ง
2) **addall():** คือการเพิ่มค่าหลายๆตัวใน 1 ครั้ง โดยแต่ละตัวจะคั่นด้วย , (comma) ใน [ ] (วงเล็บก้ามปู)
3) **insert():** ใช้เพิ่มค่าลงไปโดยระบุตำแหน่ง (index) ที่ต้องการเพิ่มลงไป
4) **insertall:** ใช้เพิ่มค่าหลายๆ ตัวโดยระบุตำแหน่ง (index) ที่ต้องการลงไป

ตัวอย่างการใช้คำสั่งเพิ่มข้อมูลในภาษา Dart

```
void main() {
   var evenList = [2,4,6,8,10];
   print(evenList);
   evenList.add(12);
   print(evenList);
}
```

Output

```
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10, 12]
```

โปรแกรมนี้จะแสดงตัวอย่างการ
ใช้ add() เพิ่มค่าลงใน List

```
void main() {
 var evenList = [2, 4, 6, 8, 10];
 print(evenList);
 evenList.addAll([12, 14, 16, 18]);
 print(evenList);
}
```

Output

```
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

โปรแกรมนี้จะแสดงตัวอย่างการ
ใช้ addall() เพิ่มค่าหลายๆ ตัวลงใน List

ตัวอย่างการใช้คำสั่งเพิ่มข้อมูลในภาษา Dart

```
void main() {
  List myList = [3, 4, 2, 5];
  print(myList);
  myList.insert(2, 15);
  print(myList);
}
```

Output

```
[3, 4, 2, 5]
[3, 4, 15, 2, 5]
```

โปรแกรมนี้จะแสดงตัวอย่างการใช้ insert() เพิ่มค่าลงใน List

```
void main() {
  var myList = [3, 4, 2, 5];
  print(myList);
  myList.insertAll(1, [6, 7, 10, 9]);
  print(myList);
}
```

Output

```
[3, 4, 2, 5]
[3, 6, 7, 10, 9, 4, 2, 5]
```

โปรแกรมนี้จะแสดงตัวอย่างการใช้ insertall() เพิ่มค่าหลายๆ ตัวลงใน List

การเพิ่มข้อมูลใน List ในภาษาอื่นๆ

Java

```
// Java Program to Add Elements to a List

// Importing all utility classes
import java.util.*;

// Main class
class GFG {

// Main driver method
public static void main(String args[])
{
// Creating an object of List interface,
// implemented by ArrayList class
        List<String> al = new ArrayList<>();

// Adding elements to object of List interface
// Custom elements
                al.add("Geeks");
                al.add("Geeks");
                al.add(1, "For");

// Print all the elements inside the
// List interface object
                System.out.println(al);
}
}
```

โปรแกรมนี้จะแสดงตัวอย่างการใช้ add() เพิ่มค่าลงใน List ของ Java โดยภาษา Java สามารถใช้ได้เพียงคำสั่ง add()

การเพิ่มข้อมูลใน List ในภาษาอื่นๆ

Python

```
numbers = [21, 34, 54, 12]

print("Before Append:", numbers)

# using append method
numbers.append(32)

print("After Append:", numbers)
```

โปรแกรมนี้จะแสดงตัวอย่างการใช้ append() เพิ่มค่าลงใน List ของ Python

```
numbers = [1, 3, 5]

even_numbers = [4, 6, 8]

# add elements of even_numbers to the numbers list
numbers.extend(even_numbers)

print("List after append:", numbers)
```

โปรแกรมนี้จะแสดงตัวอย่างการใช้ extend() เพิ่มค่าลงใน List ของ Python

Note : การใช้ append() และ extend() ใน Python จะเพิ่มค่าเข้าไปใน List ที่ตำแหน่งสุดท้ายเสมอ

จบการนำเสนอ